Biljardsimulator

Ek August
Månsson Marcus
Assadi Ramin
Johnson Oscar

2013-03-12

Sammanfattning

Syftet med detta projektarbete är att implementera en simplifierad men fungerade biljardsimulator. Valet av biljard beror på att biljard ingriper många intressanta aspekter i både fysik- och simuleringssyfte. Simuleringen delas in i fyra delar: translation, friktion, rotation och kollision. Dessa områden har först behandlats ur ett fysikaliskt perspektiv med hänsyn till bland annat den klassiska mekaniken, sedan har resultaten anpassats och implementerats i programmeringsspråket C# för skapa en visuell simulering. Resultatet av projektarbetet är att en enkel simulering gjordes med följd av ett visuellt realistiskt resultat. Simulationen har dock sina begränsningar då en del förenklande antaganden gjordes.

Innehåll

Figurer

Tabeller

Kapitel 1

Inledning

1.1 Inledning

Kursen TNM085 ”Modelleringsprojekt” VT2013 har som examinationsuppgift att implementera ett fysikaliskt system i datormiljö och området som har valts att behandla är biljard. Biljard är ett välkänt bordsspel världen över med många fysiska aspekter att ta hänsyn till. Biljard grundar sig även på mekanikens lagar inom fysiken vilket är ett välkänt område och gör det därmed intressant i simuleringssyfte.

1.2 Syfte

Syftet med projektarbetet är att genom användning av klassisk mekanik följt av implementering i en datormiljö lyckas skapa en simplifierad men ändå fungerande simulering av bordspelet biljard.

1.3 Avgränsning

Det har tidigare konstruerats biljardsimulatorer i både tre och två dimensioner.För att simplifiera simulationen har följande avgränsningar gjorts:

- När bollar kolliderar så överförs inte rotationen mellan de kolliderande bollarna.
- När bollar kolliderar så börjar de direkt rulla utan glidning.
- Stöten på köbollen sker alltid mitt på bollen alltså punkten där startvinkelhastigheten för bollen alltid är noll. Det finns alltså ingen möjlighet att utföra en under- eller överskruv.

Kapitel 2

Biljardfysik

2.1 Fysikalisk modell

Systemet är uppbyggt av 16 stycken objekt (biljardbollar) varav ett är det objektet (köboll) som tillförs en kraft (via biljardkön) för att kollidera med de resterande objekten som är placerade i en triangulär form. Vid kollisionen överförs momentkraften från köbollen till biljardbollarna via Newtons andra lag[1]. Systemet delas, i rapporten, upp i fyra delar: Translation, friktion, rotation och kollisioner mellan objekt och väggar.

2.2 Starthastighet

Först tilldelas köbollen en massa, m. En impuls, p, appliceras på bollen via kön enligt (2.1)[1] där $\vec{F}$ är den kraft som appliceras av biljardkön och Δt är den tid då kön är i kontakt med bollen.

$$p = \vec{F}\Delta t \tag{2.1}$$

Hastigheten hos bollen är då en funktion av dess massa och den impuls som appliceras på den enligt (2.2)[1].

$$\vec{v} = \frac{p}{m} \tag{2.2}$$

Ekvationerna är applicerbara förutsatt impulsen sker mitt på bollen parallellt med ytan.

2.3 Friktion och rotation

När biljardbollen rör sig över planet uppstår det friktion. Friktionen delas upp till två delar: Den kinetiska friktionen som uppstår när bollen direkt efter stöt glider längs underlaget och när bollen har övergått till rullning utan glidning då enbart en rullningsfriktion påverkar se figur 2.1. Det är även friktionen som ger upphov till hur bollen roterar[2].

[1] Sanemo, Ulf. (2010). Formelsamling TNE043 Mekanik och vågfysik. Linköping

[2] Freie Universitat Berlin, Institut für Informatik(2005) Like a rolling ball. Hämtad 2013-02-27, från http://robocup.mi.fu-berlin.de/buch/rolling.pdf

2.3.1 Kinetisk friktion

Det första stadiet av friktionens inverkan är när bollen går från vila till translation. Mikroskopiska ojämnheter i underlag och på biljardbollen kommer leda till att energi går förlorad som värmeenergi när dessa ytor rörs mot varandra.

När bollen rör sig på ytan fås att friktionskraften är proportionell mot massan enligt (2.3) där μ är friktionskonstanten, F_k är friktionskraften, m är massan och g är gravitationen. Den kinetiska friktionskraften kommer i sin tur att förändra bollens linjära och angulära hastighet.

$$F_k = \mu_k m g \tag{2.3}$$

Enligt Newtons andra lag (2.4) där $\dot{v}$ är derivatan av hastighetsvektorn kommer friktionskraften ge upphov till negativ acceleration hos bollens rörelse.

$$F_k = m\dot{v} \tag{2.4}$$

I (2.5) där I är tröghetsmomentet, R är radien och $\dot{\omega}$ är derivatan av den angulära hastigheten, beskrivs sambandet mellan friktionskraften och förändringen i angulär hastighet.

$$F_k R = I\dot{\omega} \tag{2.5}$$

Fallet med en sfärisk boll med tröghetsmoment I implicerar att förändringen kan uttryckas i dels linjär acceleration och vinkelacceleration enligt (2.6).

$$\dot{v} = -\frac{2}{5}R\dot{\omega} \Rightarrow \dot{\omega} = -\frac{5}{2R}\dot{v} \tag{2.6}$$

Detta visar att det råder ett samband mellan vinkelaccelerationen och den linjära accelerationen. Den negativa förändringen i den linjära accelerationen ger upphov till en positiv förändring i den angulära accelerationen. Sambandet är också obereonde av massan på bollen.

Uttrycket för en boll med starthastighet v_i och vinkelhastighet ω_i demonstreras i (2.7).

$$v_f - v_i = -\frac{2}{5}R(\omega_f - \omega i) \tag{2.7}$$

Sambandet mellan den linjära hastigheten och vinkelhastigheten när bollen roterar helt utan friktion kan beskrivas med (2.8).

$$v_r = R\omega_r \tag{2.8}$$

Då bollen tilldelas en starthastighet v_0 utan startrotation erhålls (2.9) från (2.8) och (2.7).

$$v_r - v_0 = -\frac{2}{5}v_r \tag{2.9}$$

Detta resulterar i det viktiga sambandet som beskriver att en boll, oberoende av radie och massa börjar rotera utan glidning efter att hastigheten har minskat enligt (2.10).

$$\vec{v} = \frac{5}{7}\vec{v_0} \tag{2.10}$$

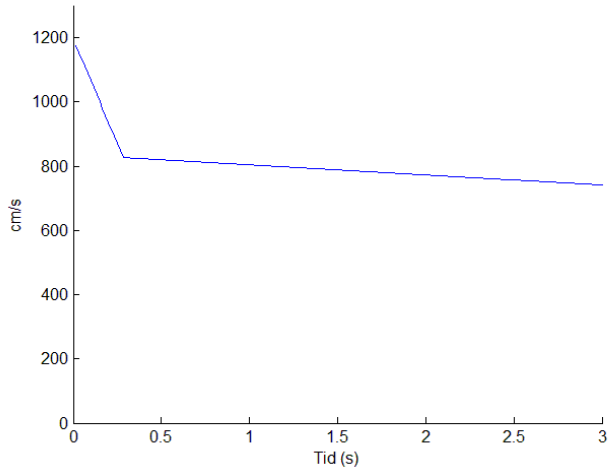
[2]Freie Universitat Berlin, Institut für Informatik(2005) Like a rolling ball. Hämtad 2013-02-27, från http://robocup.mi.fu-berlin.de/buch/rolling.pdf

[1]Sanemo, Ulf. (2010). Formelsamling TNE043 Mekanik och vågfysik. Linköping

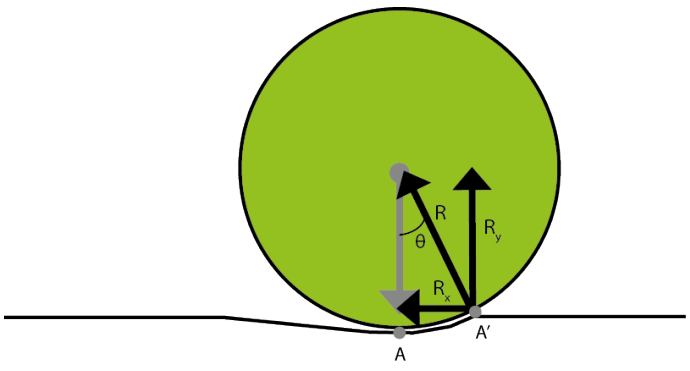
2.3.2 Rullningsfriktion

När bollen har övergått till rullning utan glidning finns det en annan typ av friktion som resulterar i att bollen, efter ett tag stannar. Denna friktion uppkommer då underlaget deformeras då bollen roterar på det och är mindre än den kinetiska friktionen. Deformationen kommer leda till att kontaktpunkten mellan bollen och underlaget inte ligger direkt under bollens tyngdpunkt. Reaktionskraften kan då komposantuppdelas så att det fås en kraftkomposant som motverkar bollens rörelseriktning. Figur 2.2 visar hur denna komposant kan uttryckas som (2.11)[3] beskriver.

$$R_x = \frac{5}{7}mg\theta \tag{2.11}$$



Figur 2.1: Grafen visar hastigheten hos en boll. Vid tidpunkten 0.4 s så syns en markant skillnad i lutningen vilket är tidpunkten då bollen går från rullning med glidning till rullning utan glidning. Hastigheten vid denna tidpunkt är ungefär $\frac{5}{7}$ av ursprungshastigheten.



Figur 2.2: Deformationen av biljardbordsduken där A är punkten direkt under tyngdpunkten, A' är den faktiska kontaktpunkten mellan boll och underlag.

[3]Spain and C Carnero University of Malaga, Spain(1995),Sliding and rolling: the physics of a rolling ball. Hämtad 2013-03-06 från http://www.engr.colostate.edu/ dga/pool/physics/Hierrezuelo_PhysEd_95_article.pdf

2.4 Kollision

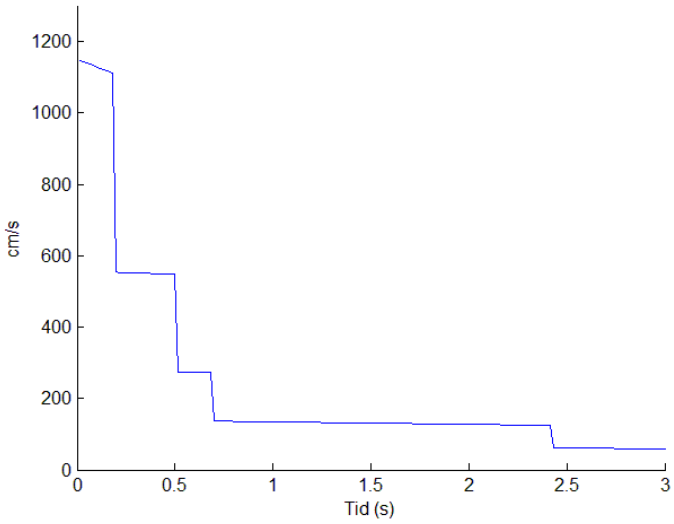
Eftersom kollisionerna mellan biljardbollarna är näst intill elastiska så bevaras nästan all energi vid stöten. Energin som går förlorad beror på den s.k. elasticitetskonstanten se (2.12)[1].

$$e|u_1 - u_2| = |v_1 - v_2| \tag{2.12}$$

- e är elasticitetskonstanten
- u_1 och u_2 är hastigheterna för bollarna innan kollisionen
- v_1 och v_2 är hastigheterna för bollarna efter kollisonen

När bollen istället krockar med en vägg är hastighen $u_2 = v_2 = 0$ eftersom att väggen inte har någon hastighet vilket leder till (2.13). Figur 2.3 visar hastighetsförändringen när sfären studsar mot väggen.

$$\vec{u}e = \vec{v} \tag{2.13}$$



Figur 2.3: Hastighetsförändring vid krockar med vägg

[1] Sanemo, Ulf. (2010). Formelsamling TNE043 Mekanik och vågfysik. Linköping

Kapitel 3

Modellering

3.1 Kollisionsteori

Vad beträffar kollisionsdetektering inom datorvetenskap så finns det två grundläggande sätt att hantera detta på; Posteriori och Priori. I biljard vill man använda dessa på sfärer ty simulatorn är uppbyggt av biljarbollar.

3.1.1 Posteriori

Posteriori[4] heter den metod som används för att kontrollera om avståndet mellan två objekt är mindre än dess kombinerade radie (sfärer). Om fallet förefaller sig att vara mindre så flyttas objekten till det momentet som inträffade precis innan kollisionen och nya hastighetsvektorer beräknas. Posteriori blir enklare att beräkna när objekten är sfärer eftersom avståndet att kontrollera alltid kommer att vara summan av dess radier.

3.1.2 Priori

Priori[4] är ett annat sätt att hantera kollisionsdetekteringen. Denna metod är mer korrekt ur fysikaliska hänseenden. Man förutser då s.k. kollisionslinjer som representerar den riktning en sfär kommer att röra sig längs med. Med hjälp av dess hastighetsvektor och momentana positionsvektor beräknas den tid objektet kommer att ha då det kolliderar med en annan sfär. Då avståndet mellan objekten beräknas till summan av dess radier implicerardet att objekten kolliderar. Genom applicering av (3.1)[5] kan tiden för detta fastställas. För att lättare kunna överblicka ekvationen har den delats upp i parametrarna; a; b och c som består av objektens hastighetsvektorer och positionsvektorer. Processen att hitta kollisionstiden mellan två sfärer upprepas sedan för alla sfärer. Detta leder till att alla kollisioner är detekterade.

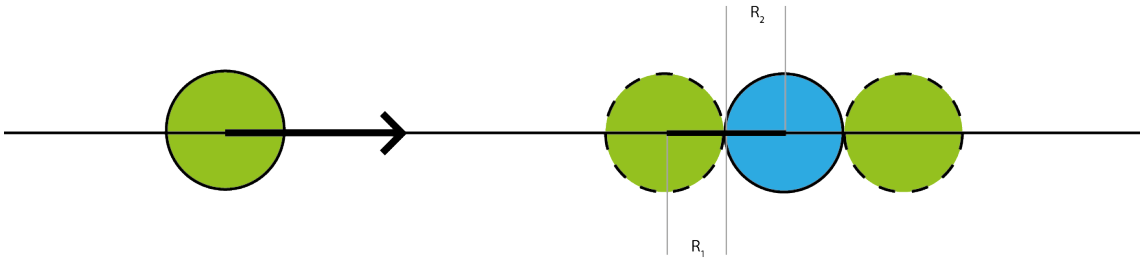
$$\begin{aligned} a &= (\vec{v_1} - \vec{v_2})^2 \\ b &= 2(\vec{v_1} - \vec{v_2})(\vec{p_1} - \vec{p_2}) \\ c &= (\vec{p_1} - \vec{p_2})^2 \end{aligned}$$

[4]Ericson, Christer. Real-time Collision Detection. Elsevier (2005)

[5]NCSA Illinois (2003), The math and physics of billjard. Hämtad 2013-02-16, från http://archive.ncsa.illinois.edu/Classes/MATH198/townsend/math.html

$$t = \frac{-b \pm \sqrt{b^2 - 4a(c - 4r_1 r_2)}}{2a} \tag{3.1}$$

Det finns tre utfall för andragradsekvationen; ingen lösning, en lösning eller två lösningar. Utfallet för ingen lösning innebär att objekten aldrig kommer att kollidera. Vid en lösning innebär det att de endast kommer att tangerar varandra. Utfallet för två lösningar anses vara mest intressant eftersom det innebär att kollision kommer att förekomma. Den lösning (tidpunkt) som är minst är alltid den egentliga kollisionen. Den andra tidpunkten är då sfärerna har passerat igenom varandra, som kan ses i figur 3.1 och är därför inte intressant ty det inte är ett möjligt utfall.



Figur 3.1: Den gröna bollens streckade motsvarigheter står för de två lösningar som beräknas ur (3.1)

I det fall då det finns ett flertal sfärer i ett objekts kollisionslinje kommer det resultera i ett flertal predikterade kollisioner. Den kollisionen med minst värde (tid) enligt (3.1) är den med högst prioritet eftersom den kommer att förefalla först. Efter att den första kollisionen har inträffat bildas det nya förutsättningar för objekten och baserat på dessa startas processen om tills frametiden är förbrukad. Nästkommande kollisioner beräknas i sådana fall i nästa tidsuppdatering.

3.1.3 Krockmomentet

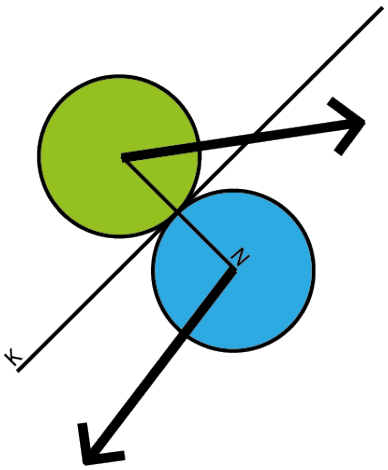
Vid kollisionstillfället måste nya hastighetsvektorer beräknas. Biljard är väldigt lätthanterligt ty de nya riktningarna kommer vara vinkelräta mot varandra efter kollisionen. Detta med undantaget för tillfället om en boll står still, man får då en front mot front kollision.

När kollisionen inträffar är sfärernas position inte exakt intill varandra (på grund av datorns numeriska begränsning) och måste därför förflyttas intill varandra. Det görs genom att flytta respektive sfär i sin hastighetsvektor multiplicerat med tiden till krocken. Utifrån dessa positioner kan kollision utföras.

Nästa steg är att att ta ut kollisionsplanet, kollisionsplanet är det plan sfärerna kommer att ha en elastisk stöt mot, visas i figur 3.2. För att få fram kollisionsplanet utnyttjas normalplanet. Normalplanet är avståndet mellan sfärerna i vektorform. Eftersom simulering i praktiken sker i ett tvådimensionellt plan kan kollisionsplanet lätt bestämmas[7].

[6]NCSA Illinois (2003), The math and physics of billjard. Hämtad 2013-02-16, från http://archive.ncsa.illinois.edu/Classes/MATH198/townsend/math.html

[7]Simon P Stevens(2011) Basic Collision Physics with Vectors. Hämtad 2013-02-23, från http://simonpstevens.com/Articles/VectorCollisionPhysics



Figur 3.2: Kollisionsplanet mellan två sfärer

$$Normalplanet : (x, y, z)$$
$$Kollisionsplanet : (-y, x, z)$$

Nu kan hastighetsvektorerna för de båda sfärerna delas in i komponenter längs med normalplanet och kollisionsplanet. Därefter beräknas de hastighetsvektorer som kommer antas i n-led efter kollisionen med hjälp av (3.2)[1] för momentet och (3.3)[1] för den kinetiska energin.

$$p = m\vec{v} \tag{3.2}$$

$$E_k = \frac{1}{2}m\vec{v}^2 \tag{3.3}$$

I kombination av sfärernas massor och dess hastighetsvektorer impliceras (3.4).

$$\overrightarrow{v_1 after} = \frac{\vec{v_1}(m_1 - m_2) + 2m_2\vec{v_2}}{m_1 + m_2} \tag{3.4}$$

3.2 Implementering

3.2.1 Stöt

Ett samband mellan kraft och hastighet ges enligt ekvation (2.2)[1]. Vid implementeringen av denna har det valts att en stöt sker under tiden 5 ms. I en funktion med kraft som inparameter beräknas hastigheten för bollen enligt *Kodstycke 1*.

[7]Simon P Stevens(2011) Basic Collision Physics with Vectors. Hämtad 2013-02-23, från http://simonpstevens.com/Articles/VectorCollisionPhysics

[1]Sanemo, Ulf. (2010). Formelsamling TNE043 Mekanik och vågfysik. Linköping

```
void cuepush(Vector3 theForce)
{
     initVelocity = 100.0f * theForce * (0.005f) / mass;
     velocity= initVelocity;
}
```

Kodstycke 1

- `theForce` är en tredimensionell vektor som beskriver den applicerade kraften. Det är dock enbart x och y komposanterna som används för bollens translation.
- `initVelocity` står för starthastigheten hos bollen
- `100.0` är skalfaktorn som skalar hastigheten från m/s till cm/s.
- `0.005` är det värde som beskriver tiden som bollen är i kontakt med kön.
- `mass` är bollens vikt.
- `velocity` är hastigheten som tilldelas bollen.

3.2.2 Translation

Translation sker genom att addera hastigheten multiplicerat med tidssteget på positionen. Beräkningen av detta visas genom *Kodstycke 2*.

```
position +=(velocity/60.0f)
```

Kodstycke 2

3.2.3 Friktion

Friktionskraften vid glidning beräknas via (2.3). Sedan används samma resonemang som vid stöten, alltså att en kraft appliceras på bollen men nu till den motsatta riktningen av bollens färdriktning. Tiden det tar för impulsen är ändrad till värdet av en frame. När bollen sedan övergår till rotation utan glidning så börjar rullningsfriktionen verka enligt (3.5)[8]. Denna friktionskraft appliceras då istället för den föregående kinetiska friktionskraften.

$$\vec{R_x} = \frac{5}{7} mg\theta \tag{3.5}$$

- R_x är en x-komposant av grundreaktionskraften.
- θ är vinkeln mellan tyngdkraften och normalkraften från underlaget. I simuleringstillfället: 0.01 radianer.

[8]Spain and C Carnero University of Malaga, Spain(1995), Sliding and rolling: the physics of a rolling ball. Hämtad 2013-02-05 från http://www.engr.colostate.edu/ dga/pool/physics/Hierrezuelo_PhysEd_95_article.pdf

3.2.4 Rotation

För varje tidssteg beräknas rotationen för varje boll. Första steget är att räkna ut kryssprodukten mellan hastighetsvektorn och axeln som är vinkelrät mot skärmens plan enligt (3.6). Vektorn normaliseras sedan och används som rotationsaxel för bollen.

$$\overrightarrow{rotAxel} = \vec{v} \times \vec{z} \tag{3.6}$$

Vinkelhastigheten beräknas enligt (3.7) då bollen roterar med glidning. När hastigheten sedan uppfyller kravet att ($\vec{v} = (5/7)\vec{v_0}$)[8] så övergår bollen till irullning utan glidning och vinkelaccelerationen beräknas då med hjälp av (3.8). När vinkelhastigheten är beräknad adderas den till rotationsvärdet som sedan används när bollen ritas ut på skärmen.

$$\omega_2 = -\frac{5}{2r}(\vec{v_2} - \vec{v_1}) + \omega_1 \tag{3.7}$$

- $\vec{v_2}$ är hastigheten i nuvarande frame.
- $\vec{v_1}$ är hastigheten i föregående frame.
- ω_2 är vinkelhastigheten i nuvarande frame.
- ω_1 är vinkelhastigheten i föregående frame.

$$\omega = \frac{\vec{v}}{R} \tag{3.8}$$

3.2.5 Kollision

Efter att ha beräknat alla objektens kollisionstider, som beskrivs i delen om kollisionsteori, och lagrat dessa i objekten sorteras objektlistan beroende på kollisionstid. De objekten som sorteras överst är de med lägst kollisionstid vilket betyder att de kommer att kollidera först. Med hjälp av kollisionstiden är det möjligt att beräkna vilken frame de kommer kollidera i. Är det den nuvarande frame'en förflyttas objekten intill varandra till precis innan kollisionstillfället. Detta visas i *Kodstycke 3*

```
if (collisionFrame == 0.0f){
   billiardBalls.ElementAt(index + 1).Position
   += billiardBalls.ElementAt(index + 1).Velocity * collisionTime;
   billiardBalls.ElementAt(index).Position
   += billiardBalls.ElementAt(index).Velocity * collisionTime
...
```

Kodstycke 3

[8]Spain and C Carnero University of Malaga, Spain(1995), Sliding and rolling: the physics of a rolling ball. Hämtad 2013-02-05 från http://www.engr.colostate.edu/ dga/pool/physics/Hierrezuelo_PhysEd_95_article.pdf

Därefter kan bollarnas nya hastigheter beräknas,första steget är att dela upp bollarnas hastigheter i komposanter längs normalplanet och kollisionsplanet enligt *Kodstycke 4*.

```
float norm_vel_i = Vector3.Dot(normalPlane, theBalli.Velocity);
float coll_vel_i = Vector3.Dot(collisionPlane, theBalli.Velocity);
```

Kodstycke 4

Nu kan den fysikaliska egenskapen utnyttjas att accelerationen i det led kollisionen sker (dvs normalplanet) är den enda som påverkas. Med hjälp av ekvationen (3.3) för kinetisk energi, ekvationen (3.2) för momentum och antagandet att det inte förekommer någon energiförlust kan man få den nya hastigheten i n-led. I ett senare skedde kompenserar vi energiförlusten genom att skala med elasticitetskoefficienten. Implementationen visas i *Kodstycke 5*.

```
float norm_vel_i_after = ((norm_vel_i * (theBalli.Mass - theBallj.Mass))+
(2 * theBallj.Mass * norm_vel_j)) / (theBallj.Mass + theBalli.Mass);
float norm_vel_j_after = ((norm_vel_j * (theBallj.Mass - theBalli.Mass))+
(2 * theBalli.Mass * norm_vel_i)) / (theBallj.Mass + theBalli.Mass);
```

Kodstycke 5

Tillsammans med k-ledets hastighetsvektor summeras hastighetsvektorerna och tilldelas Objekten. Om ingen mer kollision ska ske under denna tidsuppdatering påbörjas processen från translation av samtliga objekt. Annars börjas processen om från kollisionsdetekteringen med de nya förutsättningnara.

Om objektet skulle kollidera mot en vägg i spelplanen beräknas dess nya hastighet med hjälp av reflektionsformeln (3.9) där a är infallande hastighet och n normalen till planet som träffas. I koden skalas hastigheten med energiförlusten enligt elastisitetskofficienten;
`CoRBallToWall: 0.5` som visas i *Kodstycke 6*.

$$r = a - 2a \bullet n * n \tag{3.9}$$

```
theBalli.Velocity = (theBalli.Velocity - 2 * Vector3.Dot
(theBalli.Velocity, Vector3.Right) * Vector3.Right) * CoRBallToWall;
```

Kodstycke 6

3.3 Använda numeriska värden

Tabell 3.1: De värden som används vid implementation

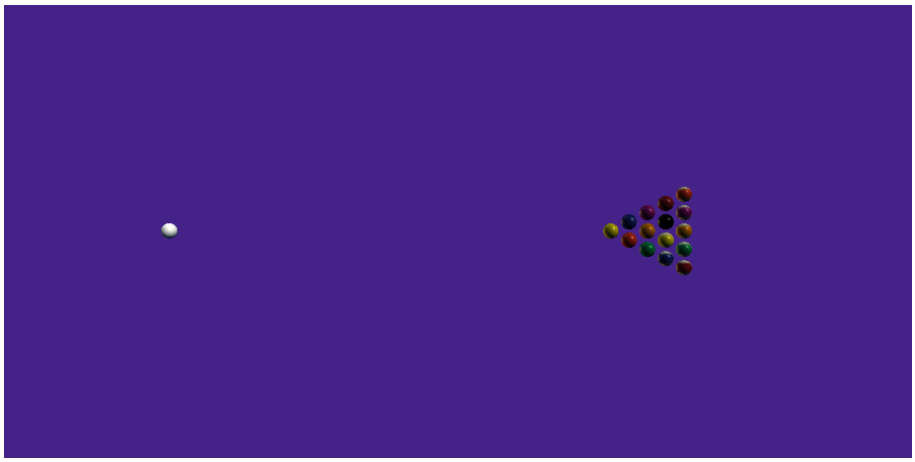
Bollens radie:	2.5 cm
Bollens vikt:	0.15 kg
Elasticitetskofficient, boll-mot-boll:	0.97
Elasticitetskofficient, boll-mot-vägg:	0.5
Mått på spelplan:	224x112 cm
Impuls stöttid:	0.005 sekunder

Kapitel 4

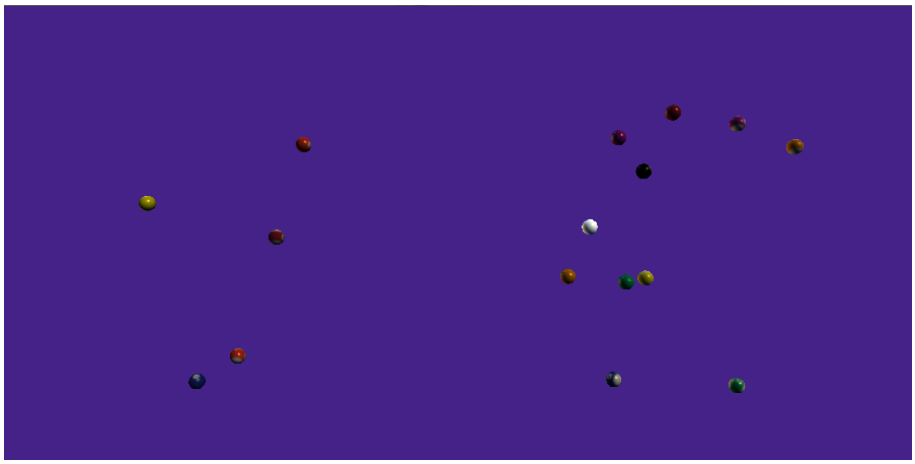
Resultat

Genom att använda begrepp så som translation, friktion, rotation och kollision har det skapats en biljardsimulator som återger ett trovärdigt visuellt resultat baserat på klassisk mekanik. Vid naturliga stötar ger simulatorn ett realistiskt resultat på kollisioner mellan bollar. Bollarna bromsar även in på ett sätt som följer fysikens lagar samtidigt som det ser korrekt ut. Rullningen hos bollarna fungerar enligt en något simplifierad modell då en del förenklingar gjordes angående rotationsöverföringen mellan bollar vid kollision vilket i sin tur ledde till en del begränsningar i simuleringen. Dessa begränsningar diskuteras mer ingående i nästa del av rapporten.

4.1 Resultatbilder



Figur 4.1: Skärmdump på startpositionen innan stöt



Figur 4.2: Skärmdump då biljardbollarna ligger i vila efter stöt.

Kapitel 5

Diskussion

Kollisionsdetekteringen spelar en stor roll i simuleringen. Den första metoden posteriori har sina för och nackdelar. En fördel är att den är enkel att implementera. Avståndet jämförs med radiernas storlek, om avståndet understiger radiernas storlek utförs nödvändiga operationer. Det blir också lätt att implementera för sfärer med olika storlekar. Nackdelen med denna hantering är att det uppstår problem med hanteringen vid höga hastigheter på bollarna.

Första problemet är att bollarna kan ha kommit så pass långt in i varandra att när de studsar ifrån varandra syns det att de gått in i varandra och studsen ser onaturlig ut. Andra problemet uppstår av samma anledning men istället hinner bollarna passera varandra innan krocken och man får helt enkelt ingen krock, bollarna passerar endast igenom varandra.

Prioris kollisionsdetektering utesluter det fel som posteriori skapar men andra problem uppstår. Antag att en grupp bollar ligger stilla och väldigt nära varandra. Där på skjuts en höghastighets boll som kolliderar med denna grupp. I och med att bollarna ligger väldigt nära varandra kan flera bollar kollidera under samma tidsframe. Den första kollisionen kommer ske felfritt, men nästa ”missas” eftersom den skulle skett i samma uppdatering. En lösning på detta är att införa en rekursivitet på kollisionsdetekteringen. Efter hantering av första kollisionen kollas alla bollar igen för att se om det uppstår en kollision i samma frame, annars kommer nästa frame.

Det som ger upphov till problemen i både posteriori- och priori-hanteringen är det tidsdiskreta steg som datorn använder sig av. Datorn använder sig av 60 frames per second, uppdateringen sker alltså 60 gånger per sekund vilket inte är tillräcklig i något av fallen. Hade det gått att uppnå en tidskontinuerlig uppdatering eller en orimligt hög fps där datorn ändå hunnit med beräkningar under frametiden, då hade båda metoderna fungerat felfritt.

Simulationens starka sida är att det är en enkel biljardsimulator med realistiska visuella resultat. Vår grundläggande hantering av translation, rotation och kollisioner ger tillsammans ett intryck av en fungerande simulator. Nackdelar i simulationen är att den inte tar hänsyn till vissa aspekter som för en riktig biljardspelare skulle ha oerhörd tyngd. Med detta menar vi först och främst att vi har negerat rotationsöverföringen vid kollisioner. Detta har i sin tur lett till att vi var tvungna att göra en ytterligare förenkling, nämligen förenklingen att bollar vid kollision direkt börjar sin fas av rullning utan glidning. I en verklig krock mellan två biljardbollar så överförs inte bara linjär hastighet utan även vinkelhastigheten. Med hänsyn till dessa avseenden skulle en krock med högst sannolikhet få ett annat utseende än den från vår simulation. Vi valde även att inte ge användaren möjlighet att utföra under- eller överskruvar så även den

intressanta aspekten bort ur simulationen. Skruvarna kan ge ett helt annat utlopp på en stöt än vad en rak stöt skulle ge.

Vi anser att slutresultatet av projektet skulle kunna användas i t.ex. ett biljardspel. Begränsningarna i simulatorn skulle kanske anses som väldigt stora för en professionell spelare men för en amatörspelare skulle simulatorn kunna ge ett tillräckligt realistiskt resultat för att kunna användas i nöjessyfte. Simulatorn är även skapad på ett sätt så att parametrar så som massor och radier på bollar går att ändra och därmed så skulle man t.ex. kunna lägga in bollar av olika storlekar och massor och låta dessa kollidera mot varandra. Parametrar för elasticitetskonstanter kan också manipuleras och på så sätt kan simulationen av olika material skapas. Simulatorn är nu gjord för biljard men med modifieringar skulle den kunna anpassas till ett annat bollspel exempelvis ishockey eller bli en mer generell bollsimulator.

Kapitel 6

Referenslista

Sanemo, Ulf. (2010). Formelsamling TNE043 Mekanik och vågfysik. Linköping

Freie Universitat Berlin, Institut für Informatik(2005) Like a rolling ball. Hämtad 2013-02-27, från http://robocup.mi.fu-berlin.de/buch/rolling.pdf

Spain and C Carnero University of Malaga, Spain(1995),Sliding and rolling: the physics of a rolling ball. Hämtad 2013-03-06 från
http://www.engr.colostate.edu/ dga/pool/physics/Hierrezuelo_PhysEd_95_article.pdf

Ericson, Christer. Real-time Collision Detection. Elsevier (2005)

NCSA Illinois (2003), The math and physics of billjard. Hämtad 2013-02-16, från
http://archive.ncsa.illinois.edu/Classes/MATH198/townsend/math.html

Simon P Stevens(2011) Basic Collision Physics with Vectors. Hämtad 2013-02-23, från
http://simonpstevens.com/Articles/VectorCollisionPhysics