Simuleringsverktyg för en fotbolls projektilbana

Johan Forslund
Fredrik Johansson
Adam Lindfors
Sebastian Delgado

8 november 2019

Sammanfattning

I denna rapport redovisas de metoder som har använts för att skapa en simulering av en fotbolls bollbana. För att skapa en matematisk modell så har differentialekvationer tagits fram med hänsyn till de fysikaliska fenomen som uppstår. Dessa har sedan verifierats med hjälp av simuleringsverktyg och slutligen visualiserats i en tredimensionell datormiljö. Resultatet är en realistisk fotbollssimulator där användaren kan ställa in fysikaliska parametrar och se hur bollens projektilbana påverkas.

Innehållsförteckning

Figurer

Tabeller

Kapitel 1

Inledning

Fotboll är världens största sport och fortsätter att växa[9]. Efterfrågan på metoder som använder mjukvara för att analysera fotboll ökar. I detta projekt skapas en sådan mjukvara, där fokus ligger på att visualisera en fotbolls bollbana genom luften.

1.1 Syfte

Syftet med projektet är att ta fram en simulator som ger kunskap och förståelse om hur en fotbollsspark fungerar samt hur olika krafter och effekter påverkar bollbanan.

1.2 Mål

Målet med projektet är att framställa en fotbollssimulator inom den tidsramen som finns. Simulatorn ska följa fysikens lagar och ska vara lätt att tyda för användaren. Användaren ska själv kunna ändra olika parametrar såsom utgångshastighet och vinklar.

1.3 Avgränsningar

För att uppnå projektets mål under tidsramen så gjordes vissa avgränsningar:

- Simulatorn tar ej hänsyn till att fotbollen i ett realistiskt fall får dess initialvärden från en spark. Det vill säga att värden så som rotationsriktning och vinkelhastighet egentligen ska definieras utifrån fotens träffposition och kraft på bollen.

- Vid en realistisk spark färdas bollen med fotbollsspelarens fot vilket ger variabla värden under tiden då bollen färdas med foten. Detta är ej implementerat i detta projekt. Avgränsningen innebär alltså att simulatorn ej tar hänsyn till fotbollsspelarens ben och fots rörelse som verkar på bollen.

- Under bollens färd i luften simuleras luftmotstånd och Magnuseffekten på bollen. Dessa två krafter är de som har störst inverkan på projektilbanan [1]. För att slutföra projektet i tid har resterande fysikaliska egenskaper uteslutits, exempelvis vind och deformation av boll. I verkligheten minskar även vinkelhastigheten med tiden, detta är inte något som tas i hänsyn i den framtagna modellen eftersom dess effekt är så pass liten

- Bollens studs är begränsad till ytor som är ortogonala mot minst ett av de plan som spänner upp koordinatsystemet. Bollens hastighet behöver då bara ändra riktning i den axel som motsvarar det träffade planets normal.

Kapitel 2

Metod

Då målet var att skapa en så realistisk simulator som möjligt krävdes en grundlig förundersökning. I denna förundersökning togs relevant fakta fram och de fysikaliska krafter som verkar på en boll studerades noggrant. Eftersom projektet bestod av flera tekniska delar var det nödvändigt att dela in arbetet i olika sektioner för ett bra arbetsflöde. Dessa sektioner var:

1. Framtagning av matematiska modeller.
2. Simulering och tester av de matematiska modellerna.
3. Grafisk implementation av simuleringarna.

Hur arbetet utfördes inom varje sektion kommer att beskrivas i detalj under respektive underrubrik.

2.1 Matematisk modell

En simpel modell av en bollbana kan beskrivas med hjälp av en kastparabel i ett tvådimensionellt plan, se (2.1) och (2.2). Projektilen har en utgångshastighet samt en utgångsvinkel och påverkas enbart av gravitationen.

$$x(t) = v_0 t \cos\alpha \tag{2.1}$$

$$y(t) = v_0 \sin\alpha - \frac{1}{2} g t^2 \tag{2.2}$$

x och y är bollens position i x- respektive y-led, v_0 är bollens utgångshastighet, t är tiden efter sparken, α är bollens utgångsvinkel och g är gravitationskonstanten. Denna modell gäller dock bara vid vakuum och tar ej hänsyn till bollens massa. I verkligheten så skulle ett liknande system enbart påverkas av gravitationskraften, enligt (2.3).

$$F_g = mg \tag{2.3}$$

m är bollens massa.

Modellen utökades sedan för att också ta hänsyn till luftmotståndet, där luftmotståndets motståndskraft kan ses i (2.4).

$$F_d = \frac{1}{2} C_D \rho A v^2 \tag{2.4}$$

F_d är motståndskraften, C_D är motståndskraftskoefficienten, ρ är luftens densitet och A är bollens tvärsnittsarea.

Motståndskraftskoefficienten är en siffra som kvantifierar motståndet på ett objekt i luft. Ju större siffra desto större är motståndet. Denna siffra kan tas fram med hjälp av experiment men i detta projekt så valdes det att sättas till 0.25[6].

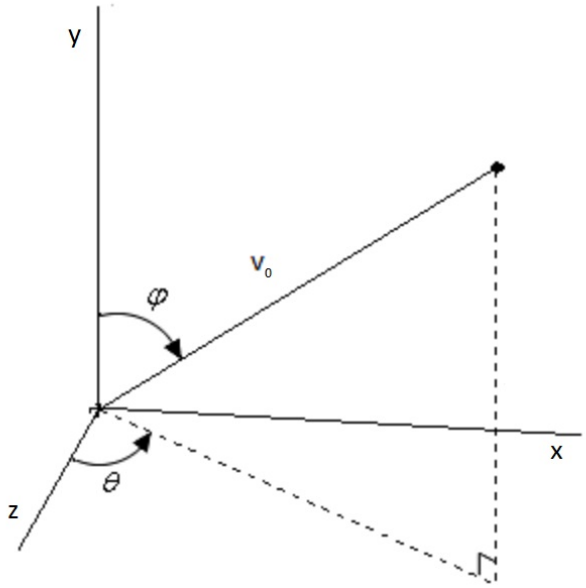
Modellen utökades sedan ytterligare för att ta hänsyn till tre dimensioner. Detta gjordes genom att skriva om projektilens initialhastighet med sfäriska koordinater. Projektilens initialhastighet berodde då på två vinklar och storleken av utgångshastigheten, vilket redovisas i (2.5), (2.6) och (2.7).

$$v_{0_x} = v_0 \sin\phi \sin\theta \tag{2.5}$$

$$v_{0_y} = v_0 \cos\phi \tag{2.6}$$

$$v_{0_z} = v_0 \sin\phi \cos\theta \tag{2.7}$$

ϕ är vinkeln mellan y-axeln och v_0-vektorn och θ är vinkeln mellan z-axeln och v_0-vektorns projektion i xz-planet. I projektet användes ett högerorienterat koordinatsystem där xz-planet representerade marken, enligt Figur 2.1.



Figur 2.1: Det högerorienterade koordinatsystem som användes.

När ekvationerna hade anpassats till tre dimensioner kunde Magnuseffekten implementeras. Magnuseffekten är namnet på den effekt som får en roterande boll att accelerera vinkelrätt mot rörelseriktningen och rotationsriktningen [3]. Bollen får en rörelse av Magnuskraften som påverkar bollen beroende på vinkelhastighet och rotationsriktning. Ekvationen (2.8) visar formeln för denna kraft.

$$F_L = \frac{1}{2} C_L \rho A v^2 \frac{\overline{\omega} \times \overline{v}}{|\overline{\omega} \times \overline{v}|} \tag{2.8}$$

F_L är Magnuskraften och C_L är Magnuskoefficienten, $\overline{\omega}$ är bollens vinkelhastighetsvektor och $\overline{v}$ är bollens hastighetsvektor. Kryssprodukten används för att få Magnuskraften ortogonal mot bollens rotationsaxel och hastighet. Därefter normeras kryssprodukten för att inte påverka kraftens storlek.

Magnuskraftskoefficienten kan jämföras med lyftkoefficienten som främst används inom aerodynamiken. Skillnaden är att istället för att enbart bero på hastigheten så beror Magnuskraftskoefficienten även på objektets rotation [1]. Denna koefficient räknades ut genom (2.9).

$$C_L = \frac{\omega a C_s}{v_0} \tag{2.9}$$

C_s är en rotationskonstant, vanligtvis för en fotboll vald till 0,5 [1].

Alla krafter som påverkar projektilen summerades sedan vilket gav (2.10).

$$F_{tot} = F_g + F_d + F_L \tag{2.10}$$

F_{tot} är den resulterande kraften. Newtons andra lag utnyttjades sedan vilket gav (2.11).

$$F_g + F_d + F_L = ma \tag{2.11}$$

Därefter substituerades F_g, F_d och F_L i (2.11) med (2.3), (2.4) respektive (2.8). Detta resultat gav differentialekvationer i x-,y- och z-led [2], vilket redovisas i (2.12), (2.13) och (2.14).

$$m\ddot{x} = -k_D\dot{x}\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} + k_L(\omega_y\dot{z} - \omega_z\dot{y}) \tag{2.12}$$

$$m\ddot{y} = -mg - k_D\dot{y}\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} + k_L(\omega_z\dot{x} - \omega_x\dot{z}) \tag{2.13}$$

$$m\ddot{z} = -k_D\dot{z}\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} + k_L(\omega_x\dot{y} - \omega_y\dot{x}) \tag{2.14}$$

$\ddot{x}$, $\ddot{y}$ och $\ddot{z}$ är bollens acceleration i respektive position, ω_x, ω_y och ω_z är vinkelhastighetens riktning i respektive koordinat. Notera att gravitationskraften enbart ger bidrag i y-led. k_D och k_L är hjälpvariabler som användes för att förenkla uträkningarna. Dessa kan ses i (2.15) och (2.16).

$$k_D = \frac{1}{2}C_D\rho_a A \tag{2.15}$$

$$k_L = \frac{1}{2}C_L\rho A(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)\frac{1}{|\overline{\omega} \times \overline{v}|} \tag{2.16}$$

$\dot{x}$, $\dot{y}$ och $\dot{z}$ är bollens hastighet i respektive dimension. Differentialekvationerna i (2.12), (2.13) och (2.14) går inte att lösa analytiskt och istället användes Eulers stegmetod för att approximera dess värden. Metoden följer ekvationen i (2.17) och ger approximativa tidsdiskreta värden av en funktion $u(t)$ som uppfyller en given differentialekvation.

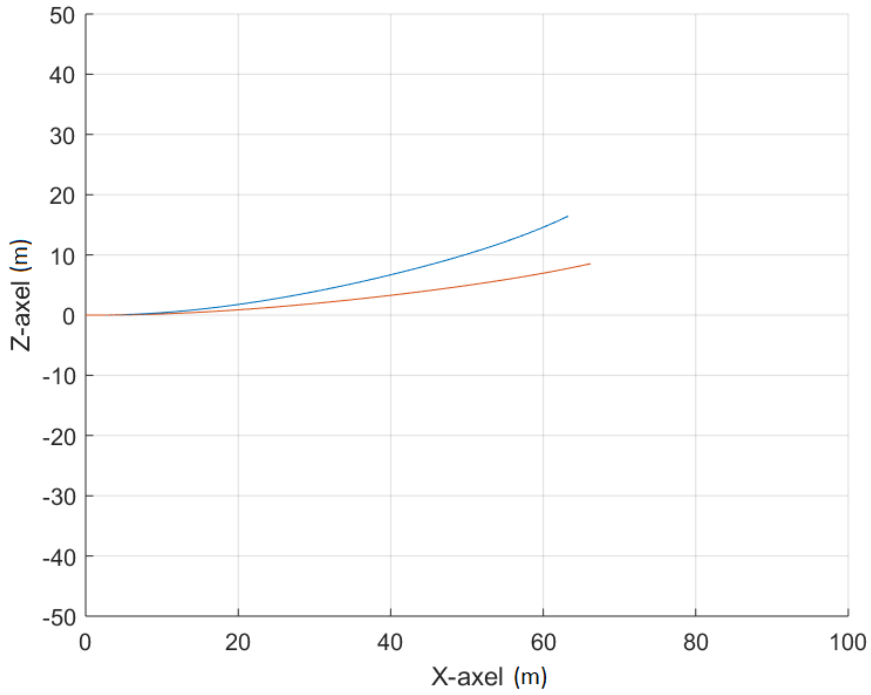
$$u_{n+1} = u_n + hf(u_n) \tag{2.17}$$

h är steglängden och påverkar noggrannheten i approximationen. u_0 kan beräknas genom (2.5), (2.6) och (2.7) för respektive dimension vilket ger u_1. Därefter beräknas u_k i en iterativ process där $k = 0, 1, 2, 3...n$. På detta vis bestämdes bollens hastighet $(\dot{x}, \dot{y}, \dot{z})$ och position (x, y, z).

2.2 Simulering

För att kontrollera att de framtagna matematiska modellerna var korrekta gjordes ett antal simuleringar i *MATLAB*. Dessa simuleringar visar bollens projektilbana beroende på modellens in-parametrar. Ett exempel på en sådan simulering av Magnuseffekten kan ses i Figur 2.2. Figuren visar projektilbanan för två olika skott där bollen har identiska in-parametrar förutom att den blå linjens boll har dubbelt så hög vinkelhastighet än den röda linjens boll.

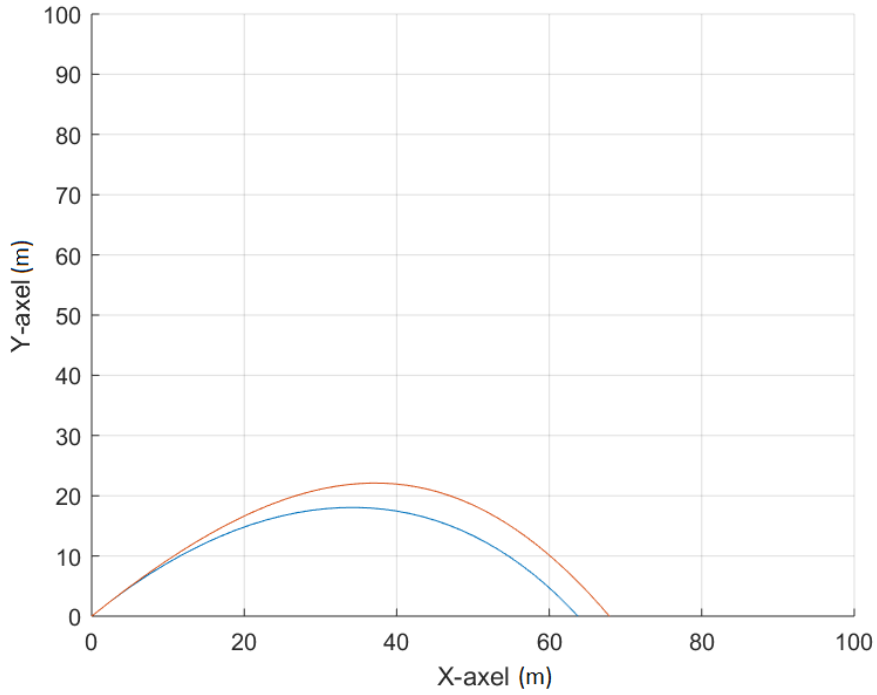
Samtliga simuleringar som utfördes i MATLAB avslutades när bollen nått en negativ position längs y-axeln, med andra ord när bollen befann sig under marken. Motiveringen för denna lösning var att implementationen av studs inte var prioriterad i detta skede.



Figur 2.2: Simulering av Magnuseffekten av två skott sett från fågelperspektiv.

Figur 2.2 visar att en boll med högre vinkelhastighet ”skruvas” mer vilket är korrekt. Alltså påvisar detta att den matematiska modellen för Magnuseffekten troligtvis fungerar. På ett liknande sätt kan resterande modeller testas genom simuleringar. En simulering av en boll med överskruv jämfört med en boll utan skruv visas i Figur 2.3. I figuren representerar den blå linjen den överskruvade bollens projektilbana. Denna skruv leder till att den blå linjens boll dyker på grund av Magnus-effekten.

Den sista funktionaliteten som adderades till simuleringen var bollens studs. Hastigheten efter studsen beräknas genom att bollens hastighetsvektor multipliceras elementvis med studskoefficienten. Studskoefficienten för en boll som träffar medellångt gräs är ungefär 0,6 och för hårda material ungefär 0,8 [1]. För att bollen ska byta riktning ändras tecknet hos hastighetskoordinaten i den riktning som kollisionen sker. Tillsist beräknas även förändringen av rotationsriktningen samt vinkelhastigheten. Rotationsriktningen beräknas med (2.18), denna formel är inte fysikaliskt korrekt utan den används enbart för att visualisera att bollens rotation ändras vid studs. Vinkelhastigheten beräknas genom multiplikation med en godtycklig konstant. I Sektion 1.3 beskrivs avgränsningar som gjordes för bollens studs.



Figur 2.3: Simulering av en överskruvad bolls effekt på projektilbanan sedd vinkelrätt mot bollens utgångshastighet i x-led.

$$\hat{\omega} = \frac{(-v_z, v_y, v_x)}{|\overline{v}|} \tag{2.18}$$

$\hat{\omega}$ är bollens rotationsriktning och $\overline{v}$ är bollens hastighetsvektor.

2.3 Grafisk implementation

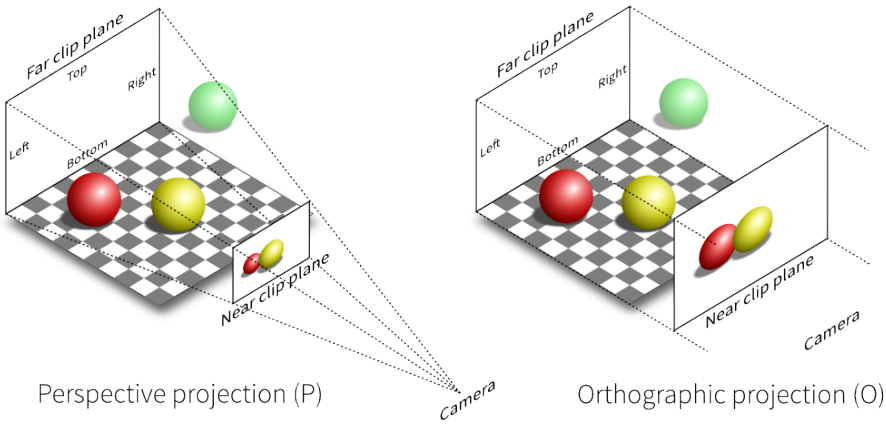
Eftersom simuleringen beror på tre dimensioner så behövdes ett ramverk för att rendera grafik i 3D. I detta projekt användes OpenGL, vilket är ett *API* (applikationsprogrammeringsgränssnitt) som används för att skriva applikationer med datorgrafik. Som programmeringsspråk valdes C++ för att maximera applikationens prestanda. OpenGL är kraftfullt men erbjuder inte mycket färdigt material. Det kräver att utvecklaren själv bygger upp de fysikaliska modeller som påverkar hur omvärlden upplevs, vilket beskrivs nedan.

2.3.1 Kamera och perspektiv

I grunduppsättningen av OpenGL så visas objekt i scenen från en konstant position i rummet. Dessutom saknas perspektivprojektioner vilket gör att ett objekt upplevs vara lika stort oberoende av dess avstånd till betraktaren. Detta löses genom ett antal olika koordinattransformationer där 4×4-matriser används.

Den första transformationen påverkar hur objekt rör sig i rummet relativt till en fixerad kamera. Den andra transformationen tillåter kameran att röra sig i rummet. Den tredje och sista transformationen påverkar kamerans perspektiv. Istället för att projicera objekt parallellt

så projiceras objekt längs en trunkerad pyramid, se skillnaden i Figur 2.4. Detta gör att objekt som ligger närmare kameran kommer att täcka fler pixlar på skärmen och därmed upplevas större.



Figur 2.4: Skillnaden mellan perspektivprojektion och parallellprojektion.

När alla dessa matriser multipliceras ihop så skapas ett system där användaren kan förflytta sig i rummet och uppleva objekt på ett verklighetstroget sätt.

2.3.2 Ljussättning

För att ge en bra uppfattning om objektens djup så krävdes en god ljussättning. För detta användes Phongs reflektionsmodell som ger scenen ett *ambient*, *diffust* och *spekulärt* ljus. I detta projekt valdes att utesluta det spekulära ljuset eftersom scenen enbart bestod av matta ytor. Ekvationen för reflektionsmodellen visas i (2.19).

$$I = k_a L_a + K_d L_d (\hat{N} \cdot \hat{L}) \tag{2.19}$$

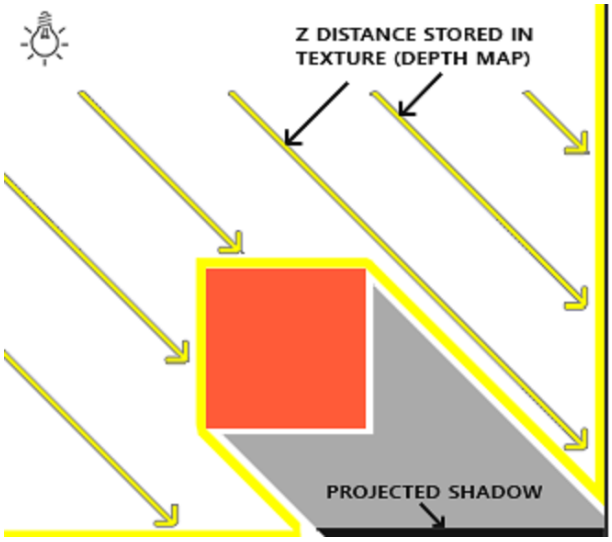
Där I är det totala ljuset, k_a är objektets ambienta reflektion, L_a är det ambienta ljuset, K_d är objektets diffusa reflektion, L_d är det diffusa ljuset, $\hat{N}$ är den normaliserade normalen vid reflektionspunkten och $\hat{L}$ är den normaliserade riktningen mot ljuskällan.

Figur 2.5 visar förbättringen som sker när det diffusa ljuset adderas till scenen, vilket även är en förutsättning för att generera skuggor.

Figur 2.5: Till vänster: Enbart ambient ljus. Till höger: Ambient + diffust ljus.

2.3.3 Skuggor

Förutom att ha en realistisk miljö så är en viktig del i grafiken att uppfatta hur en boll rör sig i luften och då krävs det skuggor för att se var bollen befinner sig. Dessa skuggor skapas genom en process som kallas *shadow mapping*. I denna process så renderas scenen tillfälligt utifrån ljuskällans perspektiv och parallella strålar skickas ut med en konstant riktning. De objekt som strålarna når först blir belysta och objekten bakom dessa blir skuggade. Processen illusteras i Figur 2.6.



Figur 2.6: Shadow mapping där ytan bakom objektet blir skuggat.

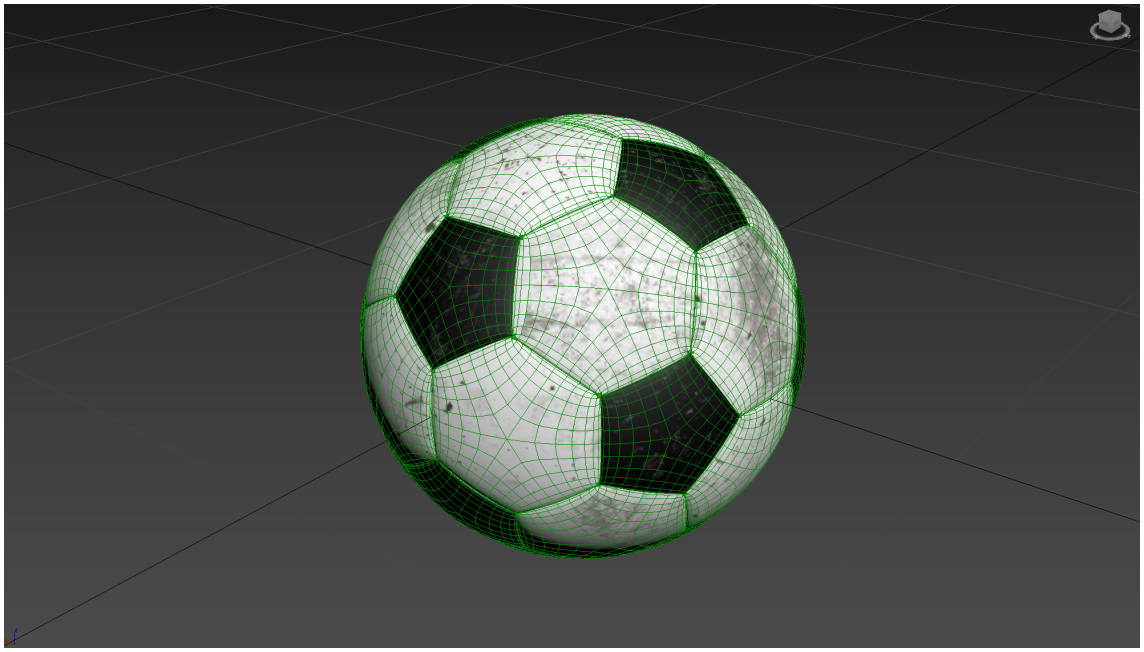
2.3.4 Skybox

En *skybox* är den miljö som alltid upplevs ligga längst bort i scenen, vilket ger scenen en rymd som är visuellt tilltalande. Trots att den upplevs ligga längst bort så ligger den egentligen alldeles runt om kameran, formad som en kub [4]. På kubens sex ytor placeras sex olika texturer som tillsammans bildar en avlägsen miljö. Genom att tillfälligt stänga av den del i renderingen som detekterar djup så visas skyboxen bakom alla andra objekt, trots att det egentligen ligger framför alla andra.

2.3.5 Modeller

För att ge simulatorn ett realistiskt utseende krävdes 3D-modeller som till exempel en boll, en fotbollsplan och ett mål. 3D-modeller är uppbyggda av trianglar med normaler riktade utåt så att texturer kan appliceras, samt reflektioner beräknas, på rätt sätt.

Istället för att skapa modeller i OpenGL, som är tidskrävande, importerades de som objekt- och materialfiler med hjälp av biblioteket *ASSIMP*. I objektfiler finns en beskrivning om modellens geometri och i materialfiler finns information om det visuella i modellen det vill säga texturer, transparens och materialegenskaper. Modellerna byggdes upp i modelleringsprogrammen *Autodesk 3ds Max* och *Sketchup*. Bollen och arenans modeller [7][8] importerades från utomstående 3D-grafiker på grund av tidsbrist. Dessa modeller modifierades sedan för att uppfylla gruppens krav. Bollens uppbyggnad kan ses i Figur 2.7.



Figur 2.7: Bollens modell i 3ds Max.

Kapitel 3

Resultat och diskussion

Resultat och diskussion om arbetet kommer tas upp i detta kapitel. Resultatet fokuserar främst på de 3 sektioner som tas upp i metoden; matematiska modeller, simulering och grafik. Diskussionen fokuserar främst på huruvida uppsatta mål är nådda och på vidareutveckling av applikationen.

3.1 Resultat

Förarbetet med de matematiska modellerna resulterade i verklighetstrogna simulationer i MATLAB. Bollens rotation påverkar bollens bana vinkelrätt mot rotationsriktningen och bollens hastighet. Luftmotståndet påverkar bollen beroende på dess hastighet. Gravitationen påverkar bollens bana längs y-axeln.

I applikationen som skapats simuleras en fotbolls bollbana. Det är möjligt för användaren att ändra utgångshastighet, utgångsvinklar, vinkelhastighet och rotationsriktning med hjälp av ett grafiskt gränssnitt. Kontrollerna för att använda simulatorn redovisas i Tabell 3.1. Utöver dessa parametrar är det enkelt att modifiera koden för att även ändra följande egenskaper:

- Luftdensitet
- Bollens radie
- Bollens massa
- Gravitationskonstant
- Koefficienter för luftmotstånd och Magnuseffekt
- Simuleringens hastighet

Applikationen är uppbyggd i en tredimensionell miljö med olika 3D-modeller (boll, fotbollsarena, mur, och mål), ljussättning, dynamiska skuggor och en skybox. Simulatorn är dessutom framtidskompatibel, vilket innebär att koden är förberedd för implementation av ytterligare krafter, exempelvis vind. Den framtagna simulatorn visas i Figur 3.1.

Figur 3.1: Utseendet av simulatorn.

3.2 Användande av applikation

För att förenkla användningen av applikationen har tangentbindningarna i Tabell 3.1 tagits fram. Fördelen med att använda tangentbindningarna är att parametrar kan ändras utan att applikationen behöver startas om.

Tabell 3.1: Tangentbindningar

F	Skjut iväg fotbollen
R	Återställ bollen till initialvärden
Q	Välj bollens startposition
Esc	Avsluta programmet
Vänster Ctrl	Navigera i meny
Vänster musklick	Placera ut bollen

Eftersom applikationen använder simpel grafik finns det inga hårdvaruspecifikationer som måste uppfyllas på datorn förutom att mus, tangentbord och skärm används. Datorn måste dock använda Windows som operativsystem.

3.3 Diskussion

I simulatorn är det även möjligt att simulera andra typer av bollprojektiler, exempelvis ett basketkast eller ett basebollsslag. För att göra detta måste vissa parametrar ändras, såsom bollens radie, massa, lyftkraftskoefficient och motståndskraftskoefficient. I en vidare utveckling av applikationen kan dessa parametrar vara beräknade för olika typer av bollar och vara valbart i menyn inne i applikationen.

Ett annat utvecklingsområde är att utöka antalet faktorer som räknas med i modellen. I ett verkligt scenario så finns det många faktorer som påverkar en bollbana. I detta projekt

prioriterades vissa faktorer vilket innebär att det finns faktorer som uteslöts, till exempel vind och bollens deformation. Detta skulle resultera i en bollbana som överensstämmer med en verklig bollbana ännu mer.

Ett problem som stöttes på under projektets gång var att simuleringarna som utfördes inte gick att jämföra med ”riktiga” experiment, vilket gjorde det svårt att verifiera att våra simuleringar stämde. Det som istället användes att jämföras med var en verklig frispark från 1997 av den professionella fotbollsspelaren Roberto Carlos. Denna frispark har studerats i många år och approximativa värden har framtagits såsom utgångs- och vinkelhastighet, riktning, lutning och avstånd från målet. Det enda värdet som man med säkerhet vet är korrekt är avståndet från målet, detta gör att simuleringen av frisparken inte kan bli identisk med Carlos frispark. Trots dessa osäkerheter så stämmer våra simuleringar bra överens med värdena från Carlos skott. Bollen når samma längd och ser ut att följa samma bollbana.

Detta projekt kan jämföras med *SoccerNASA* som är en liknande typ av simulator [5]. Simulatorn utvecklades och uppdaterades senast av NASA år 2010. Användaren kan, på samma sätt som i detta projekt, ändra värden på olika parametrar såsom utgångs- och vinkelhastighet, riktning och lutning. I NASAs simulator finns även andra parametrar som är justerbara, såsom temperatur, höjd över havet och luftfuktighet. Den största skillnaden på dessa två simulatorer är grafiken. Trots att bollen rör sig i en tredimensionell rymd i NASAs simulator så visualiseras det endast i två dimensioner vilket leder till att det blir svårare att uppfatta bollbanan.

Kapitel 4

Slutsatser

Projektet resulterade i en simulator som kan användas för att visualisera en fotbolls projektilbana, men kan även modifieras för att simulera andra typer av klot. Det visade sig vara effektivt att beskriva systemet i form av differentialekvationer eftersom Eulers stegmetod då kunde användas. Stegmetoden användes dels för att numeriskt lösa differentialekvationerna och ge diskreta värden. Den användes även för att bestämma hastigheten på simuleringen genom att ändra steglängden. De simuleringar som utfördes i MATLAB var ett viktigt steg för att verifiera de matematiska modellerna, vilket gjorde att projektet kunde fortgå med tillit till modellerna. Slutligen var den tredimensionella visualiseringen ett viktigt element för att ge användaren en god uppfattning av systemet.

Litteraturförteckning

[1] John Wesson, *The Science of Soccer, First Edition*, Institute of Physics Publishing, 2002

[2] Juliana Javorova och Anastas Ivanov, *STUDY OF SOCCER BALL FLIGHT TRAJECTORY*, MATEC Web of Conferences, 2018

[3] Daniel Stephen Price, *Advanced modelling of soccer balls*, Loughborough University Institutional Repository, 2005

[4] Ben Cook, *Computer Graphics with Modern OpenGL and C++*, udemy, hämtad: 2019-03-12
https://www.udemy.com/graphics-with-modern-opengl/

[5] NASA, *SoccerNASA Version 1.4c*, NASA, hämtad: 2019-03-14
https://www.grc.nasa.gov/www/k-12/airplane/soccercode.html

[6] NASA, *Drag on a Soccer Ball*, NASA, hämtad: 2019-03-14
https://www.grc.nasa.gov/www/k-12/airplane/socdrag.html

[7] gingerronaldinho, *Stadium*, 3dwarehouse, hämtad: 2019-03-01
https://3dwarehouse.sketchup.com/model/19698579ddc133a85ab8e23ee2d877e1/Stadium

[8] luxxeon, *Soccer Ball*, turbosquid, hämtad: 2019-02-22
https://www.turbosquid.com/3d-models/free-max-model-soccer-ball/750062

[9] Benjamin Elisha Sawe, *The most popular sports in the world*, thewordlatlas, hämtad: 2019-03-14
https://www.worldatlas.com/articles/what-are-the-most-popular-sports-in-the-world.html