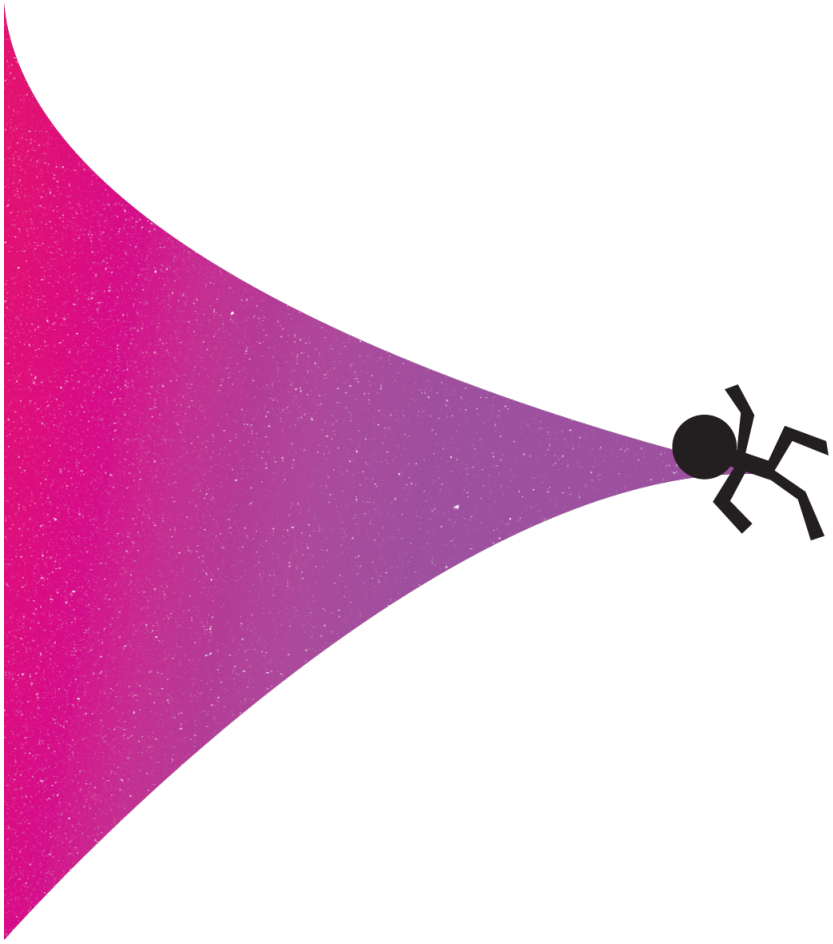


Skapandet av Moderately Accurate Human Space Golf Simulator

Uppsats för avläggande av kandidatexamen i Data- och Informationsteknik 15 hp

CHRISTOFFER BRASJÖ
NILS KENNE
JENNY LINDEWALL
ISAK PERSSON
KEVIN RASKU
AXEL SAVEN ÖSTEBO

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2017

Uppsats för avläggande av kandidatexamen i Data- och Informationsteknik 15 hp

Skapandet av Moderately Accurate Human Space Golf Simulator

En undersökning om fysikaliska fenomens inverkan på spel

CHRISTOFFER BRASJÖ
NILS KENNE
JENNY LINDEWALL
ISAK PERSSON
KEVIN RASKU
AXEL SAVEN ÖSTEBO

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet

Göteborg, Sverige 2017

Skapandet av Moderately Accurate Human Space Golf Simulator

En undersökning om fysikaliska fenomens inverkan på spel

CHRISTOFFER BRASJÖ
NILS KENNE
JENNY LINDEWALL
ISAK PERSSON
KEVIN RASKU
AXEL SAVEN ÖSTEBO

© Christoffer Brasjö, Nils Kenne, Jenny Lindewall, Isak Persson, Kevin Rasku, Axel Savén Östebo 2017.

Kandidatarbete CIUX03-17-07 2017
Inst. för Data- och Informationsteknik
Chalmers tekniska högskola
Göteborg Universitet
412 96 Göteborg, Sverige
Telefon: +46 (0)31-772 100

Handledare:	Staffan Björk,	Inst. för Data- och Informationsteknik
	Marco Fratarcangeli,	Inst. för Data- och Informationsteknik
Examinator:	Morten Fjeld	Inst. för Data- och Informationsteknik

Kontaktinformation

Christoffer Brasjö	brasjo@student.chalmers.se
Nils Kenne	kenne@student.chalmers.se
Jenny Lindewall	jennyli@student.chalmers.se
Isak Persson	pisak@student.chalmers.se
Kevin Rasku	rasku@student.chalmers.se
Axel Savén Östebo	osaxel@student.chalmers.se

Omslagsbild: En promotionbild till spelet Moderately Accurate Space Golf Simulator

Institutionen för Data- och Informationsteknik
Göteborg, Sverige 2017

Sammandrag

Fysikaliska fenomen används flitigt i spel för att skapa realism och höja underhållningsvärdet. Det är dock inte lika vanligt att de spelar en central roll i spelens *gameplay*. I detta projektarbete har en spelprototyp skapats samt en undersökning av fysikaliska fenomen i redan existerande spel genomförts. Detta har gjorts med syftet att undersöka implementations- och designmöjligheter för fysikaliska fenomen i spel samt att skapa en dokumentation av fysikaliska fenomen i spel.

För att uppnå projektets syfte har spelprototypen *Moderately Accurate Human Space Golf Simulator* utvecklats och fysikaliska fenomen inom relativitetsfysik, optik, mekanik, magnetism och hållfasthetslära undersökts i de spel presenterade i rapporten.

Spelprototypens *gameplay* går ut på att navigera en *avatar* i tyngdlöst tillstånd genom olika nivåer, där de olika fysikaliska fenomenen introduceras allt eftersom spelaren avancerar i spelet. Samtliga försök att implementera fenomenen har lyckats men i vissa fall har problem uppstått vid design av nivåer och därmed påverkat *gameplay* negativt.

Resultatet av spelundersökningen är en dokumentation av fysikaliska fenomenen kopplade till de spel som undersökts samt hur de fysikaliska fenomenen har implementerats.

Abstract

Physics phenomena is widely used in games to create a sense of realism and to enhance the entertainment value. However, it is not as common for physical phenomena to play a central role in gameplay. The work in this project has consisted of creating a game prototype and conducting an investigation into the physical phenomena used in already existing games. This has been done with the purpose to investigate the possibilities regarding implementation and design as well as to create a documentation of physics phenomena in games.

To fulfill the purpose of this project, the game prototype *Moderately Accurate Human Space Golf Simulator* has been developed and physics phenomena in relativity physics, optics, mechanics, magnetism and mechanics of materials have been researched in the games presented in this thesis.

The basic gameplay of the prototype centers around navigating an avatar in zero gravity through different levels. The different physics phenomena are introduced as the player advances in the game. The implementations have been successful but problems have occurred regarding the level design and consequently affected the gameplay negatively.

The result of the game investigation consists of a documentation of physics phenomena appearing in the investigated games and information on how the phenomena have been implemented.

Förord

Genom projektets gång har flera utomstående personer hjälpt till och bidragit på olika sätt. Hjälpen har varit allt från spel- och implementationsidéer till mentalt stöd och vägledning, då arbetsbelastningen var hög eller då implementationerna inte fungerade. Projektgruppen vill därför passa på att tacka några utav dessa personer som både gjort detta projekt möjligt och samtidigt gjort det till en rolig upplevelse för oss.

En utav dessa personer är vår handledare Staffan Björk som har ställt upp i vått och torrt, flängt mellan två olika campus för vår skull och agerat räddande medlare vid interna dispyter. Vi har inte alltid hållit med varann om hur saker och ting skall utföras men vi har verkligen tagit hans bästa lärdomar till oss.

Vi vill tacka Vadim Ledyaev, utvecklare av spelet Optika, för hans entusiastiska och hjälpsamma inställning. Under en lång Skypediskussion förklarade han allt vi ville veta om hur han utvecklade sitt spel och implementerade fysiken i det. Tack till Kyle Gabler, utvecklaren av World of Goo, för sina snabba mejlresponser och utlägg om implementation kring hållfasthet.

Ett stort tack till Gustav Magnusson, som kom med originalidén till spelet, och till Patrik Björnsson som skapade den underbara musiken i spelet. Utan er hade spelet sett helt annorlunda ut.

Vi vill även tacka dem som hjälpt till i förbifarten, vid mötet i korridoren eller som visat vägen till handledningssalen. Ni är minst lika viktiga!

Och slutligen ett stort tack till alla er som varit med och bollat idéer kring olika fysikaliska fenomen. Ni vet vilka ni är.

Christoffer, Nils, Jenny, Isak, Kevin och Axel
Göteborg, Maj 2017

Ordlista

Avatar	En virtuell figur som representerar och styrs av spelaren.
FixedUpdate()	En funktion som kallas med frekvensen 50Hz och används endast till att uppdatera fysiken i ett spel.
Frame	En enskild bildruta i en rörlig bild.
Gameplay	En sammanfattning av spelarens interaktion med spelmekaniken och den totala spelupplevelsen.
Nivå	Bana sorterad efter introduktion av fenomen
OrbitalCorrection()	En metod i spelprototypen som kan forcera spelaren in i en cirkulär omloppsbana vid rätt förutsättningar.
Ragdoll	En streckfigur med rörliga armar och ben, vilka påverkas av hur streckfiguren rör sig.
Rigidbody	Kontroll av ett objekts position genom fysiksimulation.
Spelaren	Personen som spelar spelet.
Tidskontroller	Saktar ner spelet vid drag för ivägskjutning och ger spelaren mer tid att sikta.
Tutorial	Instruktionsvideo eller instruktionsmanual med steg för steg genomgångar för enklare inlämning.

Innehåll

1 Introduktion **1**
- 1.1 Syfte . . . 2
- 1.2 Uppgiftsbeskrivning . . . 2
- 1.3 Avgränsningar . . . 3

2 Teori **5**
- 2.1 Fjädrar och Hookes lag . . . 5
- 2.2 Gravitation . . . 6
  - 2.2.1 Omloppsbana . . . 6
- 2.3 Ljuseffekter och Relativitet . . . 7
  - 2.3.1 Relativitet . . . 7
- 2.4 Elektriska och Magnetiska fält . . . 9

3 Genomförande **11**
- 3.1 Dokumentationen av fysikaliska fenomen i spel . . . 12
- 3.2 Spelidé . . . 12
- 3.3 Implementation . . . 13
  - 3.3.1 Implementation av grundläggande spelmekanik . . . 13
  - 3.3.2 Gravitation . . . 15
  - 3.3.3 Omloppsbana . . . 16
  - 3.3.4 Elektriska och magnetiska fält . . . 17
  - 3.3.5 Relativitet . . . 18
  - 3.3.6 Strålning . . . 19
  - 3.3.7 Nivådesign . . . 20

4 Resultat **23**
- 4.1 Resultat för spelprototypen . . . 23
- 4.2 Dokumentationen om Fysikaliska fenomen i spel . . . 26
  - 4.2.1 Ljuseffekter och Relativitet . . . 28
  - 4.2.2 Hållfasthetslära . . . 29
  - 4.2.3 Gravitation . . . 31
  - 4.2.4 Strömningsmekanik . . . 31

5 Diskussion **35**
- 5.1 Diskussion om avgränsningar & genomförande . . . 35
- 5.2 Implementationsdiskussion . . . 37
  - 5.2.1 Grundläggande spelmekanik . . . 37

5.2.2 Gravitation och banrörelse 38
5.2.3 Magnetism och kraftfält 39
5.2.4 Relativitet och ljusfenomen 39
5.2.5 Nivådesign 40
5.3 Diskussion kring undersökningen och dokumentationen 41
5.4 Framtida arbete & utvecklingsmöjligheter 42

6 Slutsatser **43**

Referenser **43**

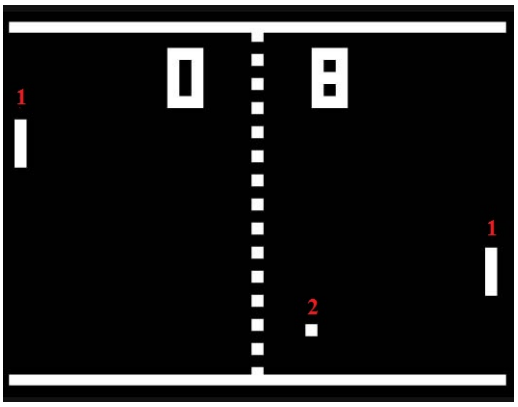
A Appendix **1**
A.1 Utritande av banapproximationen vid ivägskjutning 1
A.2 Visualisering av fältriktning 1

1

Introduktion

Datorspelen har utvecklats kraftigt sedan första versionen av spelet *Pong* släpptes 1972, vilket är ett utav de första kommersiella datorspelen någonsin [1]. I figur 1.1 visas hela spelplanen där de två rektanglarna kan röra sig i vertikal led för att stoppa bollen, vilken rör sig dem emellan. Bollen byter riktning då den träffar antingen en målvakt eller ramen och syftet är att hålla bollen i spel, utan att den passerar någon av kortsidorna. Spelets originalitet och simpla design startade ett starkt intresse världen över för datorspel och datorspelsdesign. Sedan dess har antalet spelare ökat till över 1,7 miljarder världen över år 2014 [3].

Speldesignen idag är mer komplex eftersom fler spelfunktioner och algoritmer tillkommer vid design av mer realistiska spel. För att skapa realistiska rörelser, miljöer och hur olika objekt interagerar med varandra, kan information hämtas från omvärlden. Genom att utnyttja de ekvationer som styr olika verkliga fysikaliska fenomen och implementera dessa ekvationer i spelen, finns ingen begränsning för hur verkliga spel kan komma att uppfattas i framtiden.



Figur 1.1: Första Pong-spelet från 1970. Två rektanglar (1) som kan röras i vertikal led agerar målvakter för att stoppa bollen (2). Från [2]. CC-BY-SA

Idag finns nästan oräkneligt många spel eller applikationer som använder sig av fysik och för att inte behöva skriva dessa ekvationer i varje enskilt fall används en, eller flera, så kallad spelmotorer [4]. Om syftet är att skapa ett vapen i ett spel definieras spelmotorns roll som de funktioner och algoritmer vilka styr hur vapnet ska interagera med omgivningen. Fysikaliska effekter såsom gravitation, tröghet, elastiska respektive inelastisk stötar eller andra effekter som påverkar vapnet från omgivningen i spelet, styrs alltså av en spelmotor. Genom att nyttja en spelmotor förenklas arbetet som krävs för att skapa ett spel och är därför att föredra vid en första bekantskap med datorspelsutveckling.

En positiv följd då verklig fysik implementeras i spel är att spelare även kan, genom dessa spel, lära sig fysik [5]. Även om spelet i sig inte lär ut själva ekvationerna som styr fysiken, kan spelare ändå få en förbättrad intuitiv förståelse för hur fysiken fungerar. Exempel på sådana spel är *World of Goo* där broar byggs med fysik såsom momentjämvikt, *Angry Birds* där fåglar skjuts och kolliderar med hus som raseras,

samt *Where is My Water* där en krokodil ska duscha, med vatten som styrs av strömningsmekanik. Detta talar för att mer resurser bör läggas på att utforska möjligheterna att implementera fler fysikaliska fenomen i spel och finna en bra kombination mellan realism och spelglädje.

Många fysikaliska fenomen finns dessutom redan implementerade i olika spel men i dagsläget är det komplicerat att hitta information om vilka fenomen och i vilket spel de är implementerade. Därför är det av intresse att studera och testa olika spel för att på så vis ta reda på vilka fenomen som är implementerade. Med en dokumentation blir det enklare att utforska och implementera ett fysikalisk fenomen som tidigare aldrig implementerats i spel.

1.1 Syfte

Det sammanfattade syftet med projektet är att undersöka implementerade fysikaliska fenomen i datorspel samt implementationsmöjligheter för fysikaliska fenomen i datorspel. Syftet delas därför in i två huvudsyften.

Det första huvudsyftet med projektet är att undersöka möjligheten att implementera fysikaliska fenomen i datorsimuleringar. Detta genom att studera hur fysiken och matematiken som beskriver fenomenet fungerar i verkligheten, för att sedan hitta sätt att implementera dessa i en fungerande **spelprototyp**.

Det andra huvudsyftet med projektet är en undersökning av olika fysikaliska fenomen i existerande datorspel. **Undersökningen** ska leda till en dokumentation som relaterar det fysikaliska fenomenet till på vilket sätt, och i vilket spel, det är implementerat.

1.2 Uppgiftsbeskrivning

Spelprototypens syfte är att testa hur väl implementationen går ihop med spelets *gameplay*, vilket defineras både som spelets mekanik (hur det spelas) och spelarens upplevelse av mekaniken. Grafiken och ljudet hör alltså inte till *gameplay*, vilket i andra fall utanför projektet, även kan inkludera till exempel handlingen i spelet. Detta medför att fenomenet självt eller implementationen av fenomenet, kan modifieras för att förbättra spelets totala *gameplay*.

Detta görs för att undersöka möjligheten att designa spel kring fysikaliska fenomen. Speldesigners kan sedan använda sig av dessa implementationer i egna spel, vilket leder till att spelare enklare kan komma i kontakt med fler olika fysikaliska fenomen i spel.

Varför fenomenet har implementerats på just sitt specifika sätt i aktuellt spel, med hänsyn till approximationer, förenklingar, och *gameplay*, skall också framgå i dokumentationen. Undersökningen fungerar även som en grund för det första huvudsyftet.

Anledningen är att göra det enklare för spelare att, genom denna rapport, komma i kontakt med fler fysikaliska fenomen i datorspel.

1.3 Avgränsningar

Med syftet att både undersöka och skapa fysikaliska fenomen i spel, togs beslut om avgränsningar i det första skedet av projektet. Detta då de båda ämnena är breda och det fanns många möjliga vägar att gå. Vi fokuserade på att avgränsa följande punkter.

- Hur vi skulle utveckla spelet.
- Hur vi skulle undersöka fysikaliska fenomen i spel.
- Vilken eller vilka plattformar spelet ska fungera på.

Utveckling. För att inte behöva utveckla en spelmotor, med tillhörande fysikmotor, från grunden valdes spelmotorn *Unity [6]* som primär utgångspunkt för spelutvecklingen. Detta eftersom *Unity* dels är gratis och har använts i hela 34 procent av de 1000 bästa gratisspelen till mobil 2016, men framförallt för dess välbyggda utvecklingsdokumentation och inlärnings-*tutorials* [7][8].

Dessutom avgränsades spelutvecklingen till tvådimensionella (2D) spel, då färre dimensioner ger enklare lösningar till fysikaliska ekvationer, vilka presenteras i teoriavsnitt 2, och därigenom förutspåddes att fler fenomen kunde implementeras under projektets gång.

En avgränsning gällande grafiken i spelet valdes eftersom syftet var att implementera fysikaliska fenomen, utan krav på utseende. Istället fyller grafiken syftet att förtydliga fenomenet genom enklare geometriska figurer, i motsatts till estetiska och grafiskt tilltalande effekter för den visuella upplevelsens skull.

Undersökning. Då det finns många speltuvecklingsföretag världen över, där det bara i Sverige fanns över 213 stycken 2014 med en växande trend, behövdes ett sätt att sortera dem på [9]. Framför allt behövdes ett sätt att hitta de spel, skapade av spelutvecklarna, som innehöll fysikaliska fenomen av intresse för projektet. Därför valdes *Steams* [10] samlade spelbank med över 14.755 spel 2017, varav 1.259 med söktermen ”physics” [11]. Utifrån dessa spel valdes olika fenomen och deras implementationer för att möta det andra huvudsyftet.

Plattformar. *Unity* är kompatibelt med flera olika plattformar såsom *Windows, Mac IOS, Android* och andra större spelkonsoler såsom *PlayStation 4* och *Xbox One [12]*. Detta gav möjligheten att göra spelet kompatibelt till flera plattformar. Men inledningsvis avgränsades plattformen till *Windows PC*, eftersom utvecklingsarbetet bedrevs där, men lyckade försöksversioner gjordes även till *Android*.

2

Teori

Den kända vetenskapsmannen Galileo Galilei ansåg att ”Naturens bok är skriven i matematikens språk”, om sambandet mellan fysik och matematik [13]. De går båda hand i hand där det ena inte kan vara utan det andra. Genom att studera naturfenomen kan matematik utvecklas och genom att studera matematik kan fysik utvecklas.

Lyckligtvis förenklar detta livet för en speldesigner, då matematiska formler kan skapa simuleringar i datorspel. Om ett fysikaliskt fenomen kan beskrivas matematiskt kan det också, i de flesta fall, implementeras som en simulering. Detta kan senare leda till ett spel.

I detta avsnitt finns därför en mer ingående förklaring till de fenomen som har implementerats samt några utav dem som undersökts under projektets gång.

2.1 Fjädrar och Hookes lag

Inom hållfasthetslära approximeras deformationer i elastiska material som fjädrar [14]. Enligt Hookes lag

$$\vec{F} = -kx\hat{x}, \qquad |\vec{F}| = F = |-kx| \tag{2.1}$$

där k är en fjäderkonstant och x är en förskjutning från jämviktsläget, kommer spänningen σ verkandes på en snittyta A på materialet vara lika med F/A. Med spänningen menas den tryck- eller dragtryckkraft som verkar på till exempel en balk i dess längdriktning. Detta ger upphov till en töjning $\epsilon = l/L$, där L är ursprungliga längden och l är förlängningen, alternativt förkortningen. Spänningen och töjningen följer ett linjärt samband enligt

$$\sigma = \epsilon E \tag{2.2}$$

där E är en konstant kallad elasticitetsmodulen, vilken beror på materialet [15]. Det linjära sambandet gäller dock enbart till en viss gräns, kallad sträckgränsen σ_s, där $\sigma > \sigma_s$ skapar permanenta deformationer, vilket kan leda till brott. Men om $\sigma < \sigma_s$ och sedan avlägsnas, kommer balken återgå till sin ursprungslängd, tack vare dess elasticitet.

2.2 Gravitation

En utav de fyra naturkrafterna, gravitationen, är en attraktiv kraft som verkar mellan all materia enligt ekvationen

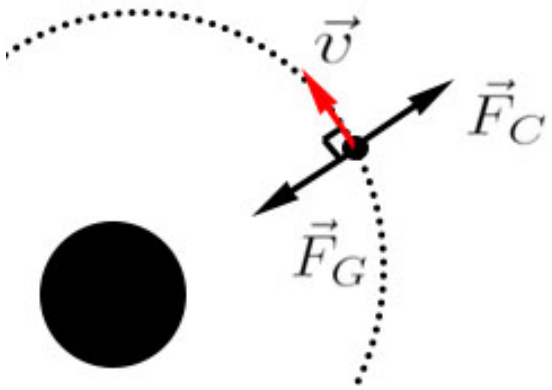
$$\vec{F} = \frac{Gm_1m_2}{r^2}\vec{e}_r \tag{2.3}$$

där G är gravitationskonstanten, m_1 och m_2 är respektive massa som kraften verkar på samt r är avståndet dem emellan [15]. Det är denna kraft som håller oss kvar på Jorden och som håller ihop vårt solsystem. Utan gravitation säger ekvation (2.3) att inga planeter, solar eller galaxer kan existera.

2.2.1 Omloppsbana

Till följd av gravitationen uppkommer fenomenet omloppsbana vilket är den elliptiska, eller specialfallet cirkulära, banan ett objekt rör sig kring en himlakropp [16]. Omloppsbanan uppkommer genom en kombination av gravitationen och att objektet har en hastighet relativt himlakroppen. Gravitationen blir därför en centripetalkraft, vilken är den kraft som verkar på ett objekt som rör sig i en cirkulär bana. Centripetalkraften är till exempel den kraft som en person måste hålla i repet till ett klot under sin roterande rörelse vid utövande av släggkastning. När personen släpper repet följer kulan en rak linje i samma riktning den hade precis innan repet släpptes.

Omloppsbanan uppkommer då objektets hastigetsvektor $\vec{v}$ relativt himlakroppen har en nästan 90° vinkel mot ortsvektorn $\vec{r}$ mellan himlakroppen och objektet. Detta gör att gravitationskraften snarare vrider objektets hastighetsvektor än att öka dess fart. Det är förhållandet mellan $\vec{v}$ och $\vec{r}$ som är avgörande om det blir en omloppsbana eller inte. Vid fel förhållande dem emellan resulterar i att objektet antingen fortsätter ut i rymden eller kraschar med himlakroppen.



Figur 2.1: En cirkulär omloppsbana uppkommer då gravitationskraften $\vec{F}_G$ är lika stor som den av objektet upplevda centrifugalkraften $\vec{F}_C$, vilken är beroende av $\vec{v}$.

Vid specialfallet av en cirkulär bana är $\vec{v}$ helt vinkelrät mot $\vec{r}$ och farten v som funktion av radien r kan beräknas genom att kombinera gravitationsekvationen (2.3) och Newtons kraftekvation för en centripetalkraft

$$\vec{F}_C = m\vec{a}_c \tag{2.4}$$

där m är objektets massa och $\vec{a}_c$ dess centripetalacceleration. Eftersom radien är konstant vid cirkulär rörelse ger jämviktsekvationen att $F_G = F_C$, alltså $m_1 a_c = Gm_1 m_2/r^2$, vilket kan ses i figur 2.1. Genom insättning av förhållandet mellan accelerationen och farten, $a_c = v^2/r$ [15], i jämviktsekvationen fås förhållandet mellan radien och farten som

$$r = \frac{Gm_2}{v^2}. \tag{2.5}$$

Då dessa förhållanden är uppfyllda fås en perfekt cirkulär rörelse.

2.3 Ljuseffekter och Relativitet

Ljus kan beskrivas både som en ström av fotoner men även som en elektromagnetisk våg [17]. Då dessa fotoner träffar våra ögon skapas elektriska signaler, vilka tolkas i hjärnan som olika färger. Varje foton har en energi

$$E = hf \tag{2.6}$$

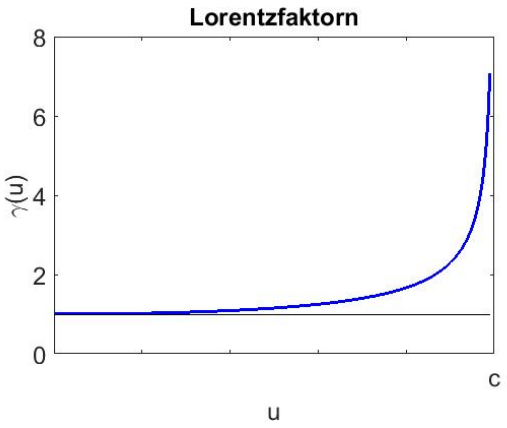
där h är plancks konstant och f är en frekvens [15]. Till denna frekvens hör en våglängd λ, vilken är avståndet mellan två vågtoppar och bestämmer dess färg. Denna elektromagnetiska våg propagerar genom rymden med ljusets hastighet $c = 299,792$ km/s och de tre förhåller sig till varandra enligt

$$\lambda = \frac{c}{f}.\text{[15]} \tag{2.7}$$

En våglängd som ligger i intervallet 350 till 700 nm ger synligt ljus där färgerna går från blått till rött med alla regnbågens färger där emellan [17]. Då ögat belyses med ljus innehållandes alla våglängder för synligt ljus upplevs det som vitt medan avsaknaden av ljus upplevs som svart.

2.3.1 Relativitet

Ljusets hastighet c i vakuum är konstant oavsett ur vems perspektiv det betraktas [18]. Till exempel färdas ljuset från en asteroid, med farten v riktad mot jorden, lika snabbt som det inkommande ljuset i teleskopet på jorden. Einsteins relativitetsteori säger oss att inget kan färdas snabbare än ljuset och eftersom ljuset hastighet är konstant ur alla observatörers perspektiv, måste istället rum och tid förändras för att Newtons lagar ska vara sanna ur samtliga perspektiv. Detta ger upphov till relativistiska effekter såsom *dopplereffekten*, *searchlight effekten*, *längdkontraktion* och *tidsdilation*. De flesta relativistiska effekter uppkommer först då objekt rör sig relativt varandra med en fart u som närmar sig ljushastigheten.



Figur 2.2: Lorentzfaktorn $\gamma(u)$ används vid relativistiska beräkningar där u är den relativa hastigheten och c är ljushastigheten.

Det är intressant att analysera *Lorentzfaktorn* γ vilken uppkommer i de flesta relativistiska beräkningar och defineras som

$$\gamma = \gamma(u) = \frac{1}{\sqrt{1 - u^2/c^2}}.[18] \tag{2.8}$$

I figur 2.2 finns en graf över Lorentzfaktorn vilken är mer eller mindre konstant lika med ett tills u är ungefär två femtedelar av c. Detta är anledningen till att de flesta relativistiska effekter uppkommer vid hastigheter nära c.

Dopplereffekten är en följd av relativiteten och kan märkas i vardagen då en ambulans vid uttryckning närmar sig en bil bakifrån på motorvägen. Chauffören i bilen upplever ljudet, eller ljudvågorna, från ambulansen som starkare och med en högre ton än då ambulansen passerar bilen och kör ifrån den. Det är frekvensen f på ljudvågorna som bestämmer tonen som upplevs i örat, där en högre frekvens ger en högre ton och tvärt om. Frekvensen hos ljusvågor motsvarar istället en viss färg. I detta fall, vid antagandet av endimensionell rörelse och att ambulansen har farten u relativt bilen, beräknas den upplevda frekvensen enligt

$$f = \sqrt{\frac{1 + u/c}{1 - u/c}} f_0.[18] \tag{2.9}$$

där f_0 är den frekvensen med vilket ljudet sänds ut. Samma formel kan appliceras på ljusvågor.

Searchlight-effekten uppkommer från ljusavvikelser då ljus sänds ut från till exempel en farkost i rörelse relativt en observatör på jorden där ljuset inkommer med en vinkel α. Då farkostens fart närmar sig ljushastigheten och har riktning jorden upplevs effekten att ljuset sänds ut som en kon, vilken blir smalare och smalare ju närmare ljushastigheten farkosten färdas. Detta innebär att ljusintensiteten ökar då konen, innehållandes hela intensiteten, fokuseras mot observatören. Därför upplevs ljuset från en strålkastare på en mötande långtradare som starkare än om den skulle stått stilla.

Längdkontraktion och tidsdilation förklarar hur objekt kan upplevas kortare eller längre vid relativistiska hastigeter respektive hur tiden upplevs olika i olika referenssystem. Till exempel kommer en stav med längd L_0 i ett referenssystem **S'** som rör sig relativt ett annat referenssystem **S** med farten u upplevas längre enligt

$$L = \frac{L_0}{\gamma(u)}.[18] \tag{2.10}$$

där L är den upplevda längden i **S**. Med referenssystem menas de perspektiv hos de två olika observatörerna som rör sig radiellt relativt varandra med fart u.

Tidsdilationen ger ekvationen

$$T = \gamma T_0 \tag{2.11}$$

där T_0 är tiden som mäts hos rymdfararen och T är tiden som mäts på jorden, vid antagandet av att båda klockor är synkroniserade och startas samtidigt. Båda effekterna uppkommer från symmetri, vilken är bevarad. Det vill säga att om observatören i **S** upplever att klockan i **S'** går långsamt upplever samtidigt observatören i **S'** att klockan går långsamt i **S** [18].

2.4 Elektriska och Magnetiska fält

Elektriska och magnetiska fält påverkar laddningar q och är sammankopplade enligt Maxwells ekvationer [17]. Alla laddningar, såsom protoner och elektroner, sänder ut ett elektriskt fält som propagerar med ljusets hastighet c i vakuum enligt

$$\vec{E} = E\vec{e}_r = \frac{q}{4\pi\epsilon_0 r^2}\vec{e}_r \tag{2.12}$$

där ϵ_0 är dielektricitetskonstanten, r är avståndet och $\vec{e}_r$ är den radiella enhetsvektorn, från källan. För protonen, med positivt laddning q, ger detta ett fält med riktning utåt medan för elektronen, med negativt laddning, ger ett fält riktat inåt. Då flera laddningar kombineras fås andra uttryck på E vilka löses med Maxwells ekvationer.

För en laddning q som rör sig i ett sådant fält, skapat av andra laddningar, upplever en kraft

$$\vec{F}_E = q\vec{E} \tag{2.13}$$

i samma riktning som fältet.

Magnetiska fält $\vec{B}$ med styrkan $|\vec{B}| = B$ uppkommer från laddningar i rörelse och endast laddningar i rörelse påverkas av magnetfältet [17]. Kraften

$$\vec{F}_B = q(\vec{v} \times \vec{B})[15] \tag{2.14}$$

verkandes på laddningen q är, enligt ekvationen, alltid vinkelrät mot både hastigheten och fältet. Detta innebär att magnetfältet enbart kan ändra riktningen på laddningens hastighet, inte dess fart. Om $\vec{v}$ enbart rör sig i ett plan med ett $\vec{B}$ fält i samma riktning som normalen till planet, ger insättning av ekvation (2.14) i (2.4) med $a_c = v^2/r$:

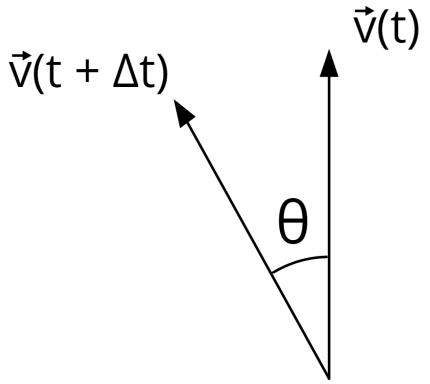
$$r = \frac{mv}{qB}. \tag{2.15}$$

Detta kan skrivas som

$$\omega = \frac{v}{r} = \frac{qB}{m} \tag{2.16}$$

där $\omega = v/r$ är vinkelfrekvensen och leder alltså till en cirkulär rörelse med radien $r = r(v, B, q, m)$.

Detta innebär att mellan ett visst tidsintervall Δt måste hastighetsvektorn $\vec{v}(t)$ roteras en vinken θ för att ge hastighetsvektorn $\vec{v}(t + \Delta t)$, enligt figur 2.3. Vinkeln



Figur 2.3: Vid en cirkulär bana måste hastighetesvektorn $\vec{v}(t)$ roteras med en vinkel θ för att efter en tid Δt ge hastighetetsvektorn $\vec{v}(t + \Delta t)$.

θ bestäms genom skalärprodukten

$$\frac{\vec{v}(t) \cdot \vec{v}(t + \Delta t)}{|\vec{v}(t)||\vec{v}(t + \Delta t)|} = \cos(\theta). \tag{2.17}$$

Med hastighetsvektorn

$$\vec{v}(t) = r(-\sin(\omega t), \cos(\omega t)) \tag{2.18}$$

i polära koordinater för en cirkulär rörelse, ger (2.18) i (2.17) att

$$\cos(\omega \Delta t) = \cos(\theta) \implies \omega \Delta t = \theta \tag{2.19}$$

eftersom $\omega \Delta t > 0$. Genom att multiplicera höger och vänster led i ekvation (2.16) med Δt och substituera $\omega \Delta t = \theta$ fås resultatet

$$\theta = \frac{qB}{m} \Delta t. \tag{2.20}$$

3

Genomförande

I detta kapitel behandlas de olika momenten som har utförts under projektets gång. Dessa inkluderar dokumentationen och undersökningen av spel och deras fysikaliska fenomen, den grundläggande spelidén samt en beskrivning av de implementerade fysikaliska fenomenen i spelprototypen.

För att undersöka fysikbaserad *gameplay* utvecklades en spelprototyp där fysikaliska fenomen implementerades och existerande spel som innehåller fysikbaserad *gameplay* undersöktes. Spelprototypen är ett tvådimensionellt spel och det utvecklades i spelmotorn *Unity [6]* som tillhandahöll en grundläggande utvecklingsmiljö. *Unity* har stöd för utveckling av tvådimensionella spel med möjlighet att implementera all spelmekanik som vårt projekt krävde vilket gjorde det till en passande spelmotor för vårt projekt.

Eftersom spelprototypen är ett tvådimensionellt spel innebär det att de fysikaliska fenomenen är anpassade till två dimensioner. För att kunna designa runt den fysikbaserade spelmekaniken, både när det gäller den grundläggande speldesignen och bandesignen, är spelet uppdelat i olika nivåer. Detta gjorde att vi kunde ha ett visst fysikaliskt fenomen på en nivå för att designa runt det och sedan ersätta det eller kombinera det med ett annat fenomen i en annan nivå.

Första versionen av spelprototypen baserades på en grundläggande fysiksimulering som huvudsakligen implementerade gravitation och rörelse för att sedan utökas till fler fysikaliska fenomen i takt med att spelprototypen växte och fler nivåer tillkom. Detta innebar att vi inte behövde vara bundna av att lösa ett enda stort problem utan istället kunde implementera de fysikaliska fenomenen var för sig. Detta för att sedan ha större frihet i att utforska andra fysikaliska fenomen i andra spelnivåer.

Genom att variera speldesignen i vår spelprototyp har vi kunnat utforska fysikaliska fenomen såsom gravitation mellan planeter och spelarens *avatar*, magnetiska och elektriska fält samt strålning. Ett exempel på ett spel som använder fysik på liknande sätt som den grundläggande spelmekaniken i vår spelprototyp är *Angry Birds [19]*, där konstruktioner av block kan raseras genom att skjuta projektiler med en slangbella.

Undersökningen av existerande spel med fysikbaserad *gameplay* gav oss en möjlighet att utforska fysikaliska fenomen som redan blivit implementerade. Genomförandet av den gav en inblick i vilka fenomen som har implementerats och hur implementationen fungerar.

3.1 Dokumentationen av fysikaliska fenomen i spel

För att uppnå andra huvudsyftet, med att skapa en dokumentation av fysikaliska fenomen i spel, har följande spel undersökts; *A Slower Speed of Light* [20], *Optika* [21], *World of Goo* [22], *Poly Bridge* [23], *Angry Birds Space* [24] och *Kerbal Space Program* [25].

Undersökningen av spelen har genomförts i flera steg. Det första steget syftade till att hitta fysikaliska fenomen i olika spel, vilket utfördes på två sätt. Det första sättet genom att söka direkt i *Steams* speldatabas med taggar som ”physics” och på det andra sättet genom att söka på internet efter ”fysikaliska fenomen”. Då hittades listor med olika fenomen och effekter vars namn sedan också kunde användas till att söka i *Steams* speldatabas.

I det andra steget låg fokus på att undersöka de sökträffar i *Steams* speldatabas som uppkom vid steg ett. Detta genom att spela vissa utav spelen, i mån av tillgång, titta på filmer med *gameplay* från spelen samt söka information om spelen från respektive utvecklare. Därefter valdes de fenomen som antingen hade bäst dokumentation och tydlig implementation, eller de som enligt projektgruppen hade mest intressant *gameplay* kring fenomenet, för att kunna inkludera dessa i dokumentationen.

I det sista steget kontaktades spelutvecklare för respektive spel som undersökts, för att på så vis få en mer detaljerad beskrivning av implementationen. Detta då informationen från spelutvecklarnas dokumentationer och hemsidor, i vissa fall, saknade information om implementationen av de fysikaliska fenomenen. Här var syftet att få svar på frågorna om utvecklarna själva studerat teorin bakom fenomenet innan det implementerades och varför de valt de approximationer och förenklingar som gjorts vid implementationen.

Resultatet av undersökningen är dokumentationen som presenteras i resultatavsnitt 4.2.

3.2 Spelidé

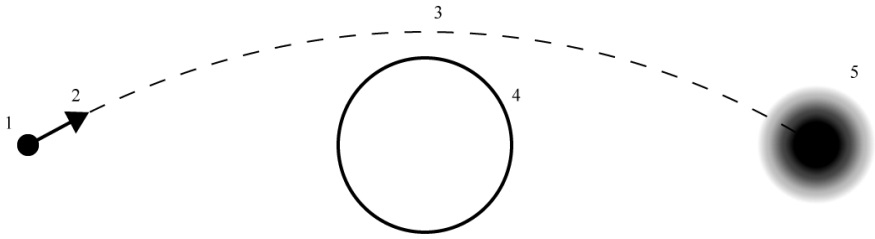
Grundidén till en egen spelprototyp, med arbetstiteln *Space Golf*, föddes utifrån efterfrågan av att kunna testa olika fysikaliska fenomen, i en och samma spelprototyp. Tanken var att skapa en grundmekanik med många utvecklingsmöjligheter. I ett tidigt skede av projektet bestämdes att grundmekaniken både skulle göra det möjligt att replikera redan implementerade fenomen i existerande spel, såsom gravitation, och samtidigt ge plats åt implementering av nya fenomen.

Därför valdes rymden som primära utgångspunkt där spelprototypens mål är att skjuta spelarens avatar genom rymden, undvika objekt såsom planeter och stjärnor, för att slutligen nå ett svart hål. Sedan skulle respektive fysikaliskt fenomen implementeras för att fungera ihop med grundmekaniken. Utifrån detta upplägg skulle varje fysikaliskt fenomen implementeras successivt för spelaren, i så kallade nivåer, där varje nivå är en kombination av ett eller flera fenomen.

Idén är främst inspirerad av spelet *Angry Birds Space* [24] som är liknande på

så sätt att det skjuts iväg projektiler som får sin bana påverkad av planeters gravitationsfält. Den största skillnaden mellan projektets spelidé och *Angry Birds Space* är målet i spelen. I projektets spelidé ska spelaren styra sin avatar till ett visst mål medan i *Angry Birds Space* är målet att skjuta sönder objekt. En annan större skillnad är att i projektets spelidé kan spelaren göra flera ivägskjutningar i realtid, i motsats till *Angry Birds Space* som är turbaserat.

Paralleller kan också dras till spelet *Kerbal Space Program* [25] vilket går ut på att konstruera och skjuta upp raketer i omloppsbana eller ut i rymden. Skillnadena här är dock stora då *Kerbal Space Program* är ett mycket avancerat spel och behandlar många fysikaliska fenomen samtidigt. Projektets spelprototyp skulle kunna liknas vid en väldigt förenklad 2D version av själva rymdmomentet i *Kerbal Space Program*, med större fokus på att utforska designmöjligheter än att vara fysikaliskt korrekt.



Figur 3.1: En skiss över den grundläggande spelmekaniken där 1 är avataren, 2 är kraftvektorn på avataren som spelaren bestämmer, 3 den bana som avataren färdas i efter att ha påverkats av kraftvektorn, 4 en planet och 5 det svarta hålet, vilket är målet med nivån. Planeten påverkar avatarens bana genom sitt gravitationsfält.

3.3 Implementation

Genom hela projektets gång, men framförallt under utvecklingsfasen av spelprototypen, gjordes flertalet implementationer. Allt från magnetiska väggar och elektriska fält till spelarrotationer och omloppsbanor, vilka beskrivs i detta avsnitt. Syftet är att läsaren ska kunna koppla ihop teoriavsnittet 2 med implementationerna. Detta för att förklara närmare de matematiska knep som användes för att ge realistisk fysik i spelprototypen, men med förenklade ekvationer och färre beräkningar.

Dessa förenklingar och approximationer leder till en del begränsningar och problem, vilka diskuteras närmare i diskussionsavsnitt 5.2 om implementationen.

3.3.1 Implementation av grundläggande spelmekanik

Unity har en inbyggd fysikmotor som användes i projektet. I huvudsak användes så kallade *rigidbodies* som kan kollidera med varandra och röra sig på ett realistiskt sätt baserat på krafter och rörelsemängd [26]. Detta användes för att simulera kollision och gravitation. Det krävdes dock extra arbete för att få avataren att attraheras mot planeterna på ett korrekt sätt, vilket beskrivs i avsnitt 3.3.2.

Spelprototypens grundläggande mekanik går ut på att avataren befinner sig i tyngdlöst tillstånd i rymden. Avataren kontrolleras genom att utföra drag som applicerar en kraft. Ett drag utförs genom att spelaren håller ned musknappen eller fingret över avataren för att sedan dra åt det motsatta hållet den önskade kraften ska appliceras. Kraften kommer vid släppt musknapp eller lyft finger att appliceras som det dragna avståndet i kvadrat vilket ger precision för små manövreringar men också en möjlighet att applicera stora krafter.

Om avataren redan har en hastighet $\vec{v}$ kommer en vinkel θ mellan den och kraften $\vec{F}$ först sätta farten $|\vec{v}|$ enligt

$$\left(\frac{\theta^2}{10} - \frac{\pi \cdot \theta}{5} + 1\right)\vec{v}. \tag{3.1}$$

Detta på grund av den ointuitiva känsla som uppstår när spelaren vill byta färdriktning motsatt avatarens nuvarande färdriktning, men behöver då flera drag. För att avataren inte ska kunna skjutas iväg med allt för hög hastighet finns också en begränsning på hur mycket kraft som kan appliceras. Denna maximala kraft representeras visuellt med en semitransparent cirkel runt avataren.

För att spelprototypen skulle fungera på en mängd olika skärmstorlekar krävdes också att avataren alltid skulle kunna skjutas iväg med maxfart. Detta medför att radien på den semitransparenta cirkeln, som visualiserar området man kan skjuta iväg avataren med, alltid understiger halva skärmens höjd eller bredd. Detta genom att först jämföra höjden och bredden för att ta reda på vilken som är störst. Detta leder till att avataren alltid kommer att kunna nå maximal ivägskjutningsfart i godtycklig riktning, oavsett skärmens bildförhållanden.

För att ge spelaren en intuitiv förståelse för hur gravitationen påverkar avataren implementerades en streckad linje, med utgångspunkt från avataren. Linjen approximerar den bana avataren kommer få efter ivägskjutning med hänsyn till de yttre krafterna. För att rita upp linjen används en grupp av tio punkter som endast visas vid uppladdning av ivägskjutning. Varje punkt uppdateras sedan med de krafter som påverkar den för att sedan uppdatera dess position. Punkternas positioner kommer påverkas av kraften som ivägskjutningen kommer ge då spelaren släppper musknappen eller fingret samt av övriga yttre krafter såsom gravitationen.

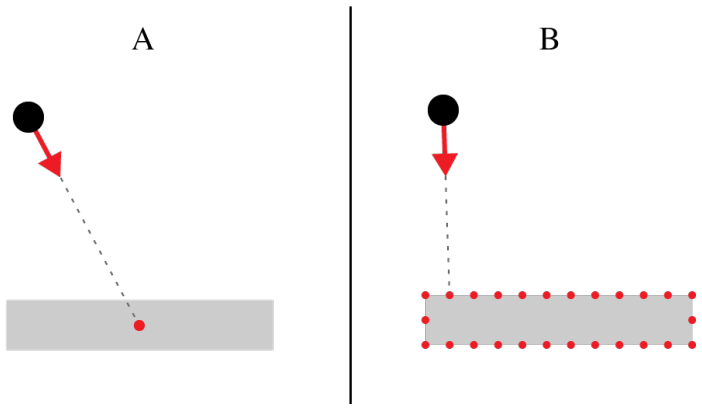
Eftersom att varje punkt ska representera avatarens position efter en viss tid måste den påverkas av tidigare punkters positioner. För att uppnå detta har varje punkt, utöver en position, en hastighetsvektor som talar om vilken riktning och hastighet avataren kommer att färdas i vid tillfället den når punkten. Varje punkt uppdaterar sin position relativt den förra, vilket innebär att den utgår från den föregående punktens position för att sedan flyttar positionen enligt den förras kraftvektor. Efter att positionen är satt utifrån föregående punkts värden uppdaterar punkten sin position utifrån övriga krafter som gravitation från planeter eller kraft från attraherande eller repellerande block. Slutligen skickar punkten sin position samt kraftvektor till nästkommande punkt vilken utför samma funktion. En grov representationen av systemets funktion finns i A.1.

Spelarens avatar består av en *ragdoll* i stilen av en streckfigur. *Ragdollen* består av 10 lemmar, två för vardera ben och arm, en för bålen samt en för huvudet. Karaktären har ett enkelt visuellt utseende som är uppbyggt av enkla geometriska

former i svart färg. *Ragdollen* har implementerats genom att varje enskild lem har tilldelats en *rigibodykomponent* för att sedan sammankopplas med överstående i lemhierarkin. Alla krafter som avataren utsätts för tillsätts sedan endast till bålen. För att undvika de svängningar som uppstår från de övriga lemmarnas rörelse sätts deras massa till noll. Detta gör deras verkan på bålen, den huvudsakliga rigidbodyn, försumbar och minskar deras funktion till att endast påverka kollisioner samt ge en visuell effekt.

För att ge spelaren mer kontroll implementerades en funktion för avataren att stå på samtliga ytor vid kollision. Detta implementerades genom att från kollisionstillfället, fram tills kontakten upphör, dämpa avatarens hastighet helt och alltså slå ut alla krafter som avataren påverkas av.

En annan funktion som infördes för att ge spelaren bättre kontroll var implementationen av en *tidskontroller*. Tidskontrollern kan användas för att sakta ner spelprototypens tidshastighet för att ge längre tid till nya drag vid snabb rörelse. Tidskontrollern implementerades genom en anropsbar metod med parametrar som bestämmer tidsskalan samt hastighet för tidsförändring. Tidskontrollern sätter sedan tidsskalan som mål och påbörjar en mjuk övergång mot denna baserat på den hastighet som har angetts. Den fixerade tidsskalan som används för fysiksimuleringar var också tvungen att uppdateras då den ligger på en enskild uppdateringsfrekvens som är relativ den allmänna tidsskalan. Den var tvungen att sättas som en kvot av den nya tidsskalan för att bibehålla 50 Hz. Metoden för att ändra tidsskalan kallades sedan på av spelarkontrollen vid ivägskjutning. Den anropades med en tidsskala på 5 procent då spelaren påbörjar ett drag och går tillbaka till normal vid ivägskjutning.



Figur 3.2: Figuren visar skillnaden i den kraft som appliceras på spelaren beroende på de hur de repellerande eller attraherande blocken implementeras. A visar hur kraften appliceras om den riktas mot blockets centerpunkt, B visar hur den appliceras om den riktas mot den närmsta punkten.

3.3.2 Gravitation

Gravitation är en attraherande kraft mellan all materia i universum. Som standard i *Unity* verkar den inbyggda gravitationen på alla *rigidbodies* i negativ y-led. Detta fungerar inte i rymden då attraktionen istället ska riktas mot planeter godtycklig

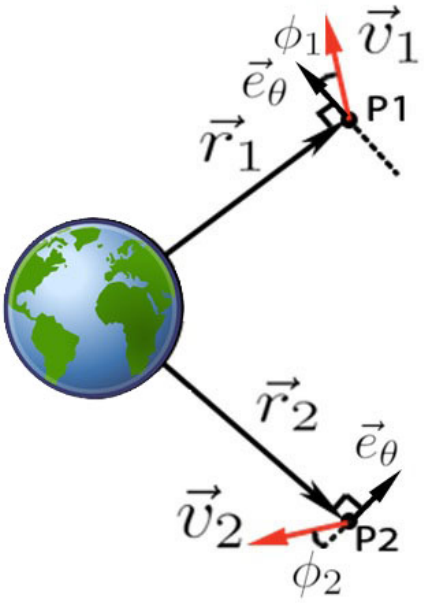
placerade i nivån. För att istället uppnå ett tyngdlöst rum med attraherande kroppar sattes gravitationsskalan till noll och gravitationskraften fick istället adderas separat till alla *rigidbodies*, baserat på avståndet till gravitationskällorna.

För att räkna ut hur stor kraft, och då hur stor acceleration, en *rigidbody* ska dras mot planeter implementerades ekvation (2.3) direkt för varje planet. För att ytterligare förenkla gravitationen, både för att spara prestanda och av spelmekaniska anledningar, fick enbart vissa objekt en *rigidbodykomponent*. Detta innebar att alla planeter implementerades som statiska objekt vilka enbart applicerade gravitationskraften på *rigidbodies* inom ett valbart avstånd, anpassat efter nivån.

3.3.3 Omloppsbana

Att hamna i omloppsbana kring en planet, likt månens omloppsbana kring jorden eller jordens omloppsbana kring solen, är en följd utav gravitationen. Då kraften från gravitationen infaller vinkelrätt mot objektets hastighet $\vec{v}$ och vid rätt förhållande mellan farten $v = |\vec{v}|$ och avståndet mellan avataren och planetens centrum $r = |\vec{r}|$, hamnar avataren i cirkulär omloppsbana. Denna effekt blev automatiskt implementerad tillsammans med gravitationen men sannolikheten att spelaren lyckades försätta avataren i en fungerande omloppsbana, var näst intill obefintlig. Detta berodde på att spelaren var tvungen att pricka in rätt v och r i realtid, då detta vanligtvis beräknas i förväg enligt ekvation (2.5).

För att göra det enklare för spelaren försätta avataren i en cirkulär omloppsbana gjordes därför en funktion *orbitalCorrection()*. Dess uppgift är att vid rätt villkor, vilka bygger på teoriavsnitt 2.2.1, forcera en cirkulär omloppsbana. Detta genom att applicera en kraft mot planeten, om avataren tenderade att åka ifrån den, eller en kraft i motsatt riktning, om avataren är påväg att krascha med planeten. I figur



Figur 3.3: Chans för omloppsbanor för avataren som inkommer moturs i P_1 och medurs i P_2.

3.3 ses de två fallen då avataren inkommer motsols mot planeten i punkten P1 och medsols i punkten P2. Deras respektive ortsvektorer $\vec{r}_1$ och $\vec{r}_2$ samt deras respektive hastigheter $\vec{v}_1$ och $\vec{v}_2$ analyseras för att bestämma om, och i så fall i vilken riktning, kraften ska appliceras. Vektorerna nedan utan index svarar mot godtyckligt index 1 eller 2. Vektorn $\vec{e}_\theta$ är alltid vinkelrät mot $\vec{r}_1$ och $\vec{r}_2$ med längd $|\vec{e}_\theta| = 1$. Om korrektionen ska ske bestäms utav två valda parametrar m och w, vilka hör till hur stor vinkeln ϕ får vara respektive hur mycket $v(r)$ och $r(v)$ får avvika från varann men ändå uppfylla ekvation (2.5).

Genom att undersöka skalärprodukten mellan $\vec{e}_\theta$ och $\vec{v}$ defineras m enligt

$$\phi = \left| \arccos\left(\frac{\vec{e}_\theta \cdot \vec{v}}{|\vec{v}|} \right) \right| < m \tag{3.2}$$

och kan väljas godtyckligt liten. Ett större m leder till att korrektionen sker vid en större avvikande vinkel ϕ.

För definitionen av w gjordes först en omskrivning av ekvation (2.5) genom att dividera med r vilket ger uttrycket

$$1 = \frac{GM}{rv^2}. \tag{3.3}$$

Detta leder då till definitionen

$$1 - w < \frac{GM}{rv^2} < 1 + w, \qquad \text{för } 0 \leq w < 1 \tag{3.4}$$

där ett större w ger korrektion vid större avvikande värden på v och r.

Riktningen på kraftvektorn beräknas genom analys av kryssprodukten mellan $\vec{e}_\phi$ och $\vec{v}$, vilka ligger i xy-planet. Tecknet på z-komponenten hos den nya vektorn som kryssproduktens ger, bestämmer riktningen på kraften beroende på fall i figur 3.3. Även styrkan på kraften hos korrektionen kan ändras genom att variera värdet på en variabel c, vilken multipliceras med kraftvektorn.

3.3.4 Elektriska och magnetiska fält

I spelprototypen har avataren givits en laddning q som kan anta värdena -1, 0 och +1, det vill säga att styrkan inte spelar roll utan enbart polariteten. Eftersom elektriska och magnetiska fält påverkar laddningar innebär detta att avataren kommer påverkas på samma sätt som laddningar enligt teoriavsnittet 2.4.

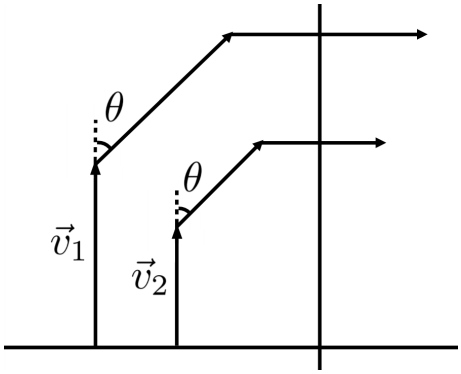
Ett elektriskt fält, implementerat i spelprototypen, adderar en kraft $\vec{F} = q\vec{E}$ på avatarens *rigidbody* inom fältet, enligt ekvation (2.13). Detta medför att hastighetsvektorer hos avataren, som inte är parallella med fältet, kommer att roteras mot samma riktning som fältet. Rotationsvinkeln θ beror av storlekten på fältet E.

Ett magnetfält, kan rotera laddningars hastighetsvektorer beroende på laddningen q och fältets styrka B enligt teoriavsnitt 2.4. Detta medför att avatarens rotation, per tidsenhet, påverkas av fältet beroende på tecknet på q samt fältets styrka B som $\theta = qB$ [rad/s], enligt ekvation (2.20).

För att rotera avatarens hastighetsvektor $\vec{v}$ θ radianer infördes därför tiden $\Delta t = 1/50$ sekund. Funktionen *FixedUpdate()*, där alla beräkningar rörande fysik utförs i

Unity, anropas med en frekvens $f = 1/\Delta t$ [27]. I *FixedUpdate()* roteras därför $\vec{v}$ med vinkeln $\theta = qB\Delta t$ 50 gånger per sekund, vilket medför att avatarens bana svänger med en viss radie r inom magnetfältet.

Radien ökar med farten v eftersom en högre fart v_1 ger en längre sträcka under tiden Δt än en lägre fart v_2 under samma tid, vilket kan ses i figur 3.4.



Figur 3.4: Visar hur farten v_1 och v_2 bestämmer radien r i den cirkulära banan avataren får i magnetfältet. Varje rotation, θ radianer, utförs med frekvensen $1/\Delta t$.

Representationen av magnetfält i implementation är en blå rektangel som innehåller punkter eller kryss. Det elektriska fältet visualiseras istället av en röd rektangel där linjer visar fältets riktning och kraft. För en mer detaljerad beskrivning av visualisering se appendix A.2.

3.3.5 Relativitet

En annan typ av fenomen som har undersökts i projektet är relativistiska fenomen. Det som implementerades var en maxhastighet som kan tänkas vara ljushastigheten c, en hastighet genom tid som beror på hastigheten genom rum och rörliga objekt som uppfattas ha olika hastigheter relativt avataren.

Detta genomfördes genom att avatarens hastighet kontrollerades vid varje uppdatering av spelprototypen och beroende på hastigheten ändrades hur fort tiden går. Detta representerades av en timer som tickade ner olika snabbt vid olika hastigheter hos spelaravataren och när den når noll, förlorar spelaren. Avatarens hastighet jämfördes också med maxhastigheten för att bestämma om avatarens hastighet ska begränsas eftersom att den inte tillåts bli högre än maxhastigheten.

Tidsskalan för timern styrdes av en variabel som var separat från spelmotorns tidsskala vilket gjorde att annan spelmekanik inte påverkades. Exempelvis kunde den globala tidsskalan varieras för att sakta ner simuleringen i spelprototypen utan att påverka den relativistiska spelmekanikens funktion.

För att implementera de relativistiska beräkningarna användes Lorentzfaktorn (se teoriavsnitt 2.3.1) med en egendefinierad ljushastighet c. Att använda en egendefinierad ljushastighet var användbart för att kunna ställa in en maxhastighet som passar spelprototypens *gameplay*.

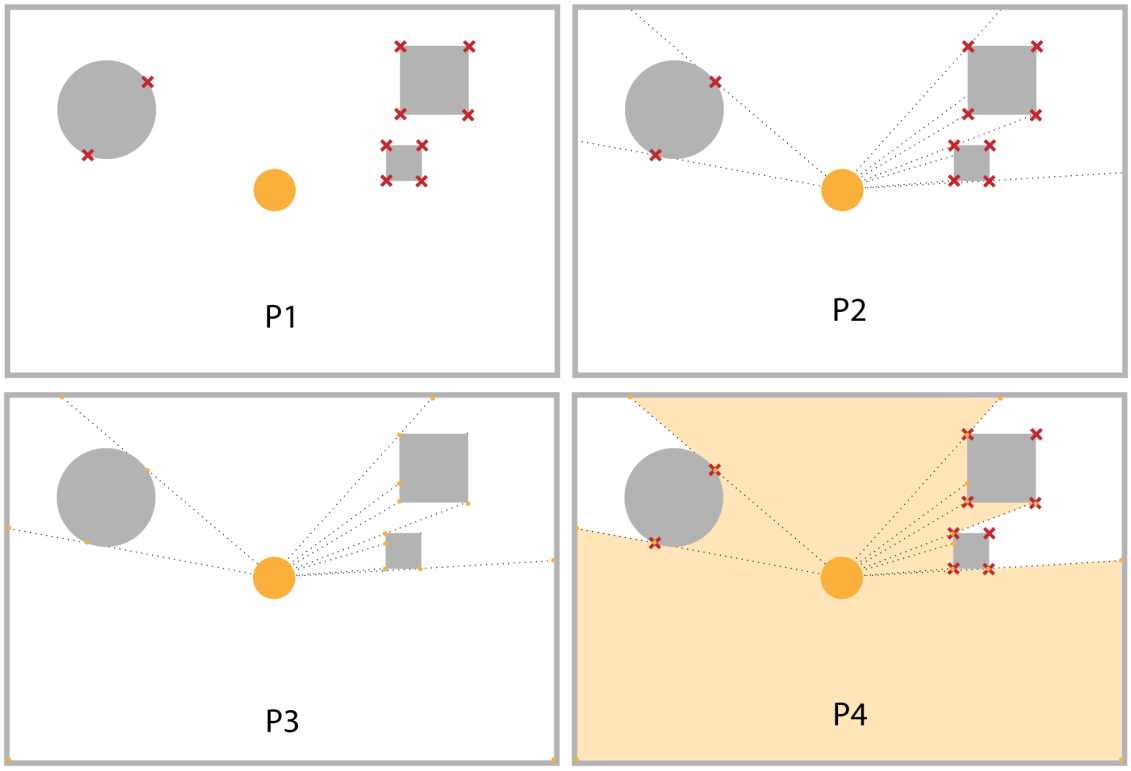
Implementationen av de rörliga objekten var i form av att planeterna modifierades till att kunna röra på sig runt det svarta hålet, vilket är målet i varje nivå.

Deras grundhastighet kopplades till en variabel som kunde ställas in för att skapa olika nivådesigner. Planeternas hastighet varierades sedan vid varje speluppdatering beroende på avatarens hastighet.

3.3.6 Strålning

För att kunna visualisera vilka områden som nås av en stjärnas strålning anvnds en synfältsrepresentation av vilka områden som påverkas av ljuset. Implementationen gjordes som sådan att avataren dör vid för lång exponeringstid i ljuset.

För att implementera detta används en algoritm som skapar en form som representerar ljuset baserat på ett visst antal punkter. För att få fram punkterna som formen ritas ut mellan använder man sig av *raycasting* utifrån en stjärnas position, i en cirkulär bana. Ett simpelt sätt att utföra detta på är genom att *raycasta* hela varvet runt i en viss gradintervall. Detta medför dock en stor prestandaförlust då detta innebär tusentals *raycasts* åt gången, 60 gånger per sekund. En mer prestandaeffektiv lösning är att enbart *raycasta* på de punkter som utgör alla solida objekts hörnkanter (se figur 3.5).



Figur 3.5: Fyra figurer som beskriver algoritmens process. P_1 visar de punkter som algoritmen hämtar från alla solida objekt. P_2 visar den *raycasting* som sker från en stjärnas mittpunkt mot de fördefinierade punkterna. P_3 visar punkterna som träffas av raycasten. P_4 visar hur en form sedan ritas ut mellan de träffade punkterna.

Detta innebär att i fallet som syns i figur 3.5 kommer endast 14 punkter tas i hänsyn; de fyra punkterna som utgör den rektangel som täcker skärmen samt de

tio punkter som hämtas från alla solida objekt. Punkterna från rektanglar tas fram genom att helt enkelt räkna ut dem baserat på rektangelns position och storlek.

För att få fram de två punkterna som representerar cirkelns relativa hörn används ekvationen nedan där $\vec{d}$ är vektorn från cirkeln till stjärnan och r är radien på cirkeln. Ekvationen ger två punkter $p_{1,2}$, vilka är cirkelns relativa hörn till stjärnan.

$$p_{1,2} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \frac{\vec{dr}}{|\vec{d}|}, \text{ där } \theta = \pm \arccos \frac{r}{|\vec{d}|} \tag{3.5}$$

För varje *raycast* som går mot ett objekts hörnpunkt *raycastas* det ytterligare två gånger, båda förskjutna en halv grad positivt och negativt från punkten. Detta är för att kunna få fram punkten bakom hörnet då den första *raycasten* kommer att träffa objektets hörn.

Alla punkter som träffas av *raycasts* sparas sedan ner i en lista och sorteras baserat på deras vinkel. Dessa punkter tillsammans med en stjärnas mittpunkt används sedan som *vertices* vid uppritande av en mesh där varje triangel är uppbyggd av två intilliggande *raycastpunkter* samt en stjärnas mittpunkt. Resultatet blir en mesh som representerar ett synfältsområde från en stjärna.

Interaktionen med avataren fungerar på ett liknande vis men hanteras separat från uppritningen. I detta fallet används bara en *raycast* som skickas i avatarens riktning och dödar avataren såvida den inte träffar ett solitt objekt.

3.3.7 Nivådesign

Spelprototypen består av flertalet fysikaliska fenomen. För att fullborda deras implementation krävdes att nivåer designades som involverar dem för att knyta samman den grundläggande mekaniken med de fysikaliska fenomenen och även för att kombinera de olika fenomenen med varandra. Dessa nivåer kunde sedan byggas till spelprototypen och implementationen av all spelmekanik blev då realiserad.

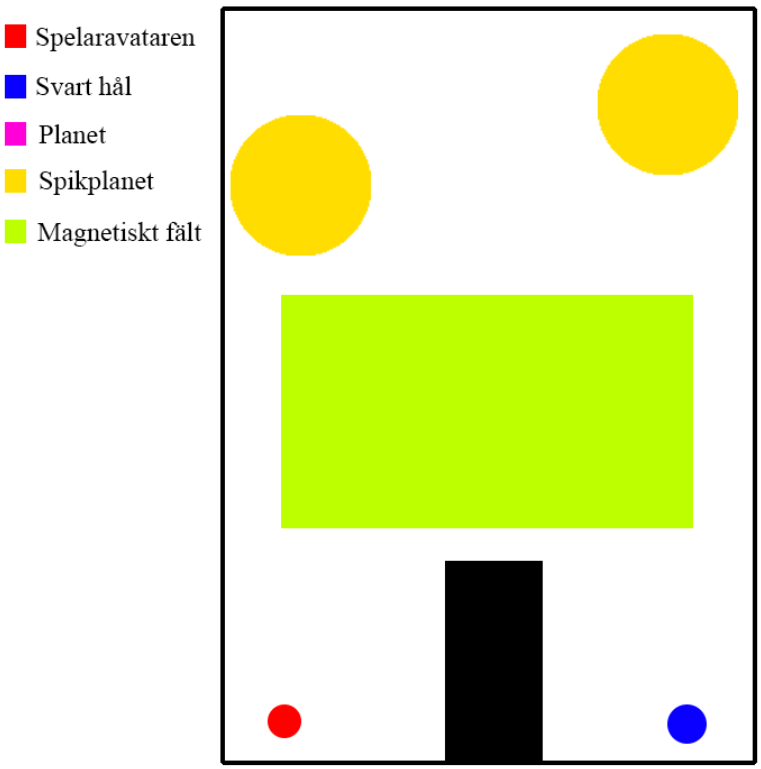
För att designa nivåerna skapades en del verktyg i form av spelkomponenter som inte var en del av den grundläggande mekaniken och som inte heller var fysikaliska fenomen. Väggar, spakar, spikar och attraherande eller repellerande block utgjorde dessa komponenter.

Väggar användes för att skapa avgränsningar till nivåer för att förhindra avataren från att nå vissa områden. Detta användes för att tvinga avataren åt ett visst håll för att interagera med ett fysikaliskt fenomen och för att förhindra avataren från att åka åt fel håll. Väggar kan även implementeras för att röra på sig och för att blockera strålning vilket skapade designmöjligheter för att demonstrera strålningsmekaniken.

För mer interaktion med omvärlden skapades ställningsbara spakar som låter spelaren aktivera eller avaktivera olika objekt, som attraherande eller repellerande block och plattformar i rörelse, när avataren kolliderar med spakarna. De användes också för att ställa om riktningar hos kraftfält.

Spikar skapades för att introducera ett hinder för spelaren. De dödar avataren vid kontakt och de kan fästas på väggar och planeter. För att representera områden på planeter och väggar används en enkel visuell representation av spikar som ritas

ut som rader av trianglar. För att slippa lägga ut dessa för hand skapades ett skript där man genom att specificera ett område, en av de fyra sidorna på rektanglar eller olika områden mellan två vinklar på planeter, kan skapa spikar via skriptet.



Figur 3.6: Exempel på en skiss för en nivådesign. Målet med den här nivån var att introducera spelaren till magnetiska fält genom att få spelaren att skjuta avataren in i fältet och märka dess effekt på avatarens rörelse.

På de specificerade områdena skapas spikar vid uppstart av spelprototypen. Spikarna dödar avataren vid kontakt vilket i sin tur deaktiverar spelarens avatar, aktiverar en partikeleffekt samt skapar ett antal ben med slumpartade kraftvektorer, så att de flyger iväg i olika riktningar. Benen är likt avataren tilldelade *rigidbodykomponenter* vilket låter dem bli påverkade av gravitation.

Attraherande och repellerande block skapades för att bilda mer avancerade hinder. Blocken fungerar på ett liknande sätt som gravitation, som tas upp under sektion 3.3.2, där den kraft blocken lägger på *avataren* avstår med distans. Blocken är implementerade med en *PolygonCollider2D* [28] som består av ett antal punkter där den punkt som är närmast *avataren* kommer anntingen attrahera eller repellera *avataren* istället för att utgå från blockens mittpunkt vilket ger en oönskad effekt, se figur 3.2. Detta beror på *avatarens* laddning q och blockens *state* s vilket kan vara negativ eller positiv. Om både q och s har samma tecken kommer de repellera varandra, om tecknen är olika kommer de istället attraheras.

Med hjälp av spelkomponenterna och de tidigare beskrivna implementationerna av de fysikaliska fenomenen och den grundläggande mekaniken designades nivåer genom att testa de olika delarna i spelmotorn och sedan skissa upp nivådesigner på

papper. Skisserna bestod av en visuell representation av utformningen av varje nivå och en förklarande text för att förklara idén bakom nivån (se Figur 3.6).

För att demonstrera implementationen av de olika fysikaliska fenomenen med en tanke på spelupplevelsen som helhet så designades först en rad simpla nivåer som introducerar den grundläggande mekaniken. Detta för att göra den tydlig för spelaren och för spelaren att lära sig den.

Sedan designades nivåer för att gradvis introducera nya fysikaliska fenomen för att göra demonstrationen av dem tydlig och för att spelaren inte ska bli överväldigad med mycket ny mekanik på samma gång. Nivåerna anpassades till det fysikaliska fenomenet nivån designas för. Exempelvis behövde nivåerna för de relativistiska fenomenen mycket anpassning. Stora nivåer gjordes för att ge mer rum för hastighetsförändringar, för att ge en möjlighet att kunna nå maxhastigheten och för att rörliga planeter skulle få mer plats att röra på sig. Utan denna anpassning var den relativistiska spelmekaniken betydligt mindre märkbar och nivån var då inte en adekvat demonstration av spelmekanikens funktion.

De andra fysikaliska fenomenen krävde annan anpassning för nivådesignen för att fenomenen skulle demonstreras. Nivådesignen för gravitation och omloppsbanor behövde planeter i nivåerna som spelaren behövde navigera runt. Elektriska och magnetiska fält behövdes placeras ut på positioner i nivåerna där spelaren inte kunde undvika dem. Nivådesignen runt strålning baserades på planeter och väggar som agerar skydd mot strålningen och som spelaren behöver flytta spelaravataren mellan på vägen till målet.

Slutligen designades nivåer där all implementerad mekanik kombinerades. Det innebar också att de anpassningar som gjordes för de olika fysikaliska fenomenen i de tidigare nivåerna behövde kombineras.

4

Resultat

Nedan presenteras de resultat som projektet givit upphov till. Först presenteras resultatet för spelprototypen och hur den fungerar, varefter resultatet för undersökningen med tillhörande dokumentation redogörs. Detta för att särskilja resultaten kopplade till respektive huvudsyfte.

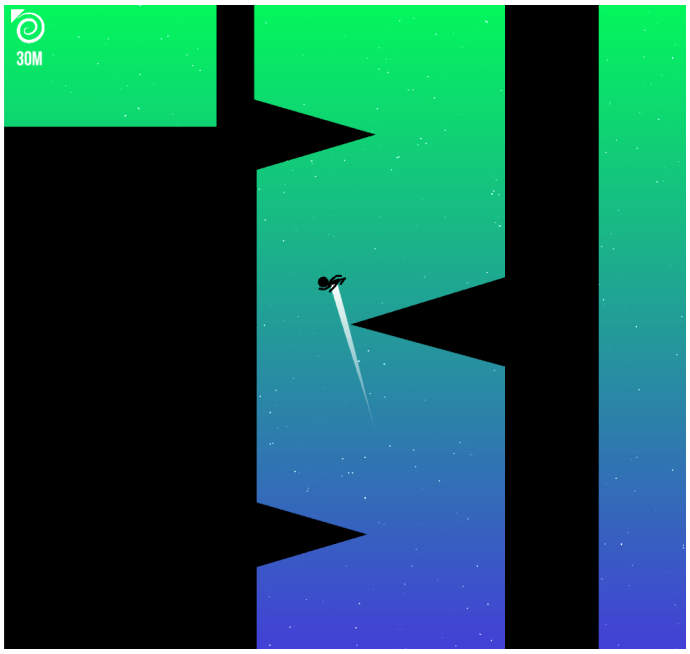
4.1 Resultat för spelprototypen

Resultatet som följde för att uppnå första huvudsyftet hos projektet är spelprototypen *Moderately Accurate Human Space Golf Simulator*, till plattformarna *Windows* och *Android*. Spelprototypen är uppbyggd av en mängd mindre nivåer med olika pussel baserade på fysikaliska fenomen som alla äger plats i rymden. Spelaren kan använda sitt finger eller en muspekare för att kontrollera *avataren* som startar varje nivå i tyngdlöst tillstånd. Varje nivå har som mål att spelaren ska styra sin avatar från dess startposition, förbi olika hinder och pussel för att försöka nå ett svart hål. Det svarta hålet kommer i kontakt med avataren avslutar nivån och låser upp nästkommande nivå. Nivåerna har en stegrande svårighetsgrad där fler och fler fysikaliska fenomen introduceras i takt med att spelaren tar sig fram genom spelprototypen. I figur 4.1 visas en av dessa nivåer.

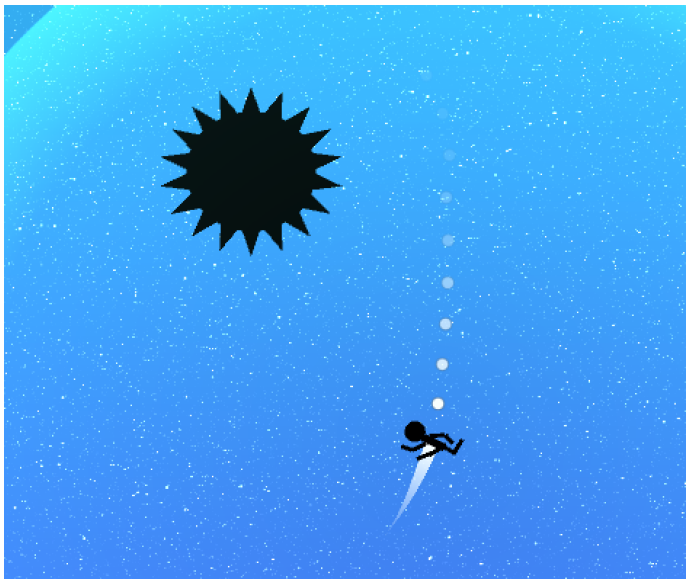
Spelaren kontrollerar avataren genom att dra fingret eller muspekaren med utgångspunkt från området runt spelaren. Så länge spelaren håller ner fingret eller trycker ner vänstra musknappen uppstår en prickad linje som approximerar hur spelaravataren kommer att röra sig vid ivägskjutning. Detta visas i figur 4.2. När spelaren släpper musknappen eller lyfter fingret skjuts spelaren iväg åt motsatt riktning med en kraft som är proportionell mot den distans spelaren drog musen eller fingret.

Spelprototypen är uppbyggt runt tre huvudsakliga fysikaliska fenomen: gravitation, elektriska och magnetiska fält samt strålning. Spelets första nivåer utspelar sig utan något av de huvudsakliga fenomenen, alltså även i gravitationslöst tillstånd. Detta för att introducera spelaren till spelmekaniken och sedan introducera fenomenen i tur och ordning för att få en mjuk inlärningskurva.

De fysikaliska fenomenen är implementerade på ett i grunden fysikaliskt korrekt sätt men är till viss del korrigerade för att dels fungera spelmekaniskt men även för att bidra till ett bra *gameplay*. Detta för att uppnå det första huvudsyftet. Det är till exempel enklare att nå omloppsbanor runt planeter på grund av den bankorrigering vilken presenterades i avsnitt 3.3.3 och kan ses i figur 4.4.



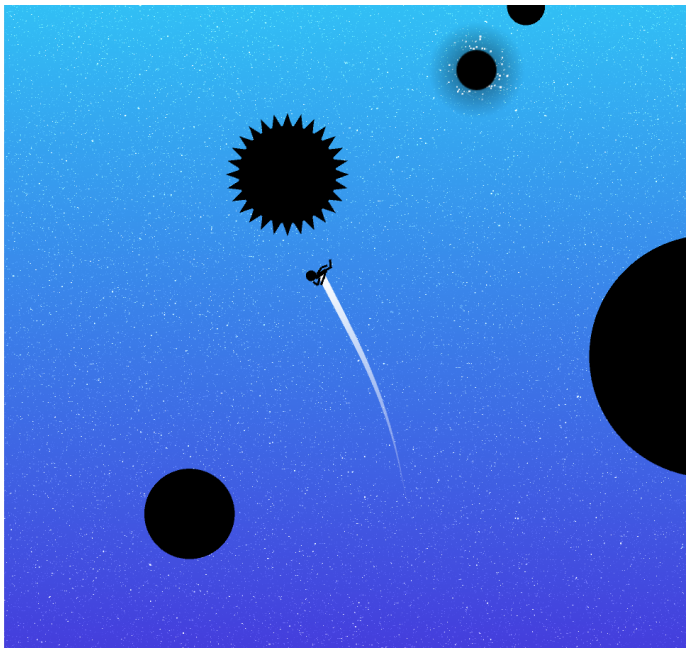
Figur 4.1: Skärmdump från en av spelprototypens tidigare nivåer. Avataren befinner sig i viktlöst tillstånd och inga andra fysikaliska fenomen har ännu introducerats.



Figur 4.2: Skärmdump som visar den approximerade banan vid ivägskjutning.

De olika objekt som *avataren* kan interagera med i spelprototypen är

- Planeter
- Stjärnor
- Elektriska och magnetiska fält
- Geometriska block
- Strömbrytare
- Spikar



Figur 4.3: Skärmdump från en en av spelprototypens nivåer efter att gravitation har introducerats.

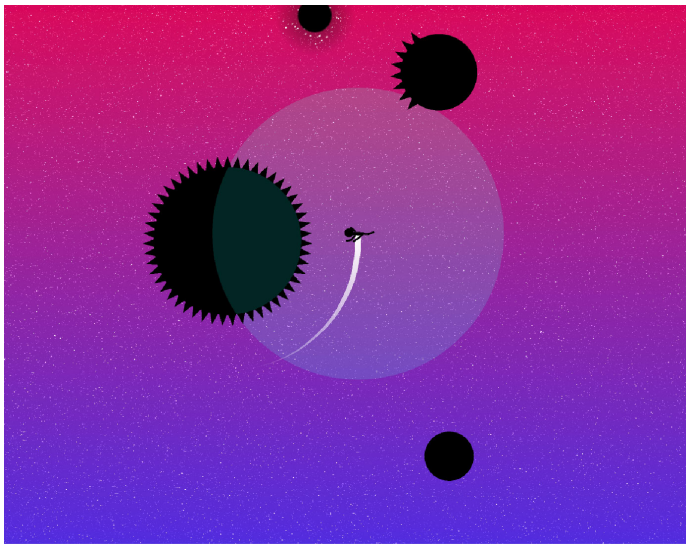
- Attraherande och repellerande block

vilka kombineras i de olika nivåerna. I figur 4.3 kan man se ett av sätten spikar har implementerats, i detta fallet utplacerade runt en planet. De geometriska blocken har använts på flera olika sätt. Allt ifrån väggar och hinder till dynamiska paneler, vilka sätts i rörelse via strömbrytare.

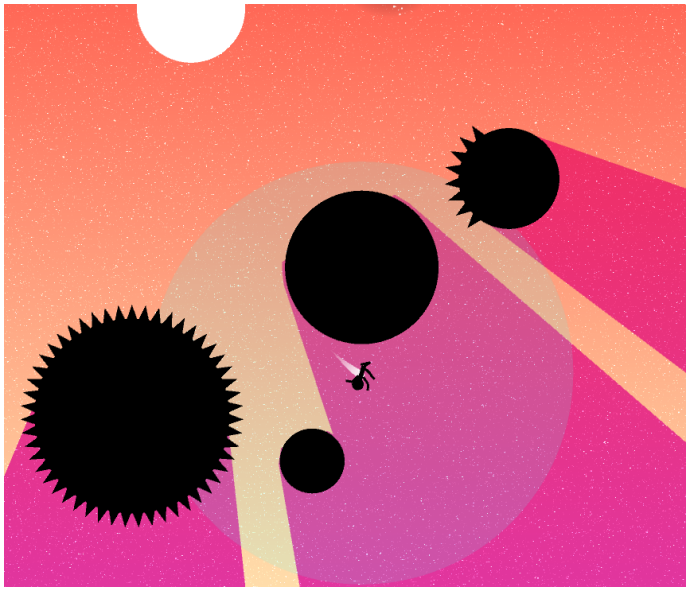
Nästan alla solida objekt i spelprototypen, som till största del utgörs av planeter, kan *avataren* stå på. Undantaget är de planeter och andra geometriska ytor täckta av spikar, vilka finns där som hinder.

Implementationen kring de elektriska och magnetiska fälten följer teorin i avsnitt 2.4 men har modifierats enligt implementationsavsnitt 3.3.4. Dessa fält påverkar avataren på ett trovärdigt sätt, trots dessa modifikationer. När *avataren* rör sig genom magnetiska fält roteras hastighetsvektorn beroende på styrkan av fältet. När *avataren* rör sig in i ett elektriskt fält, vilket kan ses i figur 4.5, tillsätts en kraft på avataren åt det håll fältet är riktat, vilket accelererar avataren i fältets riktning. Vid kombination av elektriska och magnetiska fält får avatarens bana den verklighetstrogna effekten av spiraler.

I spelprototypens senare nivåer introduceras det sista fenomenet, strålning, för spelaren, vilket kan ses i figur 4.6. Dessa nivåer är uppbyggda runt en stjärna som avger strålning. Strålningen definieras som ett område i stjärnans siktlinje. Om avataren befinner sig i stjärnans siktlinje dör den och omgången är förlorad.



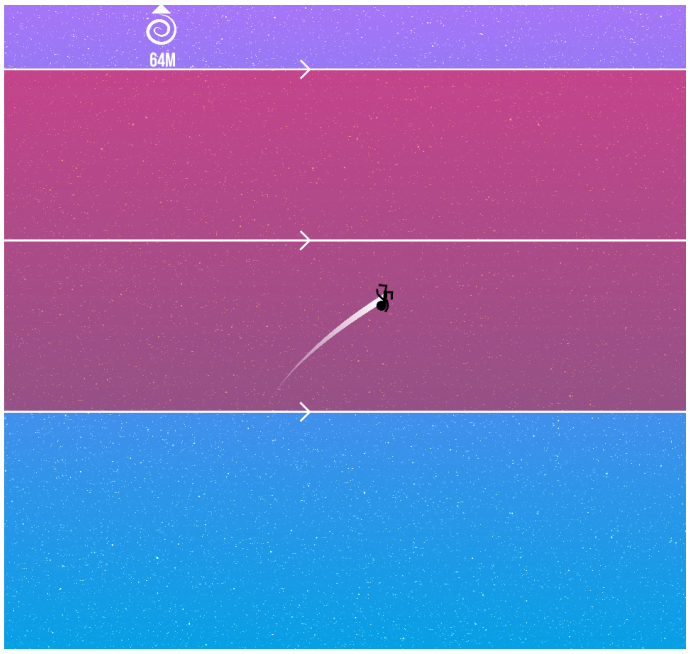
Figur 4.4: Skärmdump från en en av spelprototypens nivåer. I skärmdumpen har *avataren* en omloppsbana runt en planet som har korrigerats av *Orbital Correction()*.



Figur 4.6: Skärmdump från en av spelprototypens nivåer efter introduktionen av stjärnor som avger strålning

4.2 Dokumentationen om Fysikaliska fenomen i spel

Nedan presenteras resultatet för de olika fysikaliska fenomen funna i olika datorspel. I tabell 4.1 visas de olika fenomenen och i vilket spel de finns implementerade. Längre ner finns också detaljerade beskrivningar av hur dessa spel implementerat fenomen i avsnitten under. Vissa fenomen fanns i flera spel medan andra var exklusiva i just sitt spel. För mer ingående information kring några av de fysikaliska fenomenen som



Figur 4.5: Skärmdump från en av spelprototypens nivåer som behandlar elektriska fält. I skärmdumpen representeras det område som det elektriska fältet verkar i röd färg med pilar i fältets riktning. Avataren accelererar åt höger i bilden.

studerats nedan, finns i teoriavsnitt 2.

Tabell 4.1: Fysikaliska fenomen i spel.

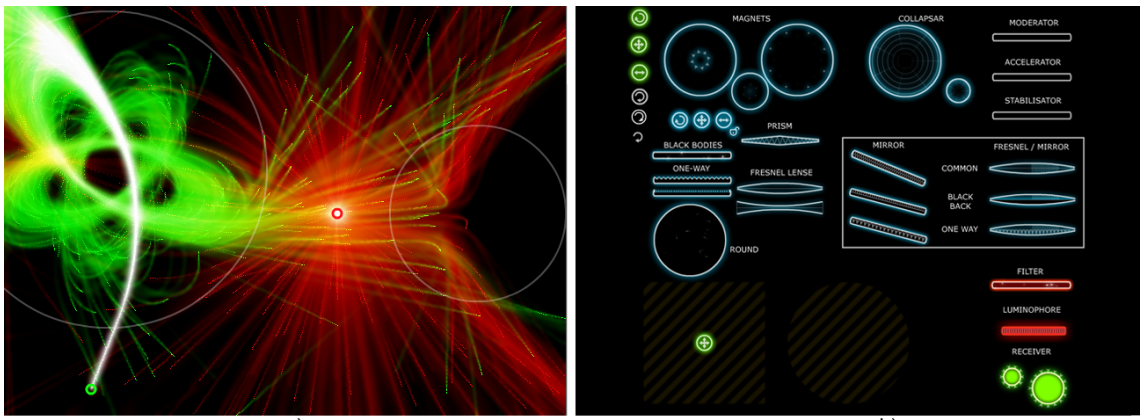
Fysikaliskt fenomen	**Implementerat i spelen**
Searchlight effekten	A Slower Speed of Light
Doppler effekten	A Slower Speed of Light
Längkontraktion	A Slower Speed of Light
Tidsdilation	A Slower Speed of Light
Optiska komponenter såsom linser, filter, prismor	Optika
Hållfastighet	Poly Bridge, World of Goo, Kerbal Space Program
Fjädrar	Poly Bridge, World of Goo
Gravitation	Poly Bridge, World of Goo, Kerbal Space Program Angry Birds Space A Slower Speed of Light
Magnetfält	Optika
Strömningsmekanik	Kerbal Space program, World of Goo

4.2.1 Ljuseffekter och Relativitet

Våren 2013 släpptes betaversionen av spelet *A Slower Speed of Light* från spelföretaget *MIT Game Lab*™, vilket är ett samarbete mellan ingenjörs- och dataveten-skapsavdelningen på *Massachusetts universitet* (MIT) [29][30]. Spelet skapades med syftet att demonstrera optiska fenomen kopplade till relativitet. Avataren rör sig i en tredimensionell värld och samlar klot vilka minskar hastigheten på det inkommande ljuset, alltså det ljus som visas på skärmen. Då avataren rör sig leder detta till visuella relativistiska effekter såsom färgförändringar, förändringar i ljusintensitet samt hur objekt trycks ihop eller dras ut på grund av längdkontraktion.

Beroende på hur snabbt avataren rör sig uppkommer dessa effekter och blir mer uppenbara vid fler plockade klot. Till exempel blir ljuset starkare då avataren rör sig mot objekt som sänder ut ljus, kallad *searchlight-effekten*, och *dopplereffekten* ändrar ljusets färger beroende på hastigheten. I verkligheten är ljusets hastighet c konstant vilket skiljer sig från den implementerade hastigheten som minskar med antalet klot. Detta gjordes för demonstrationens skull och dess syfte var inte att vara realistiskt, då de övriga effekterna anses mer realistiska [30]. Koden för implementationen finns fritt tillgänglig från företagets hemsida.

Ett annat spel som utnyttjat ljusfenomen är det egenutvecklade spelet *Optika* skapat av Vadim Ledyaev. I spelet finns flera optiska komponenter implementerade såsom ljuskällor, speglar, prismor, linser och ljusfilter, vilket kan ses i figur 4.7. Spelet går ut på att från ljuskällor styra ljus med rätt färg och rätt intensitet till



Figur 4.7: Spelet Optika där samtliga partiklar i a) är fotoner och i b) visas de olika optiska objekt som avataren kan interagera med. (Skärmdumpar från spelet, återgiven med tillåtelse från Vadim Ledyaev).

ljusmottagare kallade *receivers* i figur 4.7 b). Varje sekund sänder källan ut partiklar kallade fotoner där varje foton representeras av en färgpixel. En färgmolnseffekt kring pixeln gör att flera partiklar i rad ger illusionen av ett konstant ljussken. Varje foton har en position (x,y), en färg och en hastighet $\vec{v}$ där farten $v = |\vec{v}|$ är ljushastigheten c.

Dessa fotoner kan interagera med olika objekt som först och främst bygger på en grafisk linje mellan två punkter, lika stor som objektet självt. Interaktionen mel-

lan objekten och fotonerna beräknas genom att partikeln korsar denna linje och därigenom, beroende på objekt, får en önskad effekt.

Precis som i *A Slower Speed of Light* kan c ändras. Här finns magnetfält som accelererar fotonerna och ljusacceleratorer, vilket inte existerar i verkligheten eftersom ljusets hastighet är konstant [18]. Anledningen till att implementationen gjorts på detta sätt, enligt Ledyaev, är för att ”skapa extra spelmöjligheter som förbättrarar och utvecklar spelupplevelsen” (Vadim Ledyaev, personlig korrespondens, 2/5 2017).

Effekten av linser, som antingen fokuserar eller sprider ljuset i olika riktningar, har implementerats på sådant sätt att de fotoner som korsar mittpunkten på objektets linje inte påverkas. De fotoner som träffar mellan mittpunkten och ändpunkterna på linjen vrider hastighetsvektorn $\vec{v}$ beroende på avstånd från mittpunkten. Ett längre avstånd från mittpunkten bidrar till en större vridning.

4.2.2 Hållfasthetslära

I *World of Goo* ska bollar av en vätska, kallad *Goo*, kopplas samman för att skapa konstruktioner i en tvådimensionell miljö. Syftet med varje nivå i spelet är att konstruktionerna ska komma tillräckligt nära ett inlopp vilket visas i figur (4.8) och som, när detta sker, börjar suga upp vätskebollarna. När ett, för nivån, specifikt antal bollar har sugits upp har spelaren klarat nivån.

Konstruktionerna är inte statiska utan kan sättas i rörelse när de utsätts för yttre krafter i nivån, vilket kan hindra inloppet från att suga upp bollarna. Dessa yttre krafter är dels gravitation men även andra krafter, skapade av till exempel vind, vilka introduceras allt eftersom spelaren avancerar i spelet. Inom konstruktionerna hålls bollarna samman med elastiska strängar som visas i figur 4.9 [22].

Spelets utvecklare modellerade konstruktionerna som ”massa-fjäder-system” med hjälp av Hookes Lag, beskriven i ekvation (2.1) (Kyle Gabler, personlig korrespondens, 3/5 2017). Strängarna fungerar som fjädrar vilket gör att konstruktionerna kan vibrera. Tidigt i spelets utveckling modellerade spelutvecklarna strängarna så att de kunde brytas men detta valdes bort då det resulterade i att spelet blev frustrerande att spela (Kyle Gabler, personlig korrespondens, 3/5 2017).

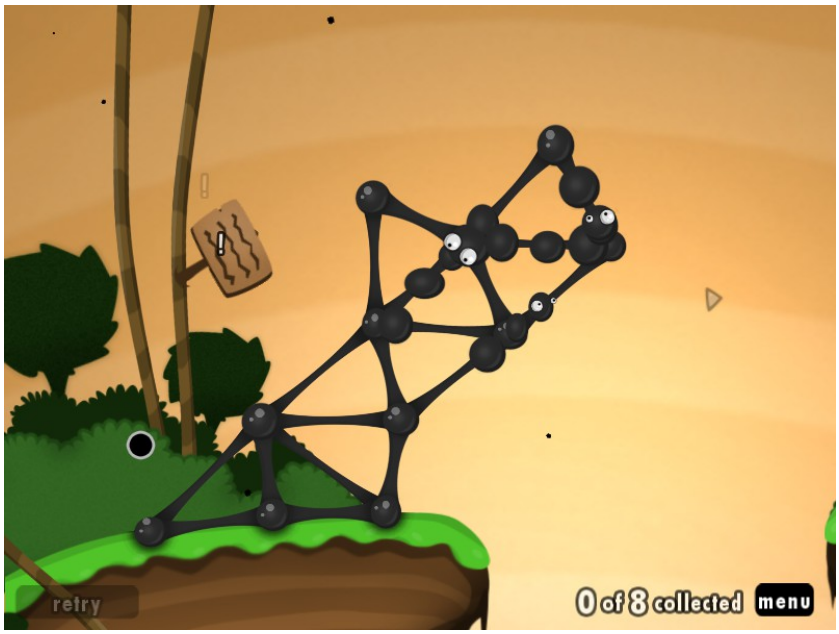
För att spelets *gameplay* skulle kännas naturligt för spelaren försökte spelutvecklarna att använda så lite undantag och justeringar av de fysikaliska formlerna i implementationen. Men för att rörelserna och vibrationerna skulle fungera valde de att justera bland annat fjäderkonstanten k i ekvation (2.1).

Spelet *Poly Bridge* [23] går ut på att bygga broar som ska klara av belastningar från diverse fordon, utan att rasa. Varje nivå i spelet har två lägen; Ett byggläge och ett simulationsläge.

I byggläget konstruerar spelaren en bro genom att koppla samman fogar med balkar av olika material med olika elasticitetsmodul E, enligt ekvation 2.2. Brons strukturella integritet avgörs av hur fogarna kopplas samman samt vilket material som använts. De tillgängliga materialen har olika egenskaper som maximalt tillåten längd på balkar och en maxbelastning [23].

Det finns även möjlighet att använda hydrauliska balkar som kan dras ihop och förlängas inom ett intervall för att skapa rörliga konstruktioner. Detta är bland

Figur 4.8: Skärmdump från nivån *Impale Sticky* i spelet *World of Goo*. Inloppet är röret i bild (Skärmdump på egen spelsession med tillstånd från spelutgivare)

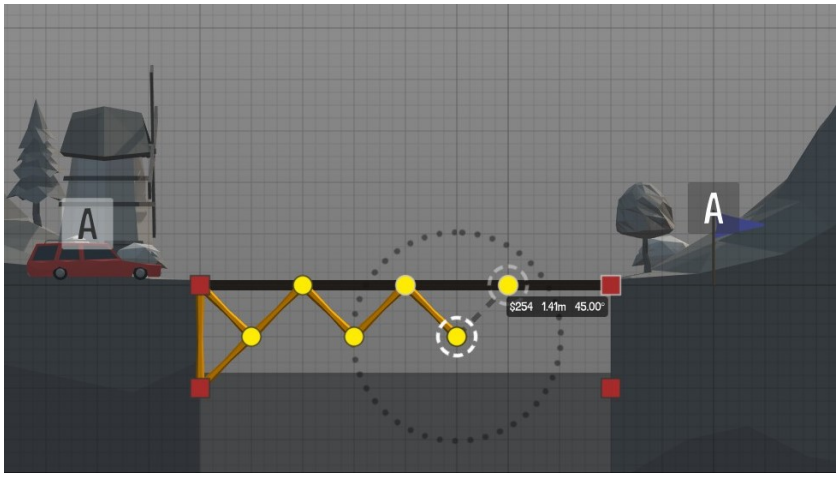


Figur 4.9: Skärmdump från nivån *Small Divide* i spelet *World of Goo*. Konstruktion av vätskebollar sammanlänkade med strängar. Vätskebollarna med ögon är de som kommer sugas in i inloppet (Skärmdump på egen spelsession med tillstånd från spelutgivare)

annat användbart på nivåer där klaffbroar kan användas för att stoppa passerande färjor från att rasera bron.

När spelaren är nöjd med sin konstruktion i byggläget testas konstruktionen i simulationsläget. Ett fordon ska då färdas över bron och nå en utsedd slutpunkt

utan att bron rasar. Håller bron genom simuleringen är nivån avklarad [23]. För att en del av konstruktionen inte ska utsättas för påfrestningar som kan leda till deformation eller brott, måste spelaren genom sin konstruktion distribuera krafterna på sådant vis att spänningen σ inte överstiger sträckgränsen σ_s, enligt teori 2.1. Ett exempel på en sådan konstruktion är att låta balkarna och fogarna bilda triangel mönster. I figur 4.10 visas byggläget med pågående konstruktion.



Figur 4.10: Skärmdump från nivå 1 i *Dry Cactus* spel *Poly Bridge*.
De röda och gula symbolerna visar fogarna och balkarnas färg varierar beroende på använt material. Den streckade cirkeln visar möjliga kontaktunker för fog av aktuellt material (Skärmdump på egen spelsession)

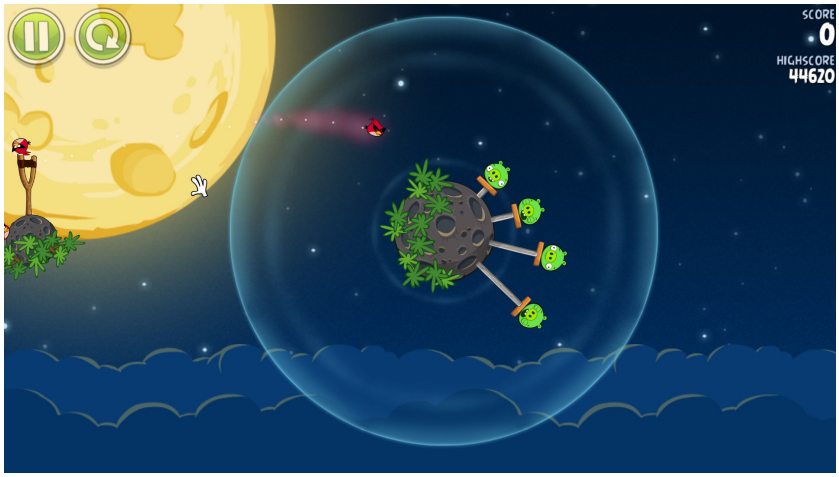
4.2.3 Gravitation

Angry Birds Space WIP
Spelet *Angry Birds Space* [24] går ut på att projektiler, i form av fåglar, skjuts med en slangbella mot måltavlor, i form av grisar i en tvådimensionell miljö. Måltavlorna kan slås ut antingen genom att träffa dem med en projektil eller genom att träffa konstruktionerna så att deras byggstenar i sin tur träffar måltavlorna. När alla måltavlor på en nivå har slagits ut är nivån avklarad. Spelaren väljer projektilens riktning och hjälps av att början av projektilens bana visualiseras av en linje som alltså beror på vilken riktning spelaren drar ut slangbellans elastiska del vilket visas i figur 4.11. Spelet utspelar sig i rymden och projektilernas bana påverkas av gravitationsfält från objekt som planeter. Detta kan användas av spelaren för att klara aktuell nivå. Ett exempel på detta är nivå 2 i spelet där faktumet att projektilen påverkas av en planets gravitation gör det möjligt för spelaren att slå ut flera måltavlor med en och samma projektil som visas i figur 4.12.

4.2.4 Strömningsmekanik

I spelet *Kerbal Space Program*, vilket har fått anställda på *NASA* att börja spela datorspel [31], bygger spelaren rymdskepp med färdiga komponenter såsom jetmo-

Figur 4.11: Skärmdump från nivå i *Rovio*s spel *Angry Birds Space*.
Den streckade linjen visar början av projektilens bana för aktuell utfallsvinkel. (Skärmdump på egen spelsession)



Figur 4.12: Skärmdump från nivå 2 i *Rovio* spel *Angry Birds Space*.
Slangbellan har här avfyrats och projektilen påverkas gravitation från planeten i mitten av bilden. Här kan gravitationen användas för att slå ut samtliga måltavlor med samma projektil. (Skärmdump på egen spelsession)

torer, bränsletankar, vingar och förarhytter. Målet är att bygga bättre och bättre farkoster för att utforska rymden och dess planeter.

Eftersom planeten Jorden har en atmosfär kommer luften påverka farkosten, vilket speltillverkaren *Squad* har valt att implementera för att ”göra spelet mer roligt” [32]. Luftmotståndet påverkar farkosten med en lyftkraft, en bromskraft och vid höga hastigheter kan farkosten ta skada och brinna upp, på grund av luftens friktion mot skeppet. Detta kann ses i figur 4.13.

Ett problem speltillverkaren hade vid en uppdatering av strömningsmekaniken var att spelare byggt sina farkoster enligt de gamla implementationerna och att nya implementationer måste då göras accepterbara av spelarna.

Lyftkraften F_l implementerades så att den är proportionell mot hastigheten v i kvadrat och att samtliga komponenter som kan bidra till kraften beror på deras

Figur 4.13: En rymdkapsel, i spelet *Kerban Space Program*, är påväg in i Jordens atmosfär där luftmotståndet skapar friktion mellan luften och kapseln, vilket resulterar i att kapseln börjar brinna. Från [33]. CC-BY-SA

yta. Detta bidrog till att farkoster kunde flyga snabbare nära marken och samtidigt behålla styrförmågan vid höga höjder.

Luftmotsåndet har implementerats utan att ta hänsyn till hur strömlinjeformat farkosten är konstruerad. Detta för att spelaren ska kunna skapa orealistiska rymdskepp som ändå fungerar. *Gameplay* skulle ”minska om enbart realistiska farkostmodeller fungerar” [32].

Tidigare nämnda spelet *World of Goo* innehåller spelmoment med strömningsmekanik. På vissa nivåer av spelet försvåras målet att nå inflödet av luftströmmar, vilka förskjuter vätskebollarna i strömmens riktning [22].

5

Diskussion

I projektet implementerades flera olika typer av fysikaliska fenomen som skiljer sig kraftigt från varandra och spelprototypen är uppdelat i olika nivåer med olika fysikaliska fenomen. Därför diskuteras de olika delarna separat i detta kapitel tillsammans med diskussioner om avgränsningar & genomförande, implementationen och undersökningen. Avslutningsvis diskuteras potentiellt framtida arbete och vidare utvecklingsmöjligheter för spelprototypen.

5.1 Diskussion om avgränsningar & genomförande

Valen av avgränsningar och besluten som togs under genomförandet fungerade väl i vissa fall och mindre bra i andra. Att använda *Unity* som spelmotor var ett bra val överlag då *Unity* är en spelmotor som är relativt enkel att arbeta med även om man saknar erfarenhet. Det finns stora mängder dokumentation om hur man gör olika saker och den tillhandahåller en grund som vi inte behöver bygga själva vilket ger oss mer produktiv tid till att faktiskt implementera spelmekaniken.

Trots att *Unity* är relativt nybörjarvänlig var nog den största svårigheten med dess användning att komma förbi den initiala inlärningsprocessen då vi endast hade begränsad erfarenhet sedan tidigare. Det var förmodligen ändå det bästa tillgängliga alternativet då vi annars hade varit tvungna att bygga grunden till spelprototypen själva vilket hade tagit mer tid än inlärningsprocessen eller använt en annan spelmotor som hade haft samma inlärningsprocess. *Unity* har även bra stöd för 2D vilket vi använde för vår spelprototyp och att implementera de fysikaliska fenomen fungerade bra genom att använda C#-skript som *Unity* är byggt för att hantera. Uppdelningen i olika nivåer och hanteringen av den genom en meny fungerade också väl i *Unity*.

Att vi använde oss av 2D var ett val med stora för- och nackdelar. Den huvudsakliga fördelen med det var att det var enklare att utveckla. Detta eftersom vi endast behöver bry oss om fysikberäkningar i två dimensioner, spel- och nivådesignen blev enklare och vi behövde inte skapa 3D-modeller och 3D-miljöer. Detta gav oss mer tid till att implementera de fysikaliska fenomenen och utforska designen runt dem vilket ju är syftet med spelprototypen.

Den huvudsakliga nackdelen med det är att vi inte hade möjligheten att utforska fysikbaserad *gameplay* i 3D. Detta gör att vi inte kunde göra fullt realistiska fysiksimuleringar eftersom det kräver de tre rumsdimensionerna för att kunna efterlikna

verkligheten.

Det är oklart hur mycket mer tid det hade tagit att använda 3D istället för 2D vilket gör det svårt att bedöma om det var ett bra val att begränsa oss till 2D. 2D-utvecklingen fungerade bra och lät oss uppfylla syftet med projektet vilket skulle kunna vara tillräckligt för att kalla det för ett bra val men det är som sagt en kraftig begränsning.

Uppdelningen av spelutvecklingen till en första prototyp och sedan olika nivåer fungerade väl eftersom det gav oss en vision och en struktur till spelprototypens utformning. Den första versionen av spelprototypen etablerade på ett ungefär hur spelet skulle se ut och hur den grundläggande spelmekaniken skulle fungera. De olika nivåerna kunde sedan användas för att implementera de olika fysikaliska fenomenen och skapa nivådesigner runt dem. Det skapade också en naturlig arbetsfördelning i projektet. Alternativet till att dela upp spelprototypen i nivåer skulle varit att låta spelprototypen vara kontinuerligt. Det hade lett till en totalt sammanhängande spelupplevelse men hade gjort det svårare att tydligt hålla isär olika fysikaliska fenomen. Eftersom syftet med spelprototypen är att utforska *gameplay* kring de fysikaliska fenomenen snarare än att värna om den generella spelupplevelsen tycks uppdelningen ha varit det bättre valet.

Metodiken kring undersökningen av existerande spel har fungerat i den mån att det går att hitta många fysikbaserade spel via *Steam* och att det finns mycket information om många av dem. Vi avgränsade oss till att använda Steam för att hitta spel men inför utvecklingen av den relativistiska spelmekaniken hittade vi spelet *A Slower Speed of Light* utanför Steam och valde att inkludera det i undersökningen av existerande spel.

En svårighet blev att urskilja exakt hur ett spel implementerat ett fysikaliskt fenomen och hur det skiljer sig från andra spel som implementerat samma fenomen. Speciellt svårt blir det för de vanligaste fysikaliska fenomenen i spel som exempelvis gravitation. Det här problemet kunde motverkas genom att vidare fördjupa sin kunskap om ett spel genom att läsa på mer om det eller spela det själv. Något vi gjorde senare i projektet var att intervjua utvecklare som utvecklat spel med fysikbaserad *gameplay* om deras spel. Detta skapade en möjlighet att få en djupare inblick i hur andra skapat fysikbaserad *gameplay*.

Den uppenbara nackdelen med dessa sätt att fördjupa vår kunskap var att det var tidskrävande. Om något skulle nämnas som ett problem med den här delen av metodiken och projektet skulle det vara att vi kunde ha lagt ett något större fokus på kartläggningen av tidigare fysikbaserade spel då det är en mycket stor uppgift som är relevant för projektet.

Förhållningssättet vi hade bestämt att ha till projektet var *agilt* [34]. Det innebär att vi jobbade iterativt och inte gjorde långa detaljplaneringar utan tog beslut som sammanföll med var vi stod, alltså vad som passade och vad som var möjligt. Vi planerade en veckolång sprint, delade ut arbetsuppgifter och utförde dem så väl vi kunde innan vi fortsatte med nästa sprint. Dock så följde vi inte någon specifik agil utvecklingsmetod eller implementation såsom *Scrum*. Detta ledde till vissa problem vid utförandet av projektet som hade kunnat undvikas om vi faktiskt följt en utvecklingsmetod. Problemen var inte tillräckligt stora för att förhindra slutförandet

av projektet men det var ändå en av de mer problematiska delarna av projektet.

Vi använde oss av versionshantering i form av *Git* [35] för att samarbeta på spelprojektet i *Unity*. Det underlättade mycket med att arbeta samtidigt med samma filer och med att experimentera då det är enkelt att ångra ändringar, något som var viktigt när vi utforskade implementationen av nya fysikaliska fenomen. Användningen av *Git* för versionshantering i projektet fungerade bra överlag. Ett problem med *Git* för det här projektet var att det kunde ske konflikter mellan filer då olika personer har utvecklat olika saker med samma filer som grund. En lösning på detta var helt enkelt att personer främst använder sin egen *branch* och att arbetet sedan sätts ihop.

5.2 Implementationsdiskussion

Vid implementering av de olika fenomenen och den grundläggande mekaniken har vi fått skapa begränsningar, lösa problem och uppfylla behov. Diskussion kring dessa samt de tankegångar vi har haft vid implementationen av de olika delarna tas upp här. Primärt tas de fysikaliska fenomenen upp som har varit fokus under projektets gång.

5.2.1 Grundläggande spelmekanik

Vårt val av grundläggande spelmekanik visades bli både en tillgång och en begränsning. Att spelmekaniken och kontrollen av spelarens *avatar* hade en simpel utgångspunkt som inte var grundad i något fysikaliskt fenomen möjliggjorde att ett brett spektra av fysikaliska fenomen kunde implementeras. Problematiken uppstod i svårigheten att designa intressant *gameplay* runt kombinationen av spelarkontrollen och de implementerade fysikaliska fenomenen. Detta då *avataren* endast kan interagera med fenomenen på ett ickeprecist sätt förflytta sig genom nivån. Att bara ha interaktion baserad på *avatarens* position och samtidigt ha väldigt oprecisa förflyttningsmedel innebär att många av de implementerade fenomenen, exempelvis ljus och relativitet, bara har ett fåtal interaktionssätt. Detta till skillnad från gravitation som passar bättre med hur *avataren* styrs på grund av de avancerade rörelsemönster och omloppsbanor som kan uppstå från ett enskilt drag.

Vad vi kunde ha gjort för att undvika detta var att redan tidigt under prototypstadiet reflekterat över om spelarkontrollen var tillräcklig för att kunna utforska de planerade fysikaliska fenomenen. Då skulle vi haft möjlighet att byta kontrollmetod för *avataren* och istället tagit en annan väg med mer avancerad styrning vilket skulle låtit oss skapa intressantare interaktioner.

Sent i projektet insåg vi att endast kontroller för att förflytta sig skulle göra det svårare att skapa intressanta nivåer och vi tog beslutet att utöver rörelsekontrollerna integrera funktioner för att använda och kasta ut olika föremål. Exempel på detta hade varit att kunna byta *avatarens* laddning från negativt till positivt eller att kunna kasta ut föremål för att blockera skadlig strålning. Vad som lät bra på pappret visade sig dock inte vara möjligt att implementera utan att äventyra spelprototypens *gameplay*. Detta på grund av att *avataren* hela tiden befinner sig i en viss hastighet

vilket innebär att sikta in och kasta ut ett föremål måste göras väldigt snabbt och exakt vilket gav spelprototypen en väldigt hög svårighetsgrad.

För att lösa detta försökte vi implementera ett system för att kontrollera tiden som innebar att spelaren aktivt var tvungen att tryckta på en knapp för att få tiden att gå och på så sätt ha gott om tid att använda föremål. Nackdelen med lösningar var att spelprototypen upplevdes som väldigt långsamt. De korta nivåerna som spelaren skulle klara genom repetition klarades istället genom att spela långsamt och metodiskt.

Funktionen för att sakta ned tiden användes, trots återgången till den ursprungliga spelarkontrollen, till att underlätta för spelaren vid en ny ivägskjutning. Förändringen till 5 procent tidsskala ger spelaren tillräckligt med tid till att reagera på omgivningen och placera ut den nya kraftvektor precist. Utöver det bidrog den också till en bra visuell effekt när många partiklar och objekt är i rörelse.

De attraherande och repellerande blocken löstes på samma sätt som vi löste gravitation och i slutet har en liknande implementation. Den större begränsning hos blocken ligger vid hur objekt, i spelprototypens fall *avataren*, söker sig mot eller ifrån blocken. Blocken har tilldelats en mängd punkter och den närmaste punkten söker sig objekten till. Detta är inte helt i enighet med verkligheten då om en blocken har få punkter blir upplevelsen udda och orealistisk. Väljer man tillräckligt många punkter så kommer begränsningen inte ha en större påverkan på spelprototypens gång.

5.2.2 Gravitation och banrörelse

Flera begränsningar finns när det kommer till gravitationen och allt som påverkar den. Ett av de större problem med den implementerade gravitationen är att vid stark gravitation från en planet kan det vara svårt att komma ifrån planeten vilket försvårar nivådesignen. Det finns ingen lösning för detta i implementationen men en lösning skulle kunna vara att justera gravitationskraften för *avataren*, då den står på planeten. Till exempel genom att antingen ändra styrkan på kraften i en viss tid efter ivägskjutning eller låta kraften minskas beroende på avståndet från planeten.

Antalet planeter kan även påverka spelets gång genom prestandaförluster då det finns ett stort antal planeter på nivån samtidigt. Detta ger en begränsning för hur många planeter som kan användas vid nivådesign. Detta har dock inte varit något större problem vid nivådesign men begränsningen finns fortfarande. Ett större antal planeter kan även ge ett oönskat beteende hos *avataren* då gravitation från flera planteter påverkar *avataren* samtidigt, vilket har lösts delvis med hjälp av att svagt prioritera den planet som är starkast och närmast *avataren*.

För att få ytterligare förbättrad prestanda begränsades dessutom deras gravitationsfält till ett visst avstånd vilket inte är verklighetstroget. Detta medför en direkt ökning av gravitationskraften när en *rigidbody* når planetens gravitationsfält istället för den exponentiella ökning från noll, vilket stämmer enligt ekvation (2.3).

Bankorrigeringen *orbitalCorrection* forcerar cirkulära omloppsbanor vilket fungerade för små korrektioner. Vid för stark korrigering blir den korrigerande kraften så stor att villkoren för *orbitalCorrection* inte längre uppfylls. Men tack vare gra-

vitationskraften kan villkoren uppfyllas igen efter en kort tid och resultatet blir att *avataren* får en sinusformad rörelse runt den forcerade omkretsens linje. Detta är alltså en begränsning hos funktionen och stark korregering kan alltså inte väljas godtyckligt stor. Vid mindre värden på korregeringen fungerar funktionen som avsett.

5.2.3 Magnetism och kraftfält

Begränsningarna vid implementering av fälten påverkade inte det färdiga spelprototypen mycket och gjordes mestadel för enkelhetens skull.

De magnetiska fälten ska rotera *avataren* i en riktning beroende på *avatarens* laddning utan att påverka *avatarens* fart. Vanligtvis vill man kunna skapa en implementation från de realistiska ekvationerna i kapitel 2 vilket i vårt fall resulterade i mindre fel vid varje beräkning. De mindre felen kvarstod vid varje successiv beräkning vilket i slutändan kunde starkt påverka *avatarens* fart. Inte nog med det ökade storleken av felen om *avataren* hade högre fart. Eftersom vi endast sökte en enkel rotation som var beroende av fältets egna storhet samt *avatarens* laddning använde vi oss av en matrisrotation där vinkeln är beroende av magnetfältets styrka och *avatarens* laddning, vilket förenklade lösningen och kan även ha förbättrat spelprototypens prestanda. Detta eftersom färre beräkningar behövde göras.

Laddningen på *avataren* i implementation blev en enkel lösning där -1 står för negativ laddning, 0 står för ingen laddning och 1 står för en positiv laddning. Med detta har vi ingen storlek på laddningarna och kan därför basera styrkan av påverkan t.ex. fälten har på *avataren* genom fältens egna styrkor vilket förenklar skapande och design av nivåer.

Två behov som uppkom under utvecklingen av spelprototypen för fält var ett sätt att ändra *avatarens* laddning. Att kunna ändra *avatarens* laddning eller fältens laddning var uppenbart ett behov för att kunna designa intressanta nivåer runt fenomenen vilket ledde till utveckling av ställningsbara spakar och enklare samt eleganta sätt att ändra *avatarens* laddning under speletsgång. Eftersom båda får samma effekt på speletsgång används endast en om nivåerna ej krävde en specifikt sätt.

5.2.4 Relativitet och ljusfenomen

Det fjärde planerade och till viss del implementerade fysikaliska fenomenet till spelprototypen var relativitet. Tanken med fenomenet var att utforska funktionen mellan hastighet och tid för att bygga intressant nivådesign runt detta.

Under utvecklingen av den relativistiska spelmekaniken samt designen av de medföljande nivåerna gjordes observationer om de svårigheter som stöttes på. En sådan svårighet med att implementera relativistisk spelmekanik var att det kan vara svårt att göra den märkbar. Detta för att mekaniken kring tid och rum inte hinner märkas av vid mindre nivåer, och för att den inte är direkt uppenbar eller intuitiv.

Lösningar på den här problematiken är att skapa större nivåer med större avstånd samt att göra mekaniken mer uppenbar genom tydlig visuell representation eller

överdrivna och orealistiska värden.

Ett nytt problem som uppstår vid designen av större nivåer är att de behöver fyllas med objekt för att inte bara vara tomrum som saknar gameplaymöjligheter. Detta kan kräva en stor mängd objekt vilket gör nivådesignen till ett större och svårare arbete, och kan även ha negativa konsekvenser för prestandan i spelprototypen.

Ytterligare en observation som gjordes var att den relativistiska spelmekaniken var betydligt mer märkbar i realtid eftersom förändring i hastighet genom tid kontra hastighet genom rum blir mindre uppenbar om den sker när spelprototypen är stoppat eller har saktat ned.

Ett vidare hinder vi stötte på vid utvecklingen var att relativitet är svårt att implementera realistiskt då spelprototypen skulle vara ospelbart i hastigheter nära c. Lösningen var att istället sänka hastigheterna som krävdes för att uppleva relativistiska fenomen och omdefiniera c till en egendefinierad hastighet i alla relativistiska funktioner.

Trots de sänkta och relativt låga hastigheterna blir spelprototypen svårspelat vid de hastigheterna som krävdes för att få känslan av relativistiska fenomen. Då spelprototypen är implementerat i 2D sett ovanifrån innebär det ett begränsat synfält vilket innebär att tiden spelaren har på sig att reagera på inkommande hinder minskar linjärt mot hastigheten till skillnad från ett spel implementerat i 3D då spelaren har ett godtyckligt djupt synfält. Detta skulle kunna lösas genom att implementera en karta i spelprototypens gränssnitt som visar en överskådlig vy av nivån vilket låter spelaren vara mer beredd på inkommande hinder.

Att även implementera relativistiska fenomen runt gravitation skulle kunna vara en vidareutveckling som minskar problemet med bristen på märkbarhet eftersom de relativistiska fenomenen då skulle göras mer synliga genom att *avataren* tar sig till en planet med särskilt hög gravitation.

Ljusfenomen i form av strålning implementerades sent i projektet. Ett problem med implementationen var att det fanns en avsaknad av verktyg till att designa *gameplay* och nivåer runt fenomenet. Exempelvis skulle vi i mån av tid kunnat implementera en spelmekanik där spelaren kan lägga ut objekt för att blockera strålningen.

Av de nuvarande verktygen kunde planeter och väggar användas för att blockera strålning. Planeterna och väggarna kunde också sättas i rörelse för att skapa nivåer där spelaren måste förflytta *avataren* vid rätt tidpunkt mellan rörliga skydd.

5.2.5 Nivådesign

Designen av nivåerna var en del av projektet som var givande då det hjälpte oss att utforska fysikbaserad *gameplay*. Att designa nivåer runt den implementerade spelmekaniken var något som illustrerade hur spelmekaniken kunde användas och utnyttjas för att skapa olika situationer i spelprototypen som gav olika spelupplevelser. Det visade också på hur den inte kunde användas vilket åskådliggjorde brister och potentiella problem vid implementationen av fysikaliska fenomen. Dessa problem kan visa på potentiella svårigheter med att designa *gameplay* runt de valda fysikaliska fenomenen, både vår implementation av dem och möjligtvis också generellt.

Förutom att demonstrera användningen av de fysikaliska fenomenen i spelprototypen kunde olika nivådesigner skapa olika svårighetsgrader. Exempelvis kunde tiden som spelaren hade på sig att klara en nivå begränsas vilket minskar antalet sätt som spelaren kan klara nivån på och därmed höjs svårighetsgraden. Placeringen av hinder, både fysikaliska fenomen som kraftfält och spelkomponenter såsom spikar kunde också användas för att reglera svårighetsgraden på en nivå.

Spelkomponenterna som inte var en del av den grundläggande spelmekaniken eller ett implementerat fysikaliskt fenomen var till stor hjälp för att skapa nivådesigner. Komponenterna utgjordes av spikar, väggar, spakar, och attraherande eller repellerande block. Om vi hade implementerat fler sådana komponenter hade vi haft fler verktyg till att designa nivåer runt de fysikaliska fenomenen vilket möjligtvis hade förbättrat vår förmåga att använda spelprototypen för att utforska fysikbaserad *gameplay*. Dock var det rimligt att implementationen av de fysikaliska fenomenen prioriterades eftersom de utgör kärnan i projektet.

Implementationen av de fysikaliska fenomenen hade svårigheten att balansera realism med spelupplevelse. Nivådesignen hade liknande svårigheter med att balansera design för att demonstrera de fysikaliska fenomenen gentemot design för att skapa intressant *gameplay*. Båda designaspekterna är relevanta för projektet och båda togs hänsyn till i den grad det var möjligt. Att designa runt de fysikaliska fenomenen prioriterades över spelupplevelsen som helhet då det var mer direkt kopplat till syftet med projektet, att utforska designen runt fysikbaserad *gameplay*.

Att anpassa nivådesignen till implementationen av de olika fysikaliska fenomenen skapade svårigheter både på grund av den tidigare diskuterade bristen på spelkomponenter som kunde användas som verktyg och även på grund av att vissa kombinationer av de fysikaliska fenomenen var svåra att designa nivåer för. Detta för att anpassningarna de olika fysikaliska fenomenen krävde inte alltid var kompatibla med varandra. Exempelvis krävdes höga hastigheter och mycket rörelse i relativitetsnivåerna för att göra den relativistiska spelmekaniken märkbar medan strålningsnivåerna behövde lägre hastigheter för mer precision för att spelaren skulle kunna navigera från skydd till skydd.

5.3 Diskussion kring undersökningen och dokumentationen

Att hitta fysikaliska fenomen i datorspel via *Steams* speldatabas fungerade bra i vissa anseende men sämre i andra. En positiv aspekt var att databasen innehöll väldigt många spel och det stora utbudet gjorde det möjligt att undersöka många fysikbaserade spel. Tyvärr kostade många utav spelen pengar vilket begränsade möjligheten att spela spelen och därigenom få en bättre bild av spelens *gameplay*. Detta försvårade arbetet att möta det andra huvudsyftet om implementation kopplat till *gameplay*.

I projektet dokumenterades enbart tio olika fenomen i totalt sex spel. Detta beror främst på att de spel som undersökts har givits en mer omfattande och detaljrik beskrivning i projektet, för att möta syftet med hur och varför fenomenen imple-

menterats som de gjorts. Om en komplett dokumentation skall utföras behövs mer resurser och tid.

Vi försökte kontakta samtliga spelutvecklare till spelen som ingått i dokumentationen för att få tillgång till en intervju angående implementationerna. Av dem vi fick svar från, genom mejl eller telefonsamtal, var det enbart en som ställde upp på intervju via telefon. De flesta ville hjälpa men kunde enbart nås via mejl. Dock fick vi mycket bättre förståelse för hur implementatörerna resonerade kring implementationerna av de fysikaliska fenomenen och vilken förundersökning de gjort angående teorin.

Om en första version av en mer omfattande dokumentation skulle släppas, exempelvis genom ett annat kandidatarbete, finns stor chans att fler utvecklare skulle vilja bidra för att ge sin version av sitt spel.

5.4 Framtida arbete & utvecklingsmöjligheter

En framtida utveckling av spelprototypen skulle först och främst innebära mycket arbete för att bli en färdig produkt. För att uppnå produktionskvalitet skulle inte bara gränssnitt och navigation kräva stora förbättringar utan även spelarkontroller och nivådesign skulle kräva arbete och fler iterationer baserade på användarfeedback.

Moderately Accurate Human Space Golf Simulator kan på grund av att det är byggt i *Unity* enkelt anpassas och exporteras till flera nya plattformar varav *Mac OS* samt *iOS* är de mest relevanta. Möjlighet att exportera till fler plattformar finns men innebär större ingrepp i spelprototypens kod för att anpassa spelarkontrollerna till att också stödja handkontroller.

Spelprototypen kan också, på grund av den simpla grundmekaniken, byggas ut till att stödja fler fysikaliska fenomen. Exempel på dessa är vätske och strömningsmekanik i viktlöst tillstånd samt hållfasthetslära. Det är också möjligt att bygga ut och öka interaktioner mellan olika, redan implementerade, fenomen som ljusböjningar från stark gravitation.

En annan möjlighet för vidareutveckling är att bara använda sig av en eller ett par av de implementerade fenomenen och basera ett helt spel på den. Exempelvis skulle ljusfenomenet kunna användas i ett spel för att visualisera fienders eller *avatarens* synfält.

Alla implementationer av fysikaliska fenomen kan dessutom lyftas ut från spelprojektet till fristående paket som kan användas till andra spel. Detta innebär att man kan designa helt ny *gameplay* runt många av fenomenen som har varit svåra att utforska med den nuvarande spelmekaniken.

6

Slutsatser

Vi har i spelprototypen implementerat de fysikaliska fenomenen gravitation, elektriska och magnetiska kraftfält, samt strålning. Vi har delvis implementerat relativistiska fenomen men valt att inte inkludera det i spelprototypen. Detta på grund av svårigheter med att skapa *gameplay* runt fenomenet vilket var ett syfte med projektet.

De implementerade fysikaliska fenomen i spelprototypen har vi implementerat utifrån de matematiska formler som beskriver fenomenen. Modifikationer har dock gjorts för att förbättra *gameplay*. Inget av de implementerade fenomenen är unika för vår spelprototyp utan har implementerats på liknande sätt i andra existerande spel.

Vi har haft varierande framgång att kombinera implementationen av de fysikaliska fenomenen med den grundläggande spelmekaniken. Exempelvis uppkom svårigheter vid design av *gameplay* runt relativistiska fenomen i vår spelprototyp. Detta säger inte att bra *gameplay* inte kan designas runt relativistiska fenomen, men att det måste vara kompatibelt med spelets grundläggande spelmekanik.

Designmöjligheter varierar alltså beroende på de valda fysikaliska fenomenen och valet av den grundläggande spelmekanik som fenomenen implementeras runt.

Vi har också påbörjat en dokumentation om fysikaliska fenomen samt vilka spel dessa finns implementerade i. Dokumentationen inbegriper ljuseffekter, relativistiska fenomen, hållfasthetslära, gravitation och strömningsmekanik. För att uppfylla det andra huvudsyftet skulle en mer utförlig dokumentation behöva skapas. En komplett dokumentation skulle inbegripa alla fysikaliska fenomen som kan användas för *gameplay*, i vilka spel de finns implementerade i samt hur implementationerna skiljer sig från varandra.

Litteraturförteckning

[1] R. Dillon. *The Golden Age of Video Games: The Birth of a Multibillion Dollar Industry*. A K Peters/CRC Press, 2011.

[2] Owltom. Illustration bildschirm des videospiels "pong". Tillgänglig: `https://commons.wikimedia.org/wiki/File:Pong_bs.png`, 2004.

[3] Statista. Number of video gamers worldwide. Tillgänglig: `https://www.statista.com/statistics/293304/number-video-gamers/`, 2014. Online], åtkomst: 2017-02-01.

[4] A.Thorn. *Game Engine Design and Implementations*. Jones & Bartlett Learning, 2011.

[5] B. Nelson M. Martinez-Garza, D.Clark. Advances in assessment of students' intuitive understanding of physics through gameplay data. *International Journal of Gaming and Computer-Mediated Simulations*, 2013.

[6] Unity Technologies. Unity. Tillgänglig: `https://unity3d.com/`, 2005. Online], åtkomst: 2017-05-08.

[7] Unity Technologies. The leading global game industry software. Tillgänglig: `https://unity3d.com/public-relations`, 2016. Online], åtkomst: 2017-05-08.

[8] Unity Technologies. The leading global game industry software. Tillgänglig: `https://unity3d.com/learn`, 2016. Online], åtkomst: 2017-05-08.

[9] Dataspelsbranschen. Spelutvecklarindex 2015. Tillgänglig: `http://www.dataspelsbranschen.se/media/152177/spelutvecklarindex%202015.pdf`, 2015. Online], åtkomst: 2017-05-08.

[10] Valve Corporation. Steam. Tillgänglig: `http://store.steampowered.com/`, 2003. Online], åtkomst: 2017-05-08.

[11] Valve Corporation. steam store search", steam. `Tillgänglig:http://store.steampowered.com/search/`, 2017. Online], åtkomst: 2017-05-08.

[12] Unity Technologies. Build once, deploy anywhere, industry-leading multiplatform support. Tillgänglig: `https://unity3d.com/unity/multiplatform`, 2017. Online], åtkomst: 2017-05-09.

[13] G. Galileo och V. Viviani. *Le Opere*. Società Editrice Fiorentina, 1848.

[14] H. Lundh. *Grundläggande hållfasthetslära*. Studentlitteratur, 2016.

[15] C. Nordling och J. Österman. *Physics Handbook for Science and Engineering*. Studentlitteratur, 2014.

[16] J.L Meriam och L.G Kraige. *Engineering Mechanics*, volume 1 & 2. John Wiley & Sons, Inc, seventh edition, 2012.

[17] D.K Cheng. *Field and Wave Electromagnetics*. Pearson Education Limited, 2013.

[18] W. Rindler. *Relativity: Special General, and Cosmological*. Oxford University Press, second edition, 2006.

[19] Rovio Entertainment Ltd. *Angry Birds [iOS game]*. Chillingo Ltd., 2009.

[20] MIT Game Lab. *A Slower Speed of Light [PC/MAC game]*. MIT Game Lab, 2012.

[21] Vadim Ledyaev (PlayZilla). *Optika [PC/MAC game]*. Vadim Ledyaev, 2016.

[22] 2D Boy. *World of Goo [PC game]*. 2D Boy, 2008.

[23] Dry Cactus. *Poly Bridge [PC/MAC game]*. Dry Cactus, 2015.

[24] Rovio Entertainment Ltd. *Angry Birds Space*. Rovio Entertainment Ltd., 2012.

[25] S.A. de C.V. Monkey Squad. *Kerbal Space Program*. Monkey Squad, S.A. de C.V., 2015.

[26] Unity. Rigibody2d documentation. Tillgänglig: `https://docs.unity3d.com/ScriptReference/Rigidbody2D.html`. Online], åtkomst: 2017-05-08.

[27] Unity. Fixedupdate documentation. Tillgänglig: `https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html`. Online], åtkomst: 2017-05-11.

[28] Unity. Polygon collider2d documentation. Tillgänglig: `https://docs.unity3d.com/Manual/class-PolygonCollider2D.html`. Online], åtkomst: 2017-05-12.

[29] MIT Game Lab. Exploring the potential of play. Tillgänglig: `gamelab.mit.edu/about/`. Online], åtkomst: 2017-04-24.

[30] MIT Game Lab. A slower speed of light. Tillgänglig: `http://gamelab.mit.edu/games/a-slower-speed-of-light/`. Online], åtkomst: 2017-04-26.

[31] Andrew Groen. Nasa's jet propulsion lab is obsessed with a certain game, and i bet you can guess what it is. Tillgänglig: `https://web.archive.org/web/20131227041757/http://www.penny-arcade.com/report/article/nasas-jet-propulsion-lab-is-obsessed-with-a-certain-game-and-i-bet-you-can` 2013. Online], åtkomst: 2017-05-02.

[32] HarvesteR. Overhauled aerodynamics. Tillgänglig: `http://forum.`

kerbalspaceprogram.com/index.php?/developerarticles.html/overhauled-aerodynamics-r154/, 2015. [Online], åtkomst: 2017-05-02.

[33] Johnster2222. The reentry stage of zond 3 enters earths atmosphere. Tillgänglig: https://commons.wikimedia.org/wiki/File:Kerbal_Space_Program_5_3_2017_5_31_04_PM.png, 2017.

[34] V.Vasiliauskas. Developing agile project task and team management practices. Tillgänglig: www.eylean.com/Publications/DownloadPublication/3443705e-1697-4557-8327-ff8644fab40b?name=Whitepaper---Developing-agile-project-task-and-team-management-practices, 2014. Online], åtkomst: 2017-05-11.

[35] L. Torvalds. Git. Tillgänglig: https://git-scm.com/, 2005. Online], åtkomst: 2017-05-08.

A

Appendix

A.1 Utritande av banapproximationen vid iväg-skjutning

```
function DisplayForceApproximation(Vector2 Force)
{
    foreach(Dots[i])
    {
        if(i = 0)
            Dots[i].position = Force;
        else
            Dots[i].position = Dots[i-1].position +
               Dots[i-1].Force;

        foreach(Planet)
            ApplyForce(Dots[i]);
    }
}
```

A.2 Visualisering av fältriktning

Representationen av magnetfält i implementation är en blå rektangel som innehåller en mängd M punkter eller kryss. Mängden bestäms enligt formeln $M = B^2$ samt att både punkter och kryss placeras ut med mellanrum i x-led, dx, där

$$dx = \frac{width}{|B| - 1} \tag{A.1}$$

samt liknande i y-led för

$$dy = \frac{height}{|B| - 1}. \tag{A.2}$$

Den funktion som placerar ut alla punkter och kryss blir då

```
function PlacePoints()
{
    for (i < |B|)
```

```
        {
            for (j < |B|)
            {
                point[i, j].x = j * dx;
                point[i, j].y = i * dy;
            }
        }
    }
```

Den visuella delen för de elektriska fälten är liknande implementationen av magnetfältet. Det elektriska fältet är av rektangulär form med linjer som ligger i rektangelns y-led med mellanrum dy baserat på fältets styrka E samt en kompensering som kan fritt väljas vid design, k.

$$dy = \frac{height}{E + k - 1}. \tag{A.3}$$

De placeras ut enligt funktionen nedan

```
function PlaceLines()
{
    for (i < (E + k))
    {
        lines[i].y = i * dy;
    }
}
```