Gravitation i GameMakers 2D-spelmiljö: implementering och optimering

Alexander Nyberg

Examensarbete

Informationsteknik

2023

EXAMENSARBETE	
Yrkeshögskolan Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	9165
Författare:	Alexander Nyberg
Arbetets namn:	Gravitation i GameMakers 2D-spelmiljö: implementering och optimering
Handledare (Arcada):	Andrey Shcherbakov
Uppdragsgivare:	

Sammandrag:
Syftet med detta projekt är att utveckla samt undersöka implementation och optimering av två olika gravitationsmodeller i en 2D spelmiljö för att ge spelaren en möjligast rolig och engagerande upplevelse och att utforska vilkendera av dessa två spelaren föredrar. Genom att undersöka den underliggande fysiken av gravitation och lägga till den i en spelmiljö, är meningen att producera ett spel med egen gravitations motor som ger spelaren en unik och rolig spelupplevelse. Genom experimentering kommer detta projekt att utvärdera skillnaden mellan två olika implementationer av gravitation, en mera och en mindre verklighetstrogen.

Spelutveckling använder nuförtiden vanligtvis spelmotorer, en typ av mjukvaruutvecklingsmiljö som innehåller färdigbyggda komponenter och bibliotek för att utveckla spel. I detta projekt användes spelmotorn GameMaker för att utveckla två olika gravitationsmodeller, en verklighetsbaserad och en upplevelsebaserad. Den verklighetsbaserade modellen använder sig av Newtons formel för gravitation för att kalkylera kraften som objekten i spelet utsätts för. Kalkyleringarna görs i ett skript som räknar ut dragningskraften ett objekt har på alla andra objekt. Den upplevelsebaserade modeller däremot fungerar tvärtom, med enskilda funktioner på varje objekt. Dessa funktioner använder sig av fasta konstanter för att räkna ut dragningskraften andra objekt har på objektet med funktionen.

En testgrupp på 30 spelentusiaster användes för att undersöka vilken modell som ger en bättre spelupplevelse. Resultaten visade att den upplevelsebaserade modellen var mer engagerande än den verklighetsbaserade modellen. Slutsatserna var att projektets mål, att skapa ett 2D spel med självgjord gravitation, uppnåddes men att undersökningen kunde ha breddats för att ge mer tillförlitliga resultat.

Nyckelord:	Gravitation, Spelutveckling, Spelmotor, GameMaker, GameMaker Language, 2D
Sidantal:	24
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada University of Applied Sciences	
Degree Programme:	Information Technology
Identification number:	9165
Author:	Alexander Nyberg
Title:	Gravitation in GameMakers 2D game environment: implementation and optimization
Supervisor (Arcada):	Andrey Shcherbakov
Commissioned by:	

Abstract:
The purpose of this project is to develop and investigate the implementation and optimization of two different gravity models in a 2D game environment to provide the player with a fun and engaging experience, and to explore which of these two models the player prefers. By examining the underlying physics of gravity and adding it to a game environment, the goal is to produce a game with a custom gravity engine that provides a unique and fun gaming experience. Through experimentation, this project will evaluate the difference between two different implementations of gravity, one more realistic and one less realistic.

Game development commonly uses game engines, a type of software development environment that contains pre-built components and libraries for developing games. In this project, the game engine GameMaker was used to develop two different gravity models, one based on reality and one based on experience. The reality-based model uses Newton's formula for gravity to calculate the force that objects in the game are subjected to. The calculations are done in a script that calculates the gravitational force this object has on all other objects based on the object the script is on. The experience-based model, on the other hand, works the opposite way, with individual functions on each object. These functions use fixed constants to calculate the gravitational force other objects have on the object with the function.

A test group of 30 gaming enthusiasts was used to investigate which model provides a better gaming experience. The results showed that the experience-based model was more engaging than the reality-based model. The conclusions were that the project's goal, to create a 2D game with a self-made gravity engine, was achieved, but that the investigation could have been expanded to provide more reliable results.

Keywords:	Gravity, Game development, Game engine, GameMaker, GameMaker Language, 2D
Number of pages:	24
Language:	Swedish
Date of acceptance:	

INNEHÅLL

BILDER, FIGURER OCH KODEXEMPEL

1 INLEDNING

I den här delen av texten ges en kort inledning för vad arbetet handlar om och vad man kan förvänta sig få läsa om i kommande delar. Det ges en bakgrund till spelutveckling och gravitation samt syftet och målet med arbetet. Till sist en djupare analys av gravitation.

1.1 Bakgrund

Spelutveckling kan delas in i flera olika delar vilka tillsammans bidrar till produktionen av videospel, såsom programmering, grafik, ljud och speldesign. Under årens lopp har spelutveckling förändrats från simpla 2D spel till väldigt komplexa 3D spel med avancerad grafik och fysik. Gravitation är en fundamental kraft i naturen, som styr rörelsen av objekt med massa inom vårt universum. Därför spelar gravitation också en stor roll inom spelutveckling, speciellt i de spel som innehåller fysikbaserade mekanismer såsom simulationsspel, pusselspel och plattform spel. När det kommer till spelutveckling kan implementationen av gravitation ha en enorm påverkan på spelarens upplevelse. Gravitation i spel kan betyda ändringar i rörelser och funktioner av objekt inom spelmiljön, som skapar nya mekanismer och genomgång av spelet. Vilket i sin tur för med sig nya utmaningar och lager till spelet. För att kunna implementera gravitation i ett spel, måste utvecklarna ha en bra förståelse av fysikens lagar såsom massa, acceleration och kraft. De behöver också förstå hur spelmotorer fungerar och oftast kunna flera olika kodningsspråk när det kommer till att kunna utveckla egen gravitation som fungerar väl med spelmekanismerna.

Sammanfattat, är spelutveckling och gravitation två komplexa och relaterade ämnen som kräver djupt förstående för både fysik och datorteknik. Genom att kombinera dessa två ämnen kan spelutvecklare skapa spel som är uppslukande och ger spelarna en unik men rolig upplevelse, samtidigt som spelaren har möjligheten att lära sig hur världen fungerar.

1.2 Syfte, mål, forskningsfråga och problemformulering

Syftet med detta projekt är att utveckla samt undersöka implementation och optimering av två olika gravitationsmodeller i en 2D spelmiljö för att ge spelaren en möjligast rolig och engagerande upplevelse och att utforska vilkendera av dessa två spelaren föredrar.

Genom att undersöka den underliggande fysiken av gravitation och lägga till den i en spelmiljö, är meningen att producera ett spel med egen gravitations motor som ger spelaren en unik och rolig spelupplevelse. Genom experimentering kommer detta projekt att utvärdera skillnaden mellan två olika implementationer av gravitation, en mera och en mindre verklighetstrogen.

Målet med undersökningen är att framställa ett 2D spel som skall innehålla en självgjord gravitationsmotor. Genom genomförande, testning och iteration, av två olika typerna av implementation av gravitation skall jag få svar på vilkendera versionen av de två som är mera engagerande för spelaren. Versionerna kommer att testas av spelentusiaster. De kommer att ge de olika versionerna ett vitsord mellan ett till fem. Dessa vitsord kan sedan kvantifieras och användas för att få fram ett medeltal för varje version skilt. Spelmiljön bör innehålla åtminstone fyra (4) olika typer av objekt som skall påverkas av varandra, och som alla skall ha olik grad av dragningskraft.

Vilken typ av gravitationsmodell, verklighetstrogen eller förenklad, ger den mest engagerande och underhållande spelupplevelsen i en 2D-spelmiljö? Det här projektet siktar på att undersöka gravitation och de två olika genomförda modellerna för att utveckla en möjligast optimerad gravitationsmotor som för med sig en möjligast rolig spelupplevelse för spelaren. Hur rolig upplevelsen är kommer att baseras på testgruppens givna vitsord. Optimeringen är baserad på att skriva om koden så att möjligast få kalkyleringar görs för samma resultat. Detta görs för att minska på beräkningskraften datorn måste använda och på så vis göra spelet mera tillgängligt.

De existerande implementationer av gravitation i 2D spelmotorer tillåter inte skapande av unika skräddarsydda spelscenarion. För att lösa detta problem behöver spelutvecklaren ibland skiva en egen implementation av gravitationskraften. Implementation av gravitation i 2D spel för med sig tekniska och kreativa problem som måste överkommas för att kunna producera en övertygande spelupplevelse. Att implementera egen gravitation kräver av utvecklaren att denne besitter en grundläggande förståelse för hur gravitation fungerar i verkligheten samt kunskapen och förmågan att applicera denna kunskap i en spelmiljö. Utmaningen här kommer i formen av att kunna balansera gravitationens realism med den roliga och uppslukande upplevelsen spelare förväntar sig av spel.

1.3 Gravitation

Gravitation är en grundläggande naturlag som styr rörelsen av objekt med massa och är ett av de viktigaste koncepten inom fysik. Den är en kraft som existerar mellan vilka två som helst objekt i universum, oberoende av form, storlek eller komposition. Gravitationskraften mellan två objekt är direkt proportionell till deras massa och omvänt proportionell till deras distans från varann. (Khan Academy, 2023) Matematiska formeln för denna kraft, känd som Newtons lag om universell gravitation, skrivs så här:

$$F = G\frac{m_1 m_2}{r^2}, \qquad G = 6.67 \times 10^{-11}\left(\mathrm{N}\frac{\mathrm{m}^2}{\mathrm{kg}^2}\right)$$

Där *F* är gravitationskraften mellan objekten, m_1 och m_2 är massan av de två objekten, *r* är distansen mellan objektens mittpunkter och *G* är en konstant som bestämmer styrkan på gravitationskraften. Mera exakt är *G* en fundamental konstant inom fysiken som förekommer i formeln för gravitationskraft. Den är en proportionalitetskonstant som relaterar styrkan av gravitationskraften till massan av objekt och distansen mellan dem. I enklare termer, berättar *G* hur stark gravitationskraften är mellan två objekt, given deras massor och avståndet mellan dem. Det är ett väldigt litet tal, ungefär 6.67 x 10^{-11} N m²/kg², vilket betyder att gravitationskraften är en mycket svagare kraft jämfört med andra fundamentala krafter, som till exempel elektromagnetiska kraften. (Space, 2022)

1.4 Vad är en spelmotor?

Vid utveckling av spel används nuförtiden någon form av spelmotor. En spelmotor är en typ av mjukvaruutvecklingsmiljö som innehåller en uppsättning av verktyg för att utveckla spel. De innehåller vanligtvis färdigbyggda komponenter och bibliotek för grafiktolkning, fysiksimulation, inmatnings hantering, ljuduppspelning samt annan funktionalitet. Till exempel kan spelmotorer innehålla antingen en 2D- eller 3D-grafikåtervinningsmotor, ofta båda två, som är kompatibla med olika import format. En grafikåtervinningsmotor används för att visuellt framställa de olika modellerna i spelvärlden, såsom till exempel en asteroid. Genom att använda spelmotorer kan utvecklare effektivera spelutvecklingsprocessen och fokusera på att skapa funktioner på högre nivå såsom tillexempel spelmekanik och användargränssnitt. Spelmotorer är ofta gjorda för att vara flexibla och

anpassningsbara, vilket hjälper utvecklare att modifiera och förstärka spelmotorns inbyggda funktioner för att bättre passa de specifika målen utvecklaren har för sitt projekt. Spelmotorer kan därmed användas för att utveckla ett brett utbud av olika spelgenrer och stilar, från enkla 2D spel till komplexa 3D förstapersonskjutare. På grund av att spelutveckling har blivit mera komplext och resurskrävande med tiden, har spelmotorer blivit ett viktigt verktyg för spelutvecklare av alla storlekar, från indiestudior till storskaliga speltillverkare.

Förutom deras tekniska kapacitet, kan spelmotorer också ha stor inverkan på den artistiska och kreativa delen av spelutveckling. Många spelmotorer erbjuder kraftiga verktyg för att skapa och manipulera speltillgångar, såsom animationer, karaktärmodeller och texturer. Detta kan tillåta utvecklaren att skapa visuellt otroliga spel med detaljerade och uppslukande spelvärldar. Sammanfattat, har spelmotorer en betydande roll inom modern spelutveckling genom att erbjuda en kraftfull men flexibel plattform som utvecklare kan använda för att ge liv till sina kreativa idéer. Valet av spelmotor som skulle användas för detta projekt var baserat på skillnaderna emellan de tre mest kända spelmotorerna på marknaden som är fritt tillgängliga utan kostnad, Unity, Unreal Engine och GameMaker.

1.5 Unity

Unity är känd för dennes enkla användning och mångsidighet. Den används ofta för att producera spel över en mängd olika plattformar såsom mobil, bordsdator och konsol. Unity erbjuder ett brett val av verktyg och funktioner, såsom visuell skriftning och stöd för en mängd olika programmeringsspråk. Dock implementerar Unitys standardfysikmotor endast gravitation längs den vertikala axeln. Unity är också känd för sin starka gemenskap bland användarna och deras stöd. Därpå har Unity sin mycket täckande tillgångsbutik, som erbjuder tusentals resurser, både gratis och emot betalning, till användarna. (Unity, 2023)

1.6 Unreal Engine

Unreal Engine å andra sidan, är en kraftig spelmotor som är känd för sina avancerade funktioner och grafik möjligheter. Den används ofta för att producera komplexa och

visuellt gripande 3D spel för bordsdatorer och konsoler. Unreal Engine erbjuder ett omfattande set av verktyg och funktioner för spelutveckling, som inkluderar avancerad fysiksimulation och kinematografiska verktyg. Den är också känd för sina marknadsledande grafiska egenskaper som tillåter utvecklare att skapa extremt detaljerade och realistiska spelmiljöer och karaktärer. (Unreal Engine, 2023)

1.7 GameMaker

GameMaker däremot är en spelmotor som specifikt är gjort för endast 2D spel. Den är känd för sin enkla användning och tillgänglighet vilket gör den till ett populärt val bland nybörjare och indiespeltillverkare. GameMaker erbjuder också flera olika verktyg och funktioner, bland annat sitt dra och släpp gränssnitt. Därpå har GameMaker sitt enkla kodningsspråk GameMaker Language, förkortat GML, som är ett mycket nybörjarvänligt språk. GML har en välskriven dokumentation som låter användaren enkelt söka upp och ta reda på vad det finns för inbyggda funktioner i språket. (Yoyogames, 2023)

2 METODER

I metod delen av texten beskrivs spelutvecklingens och utforskningens metodologier. Sedan ges en förklaring på val av spelmotor för arbetet. Till sist ges definitioner på viktiga ord inom arbetet som kommer att hjälpa en förstå ämnen som uppkommer i senare delar.

2.1 Spelutvecklingsmetodologi

Målet med detta arbete är att framställa ett 2D-spel med två olika modeller av gravitation och därefter undersöka vilken av dem som engagerar spelaren mest. Metoden för utvecklingen av spelet kan delas in i en process med sju olika steg.

1. Definiera spelets koncept och mening
2. Val av spelmotor
3. Skapa spelgrafiken
4. Planera spelmekaniken

5. Implementera spelfunktionerna
6. Testa och felsök
7. Ge ut ny version och iterera

Efter att spelet har genomgått den första rundan av dessa steg har en ny version distribuerats till testgruppen. Efter deras tester och respons har processen återupptagits från steg fyra. Denna iterativa process har upprepats tills det önskade målet har uppnåtts.

2.2 Utforskningsmetodologi

Enligt Cohen, Manion och Morrison (2017) är kvantitativ undersökning numerisk representation och manipulation av observationer med ändamålet att beskriva och förklara de fenomen som dessa observationer speglar.

Utforskningen av vilken version som är mer engagerande för spelaren har genomförts med en kvantitativ metod. Denna metod valdes på basis av en del orsaker. Datan som samlas in är mätbar och kvantifierbar. Planerade storleken på testgruppen var tillräckligt stor för att samla in tillräckligt med datapunkter för att göra statistiska analyser. Genom att jämföra medelvärdet för varje version möjliggörs bedömningen av förändringarna för varje individuell version.

Under utvecklingsprocessen har en testgrupp bestående av 30 personer bedömt den övergripande känslan av gravitationen på en skala från 1 till 5. Spelet har sedan vidareutvecklats och en ny testomgång har genomförts. Spelversionerna har laddats upp och datan har samlats in via en gemensam plattform. Den kvantitativa datan som samlats in vid varje testomgång har använts för att beräkna varje versions medelvärde.

2.3 Val av spelmotor

Valet av spelmotor för projektet baserades på tre faktorer.

1. Enkel användning: Med tanke på mina kodningskunskaper inom spelutveckling var det viktigt att välja en spelmotor som låter mig koncentrera på kodningen av gravitation och gör resten så enkelt som möjligt. GameMaker är känt för sin

enkelhet vilket gör det till ett bra val för någon som jag som kodar på en av sina första spel.

2. 2D-specifika funktioner: Fastän det är fullt möjligt att med hjälp av Unity och Unreal Engine utveckla ett 2D spel, är GameMaker specifikt gjort endast för detta syfte. Det här betyder att den erbjuder ett stort sortiment av 2D specifika funktioner. Exempel på det här är GameMakers avancerade animations system, som låter användaren enkelt rita upp figurer och sedan animera dem, allt inom spelmotorns egna plattform.

3. Kostnad: Alla tre ovannämnda spelmotorer är gratis att använda, men vill man utnyttja deras fulla kapacitet måsta man betala och då är GameMaker ett billigare val än både Unity och Unreal Engine.

I detta fall där projektet innebär kodandet av ett 2D spel med egen gravitation, var GameMaker det bästa valet. Dessutom är projektets spel planerat att vara tillgänglig för bordsdatorer med framtida möjlighet att utvidga till mobil, vilka båda har utmärkt stöd på GameMaker. Därpå kan spelet, inifrån plattformen utan kostnad för utvecklaren, laddas upp till en nätsida där andra kan hitta och testa spelet på egen hand utan att behöva ladda ner eller betala något.

2.4 GameMakers definitioner på upprepande ord i texten

I denna del ges definitionen på ord som förekommer i texten en eller flera gånger som utanför GameMaker kan ha en annan definition. Meningen med denna ordlista är att underlätta läsandet genom att klargöra ordens betydelse för läsaren.

Objekt – Det finns flera definitioner på objekt inom olika IT-ämnen. Det här är en definition på vad ett objekt är inom spelmotorn GameMaker. I GameMaker har du objekt och instanser. Objekt utgör grundmallen för instanser, och är därför aldrig direkt närvarande i spelmiljön, endast instanser av objektet placeras i spelmiljön. Detta innebär att om du vill ändra något för alla instanser du kommer att skapa, kan du göra det genom att ändra objektet. (Yoyogames, 2023)

Instans - Inom GameMaker spelmotorn kan en instans ses som en specifik förekomst av ett objekt. Objekten fungerar som en grundmall vilken beskriver vilka egenskaper och beteenden en instans ska ha. En instans är alltså en konkret "kopia" av ett objekt som sedan kan placeras och användas i spelet på olika sätt, t.ex. genom att flyttas runt på skärmen, interagera med andra instanser eller utföra specifika uppgifter. Genom att arbeta med instanser kan man skapa en dynamisk och flexibel spelvärld där varje objekt kan ha flera olika instanser med unika egenskaper och beteenden. (Yoyogames, 2023)

Föräldraobjekt - Inom spelmotorn GameMaker kan objekt tilldelas en hierarkisk struktur där objekt kan fungera som "föräldrar" och "barn". Ett "barn" objekt ärver all kod och beteende från sin förälder medan ett föräldraobjekt inte ärver någon kod eller beteende från sina barn. Genom att använda sig av ett föräldraobjekt som en grund för flera andra objekt, kan man undvika att duplicera kod och därmed minska arbetsbelastningen. Om man sedan gör ändringar i föräldraobjektet, ändras också dessa funktioner i alla relaterade "barn" objekt. Samtidigt kan man fritt ändra på koden och beteendet i barnobjekten utan att det påverkar något annat objekt. (Yoyogames, 2023)

Skript - Skripttillgångar är i huvudsak en samling av en eller flera användardefinierade funktioner eller variabler som du själv skriver som kodavsnitt i Skriptredigeraren. Funktionerna du definierar i ett skript kan lösa uttryck, returnera värden eller göra något annat som kodningsspråket GameMaker Language tillåter. Skriptfunktioner bör i allmänhet användas om du har ett kodblock som du använder på mer än ett ställe eller objekt. (Yoyogames, 2023)

3 RESULTATREDOVISNING

I den här delen av texten ges en förklaring på hur projektet har påbörjats och hur de två olika gravitationsmodellerna har kodats. Efter det kommer resultatet att framföras samt en analys med egna tankar kring resultatet.

3.1 Basen till spelet

Processen av att utveckla 2D spel börjar oftast med samma steg. För att få igång utvecklingen av projektet, och få en bra bas att arbeta vidare på, baserades projektet på en

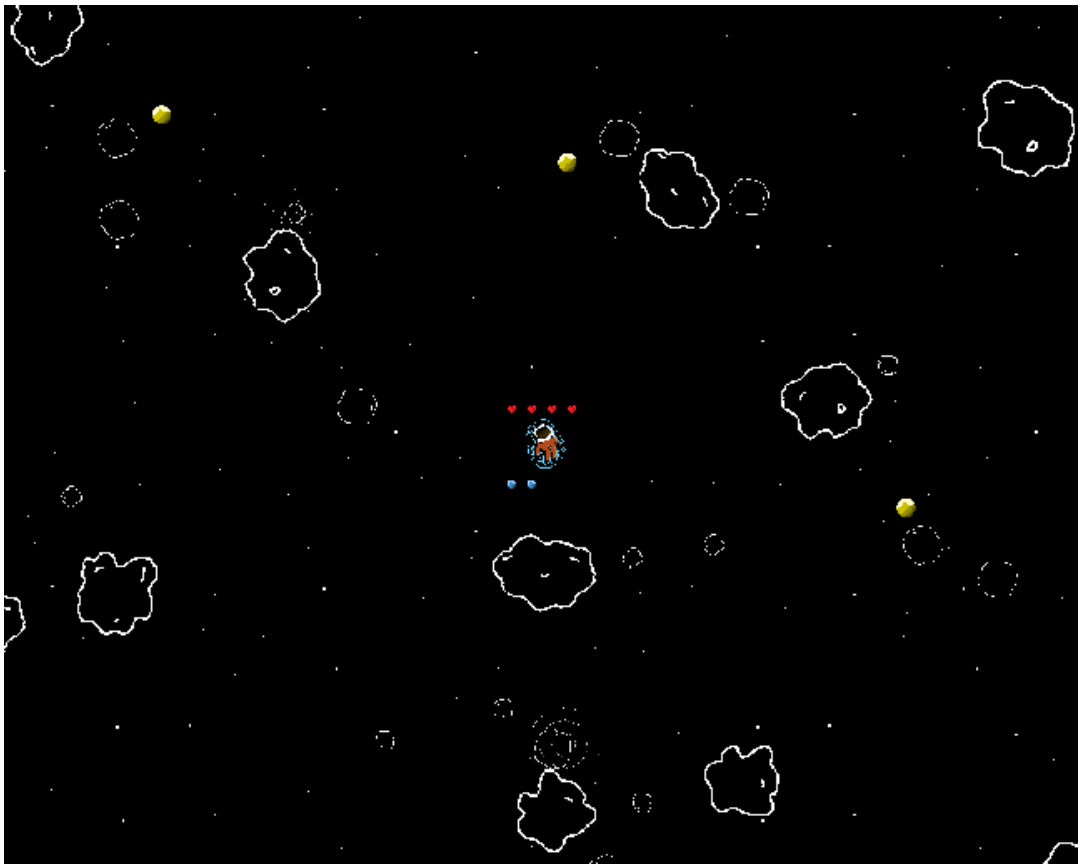
handledning som GameMaker gjort för ett spel som heter *Space Rocks*. Handledningar inom spelutveckling innebär en steg för steg genomgång av producerandet av ett specifikt spel. De innehåller exakta kodsnuttar och klara visualiseringar på hur man skall åstadkomma slutresultatet. Handledningar görs ofta till mycket enkla spel. *Space Rocks* har stora likheter till vad projektet ämnar utveckla och var på så sätt ett perfekt sätt att påbörja arbetet.

Handledningen började med att koda rörelsen för spelkaraktären. Ett skepp som skulle röra sig runt i rymden. Skeppet skulle kunna rotera till höger och vänster samt accelerera framåt. I rymden finns det inget luftmotstånd, vilket betyder att objekt i rymden inte byter hastighet i andra fall än om de kommer i kontakt med andra objekt eller gravitationskrafter. Så skeppet skulle röra sig framåt utan att stanna efter att man gett det fart. För att bromsa måste man bara rotera skeppet och accelerera i motsatt riktning. När spelkaraktärens rörelse var klar, lade man till några asteroider. Det lades till en liten, en mellan och en stor variant. Asteroiderna skulle komma till och sedan bli given en hastighet och riktning slumpmässigt. Efter det var nästa steg att koda vad som händer när spelkaraktären kolliderar med en asteroid, eller när de kolliderar med varandra. Träffar två asteroider varandra skall de gå sönder och bli mindre, medan spelaren skall förlora ett liv om karaktären träffar någonting. (GameMaker. 2021)

Sedan måste man få ett sätt för spelet att ha mening, annars finns det ingen poäng med spelandet. Till detta projekt lades det till fyra liv för spelaren, vilket betyder att om man krockade med andra objekt fyra gånger, dör karaktären. Om detta sker, förlorar spelaren. Sedan för att ge ytterligare mening och åter spelbarhet till spelet lades det också till ett poängsystem samt en butik med varor. För varje sekund spelaren hålls vid liv, får denne poäng. Poäng mängden man får per sekund, är baserad på mängden andra objekt i spelmiljön. Ju fler eller större objekt som finns desto mer poäng per sekund tilldelas spelaren. Till spelmiljön lades också till gula pengar, som spelaren efter att har förlorat kunde använda i butiken. Butikens varor ger spelaren diverse uppgraderingar till antingen hastighet, liv eller samlande av mera pengar. Till näst ökades spelbarheten ytterligare genom att spelvärlden gjordes större och takten på hur fort asteroiderna skapades in till rummet under spelets gång ökades. För att få större föremål att skapas i världen, kodades en funktion. Funktionen gav alla asteroider en massa, och gjorde så att när de kolliderar absorberar den med större massa den mindres massa. Om en asteroid blev tillräckligt tung, blev den

till en planet. Planeter fick fungera på samma vis, men när de blev tillräckligt tunga, skulle de bli till solar. Nu när det fanns en fungerande bas var man kunde röra sig runt bland en massa asteroider och förlora ifall man träffade dem fyra gånger, kunde utvecklingen av det egna gravitationssystemet påbörjas.

Skärmkap på spelkaraktären under en spel runda



3.2 Verklighetsbaserad gravitations modell

I den här delen av texten förekommer en del speljargong. För att underlätta läsandet finns det en ordlista längre upp i metod delen av texten. De ord som är viktigast att förstå, för att kunna följa med texten finns förklarade där.

För den mer verklighetstrogna modellen gjordes ett script, som nämndes *scr_gravity()* som kunde läggas på ett objekt, vilket betyder att alla instanser av detta objekt kommer att rulla scriptet. Först måste det kodas in vilka typer av objekt som instansen skulle tillfoga sin dragningskraft på. För att få ihop en sådan lista skapades ett föräldraobjekt under vilket det lades till alla objekt som skulle påverkas av gravitation. Efter det här

lades det till en del inbyggda funktioner för att kunna kalkylera nödvändig information. En inbyggd funktion i GameMaker är något som tar in parametrar och på basis av den ger ett resultat. Detta innebär att man inte själv behöver räkna ut till exempel distansen mellan två instanser i spelmiljön. Utan kan istället använda en inbyggd funktion.

Första inbyggda funktionen som användes var *collision_circle_list*. Den observerar alla objekt av en viss typ, som finns inom en viss radie från instansen som kör funktionen, och gör sedan en lista av dem. Den inbyggda funktionen tar in åtta olika parametrar. De olika parametrarna är positionen på x-axeln för mittenpunkten av cirkeln, positionen på y-axeln för mittenpunkten av cirkeln, cirkelns radie, objektet som skall upptäckas, noggrannheten av kalkylationen för kollisioner med cirkeln, om man skall upptäcka instansen som kör funktionen eller inte och till sist en tom lista och hur listan skall sorteras. För positionerna på x- och y-axeln användes instansens egna mittpunkt. Cirkelns radie räknades ut på basis av instansens vikt, ju större vikt, desto större radie. För objektet som skall upptäckas lades föräldraobjektet som nyligen skapades. För noggrannheten av kollisionerna, valdes det som kräver minst bearbetningskraft av datorn. Kalkyleringarna för gravitationen är redan så intensiva för datorn att upptäckande av objekt måste ske så smärtfritt som möjligt. Instansen själv skall inte vara med i listan så för den parametern skickades in *false*. För sorteringen valdes att listan skulle sorteras enligt vikt, från störst till minst. Valet för detta baserades på att man lättare märker om stora objekt inte påverkas av gravitation tillräckligt snabbt. Nu var skapandet av listan av instanser som skulle påverkas av gravitation konfigurerad.

Till näst skulle koden skrivas som utsätter varje enskild instans i listan till rätt mängd av gravitationskraft. Här användes Newtons formel för gravitationskraft. Till den behöver man massan av det första objektet, massan av andra objektet, en gravitationskonstant och distansen mellan objekten. Massan av de båda objekten finns lätt tillgängligt inom instansernas värden. Gravitationskonstanten lades till en början som ett. Efter det måste distansen mellan instanserna tas reda på. Det gjordes med den inbyggda GameMaker funktionen *point_distance*, som tar in två punkter i spelvärlden. Här användes båda instansernas x och y positioner. Sedan gjordes en ny variabel *pullForce* vars värde beräknades med Newtons formel.

Formeln är formulerad på följande sätt, summan av gravitationskraften gånger massan av ena instansen gånger massan av andra instansen dividerat med distansen mellan instanserna i kvadrat, vilket ser ut såhär $(G*m1*m2)/r^2$. En ytterligare variabel, nämnd *scaler*, introducerades för att konstgjort öka avståndet och ge en mer realistisk numerisk representation på distansen, $(G*m1*m2)/(r*s)^2$. Efter det måste det tas reda på i vilken riktning instansen skulle drabbas av dragningskraften. Här användes en inbyggd GameMaker funktion *point_direction*. Den tar också in två punkter i spelvärlden och ger tillbaka en riktning. Till sist användes inbyggda GameMaker funktionen *motion_add* för att lägga till rörelse på instansen. Den funktionen tar in en riktning och en kraft, här kunde nu användas de två värden som nyligen räknats ut. Koden för skriptet finns i textformat som bilaga, kodexempel 1.

3.3 Upplevelsebaserad gravitationsmodell

För den mindre verklighetstrogna modellen skulle det fokuseras mera på spelbarhet än verklighet. En sådan versions funktioner är enklare och innehåller mindre rörliga kalkyleringar. Detta är snabbare för datorer att bearbeta vilket ger spelet en smidigare känsla. Fastän en sådan modell är enklare i kodnings syfte, kräver den dock mera iterationer för att uppnå ett bättre resultat. På grund av att de flesta faktorer i en sådan kalkylering är konstanta, måste man genom test och iteration finna de värden på konstanterna som passar bäst. Den mera verklighetstrogna modellen tar inte i beaktande spelbarhet, utan bara bearbetningskraft och verkliga faktorer. Därför kan man använda sig av riktiga formler skalade upp med en skalnings faktor i dem. I den upplevelsebaserade modellen är funktionerna inte samma för olika objekt. Därför lades funktionerna till, rakt på varje olik storleks objekt skilt istället för att skapa ett gemensamt skript för alla objekt som i den första versionen.

Kodningen för denna version började med att lägga till funktionerna för att kalkylera dragningskraften som påverkar den lilla asteroiden. Funktionerna fungerar på följande sätt. De börjar med att kolla om en instans av ett vist objekt för tillfället finns i spelmiljön, till exempel en medelstor asteroid. Efter det tar den reda på vilken instans av detta objekt som är närmast. Detta sker med en inbyggd GameMaker funktion som heter *instance_nearest*. Sedan används den informationen för att ta reda på distansen till denna

instans. Det får man reda på med inbyggda funktionen *point_distance*. Efter det tas det reda på i vilken riktning instansen är med hjälp av inbyggda funktionen *point_direction*. Sedan, bara om instansen är under en viss längd ifrån, kör man inbyggda funktionen *motion_add* för att lägga kraft i en viss riktning på instansen som kör koden, till exempel en liten asteroid. Här användes inte en kalkylerad kraft på basis av en formel, utan en konstant. På så vis kommer nu den lilla asteroiden att dras med en konstant kraft emot den närmaste medelstora asteroiden. För att få detta att kännas så bra som möjligt, ändrar man på kraft konstantens värde och gör tester.

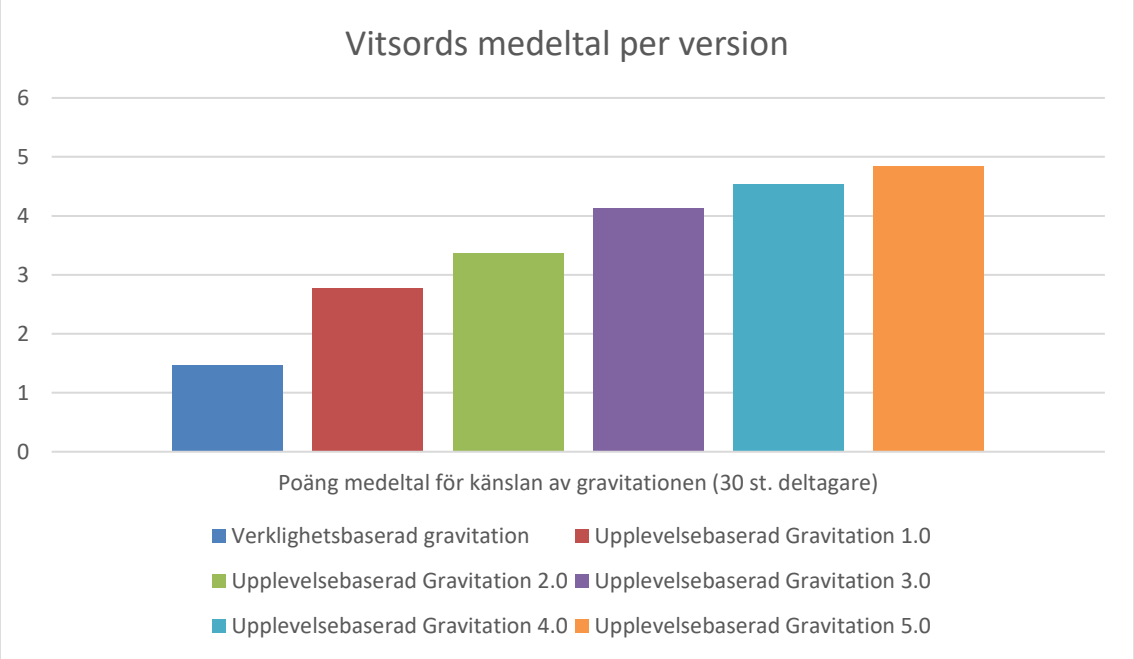
En likadan funktion lades till för varje objekt som kan finnas i spelmiljön. Konstanten som bestämmer kraften med vilken lilla asteroiden dras emot de olika objekten måste bestämmas och testas för varje objekt skilt. När alla objekt hade kod som kalkylerar dragningskraften till alla andra objekt, kunde testningen påbörjas för att finna de värden på konstanterna som ger det bästa möjliga slutresultatet.

3.4 Resultat

Målet med projektet var att utveckla ett 2D spel med självgjord gravitation. Detta uppnåddes genom två olika versioner av spelet, en upplevelsebaserad och en mera verklighetsbaserad gravitation. Spelen baserar sig på samma objekt, karaktär, spelmekanismer och grafik, skillnaden är endast gravitationen. Båda versionerna lägger till en kraft på objektens instanser i spelmiljön för att påverka deras rörelse, men på två olika sätt. Den mera verklighetstrogna versionen använder sig av ett ständigt förändrande värde för kraften som påverkar instanserna. Detta värde beräknas genom användningen av Newtons lag om universell gravitation. Medan den andra, upplevelsebaserade metoden av gravitation använder sig av en fast konstant för kraften som påverkar instansernas rörelse. Resultatet blev två fysikmässigt helt olika spel, båda med fungerande självgjord gravitation.

Det skulle också utforskas vilken av de två metoderna, verklighetstrogen eller upplevelsebaserad gravitation, som skulle ge spelaren en mer tillfredsställande spelupplevelse. För att utforska detta genomfördes en experimentell undersökning där 30 stycken testpersoner spelade spelet med en version av den upplevelsebaserade gravitationen. Sedan poängsatte testarna gravitationens känsla jämfört med verklighetstrogna versionen av gravitation samt tidigare versioner av upplevelsebaserade gravitationen. Undersökningen innefattade

fem olika versioner av den upplevelsebaserade gravitationen, varav varje version byggde på feedbacken från föregående versioner. Resultaten av undersökningen visade att samtliga versioner av upplevelsebaserad gravitation fick högre betyg än den verklighetstrogna versionen. Detta antyder att spelarna föredrar den upplevelsebaserade approachen i det här sammanhanget. Intressant nog slog varje ny version av upplevelsebaserad gravitation även de tidigare versionerna, vilket indikerar att en iterativ process för att förbättra spelupplevelsen genom en upplevelsebaserad approach kan vara effektiv. Denna studie bidrar därmed till förståelsen av hur spelupplevelsen kan påverkas av olika tillvägagångssätt inom spelutveckling, och ger insikter i hur iterativa förbättringar kan användas för att skapa en mer tillfredsställande spelupplevelse. Vidare kan denna studie också bidra till att öka medvetenheten kring vikten av att ta hänsyn till användarens upplevelse och preferenser i spelutvecklingsprocessen.



Medeltalsvitsord för varje enskild gravitationsmodell/version

3.5 Analys

Metodologin för hur undersökningen gjordes var mycket enkel. Testgruppen bestod av en grupp aktiva spelentusiaster. Gruppen samlades in via en plattform som används främst för kommunikation mellan spelare, som heter Discord. På Discord kan personer ansluta sig till servrar där de kan kommunicera med varandra genom text eller röst kanaler. Dessa servrar bildar ofta små ”online samhällen” där likasinnade personer kan hållas i kontakt i samband med spel. Eftersom hela testgruppen bestod av personer ifrån en av dessa servrar där jag själv är aktiv, kommer resultaten av testerna ofta att spegla mina egna åsikter. På grund av detta är min undersökning klart partisk. Även så tror jag att resultaten skulle har blivit mycket liknande också i andra testgrupper. Något som jag har lagt märke till i spel, är att strävan efter realism inte alltid ger den roligaste spelupplevelsen. Därför ville jag inte bara implementera en typ av gravitation. Redan medan jag utvecklade spelet var det klart vilken version som aktiva spel entusiaster kommer att lägga sin röst på. Med facit i hand märkte jag att undersökningen känns mera som en undersökning av vilken version av upplevelsebaserade gravitationen som är bäst. Och eftersom jag baserade varje ny version av gravitationen på testgruppens respons, var det klart att den nyaste versionen kommer att vara bättre än sin föregångare. Därför är inte resultaten enligt mig överraskande utan snarare förväntade.

3.6 Spelet

De två versionerna av spelet är tillgängliga på Gx Games. För att kunna testa spelen måste man använda *Opera Gx* webbläsaren. Båda spelen fungerar spelmässigt på samma sätt, det är bara gravitationen som är olik.

Spelet med upplevelsebaserad gravitation.
https://gx.games/games/q95jz6/gravity-2/

Spelet med verklighetsbaserad gravitation.
https://gx.games/games/3esntj/gravity-more-realistic-gravity-version-/

4 DISKUSSION

I den här delen av texten ges mina slutliga tankar om kodningsprocessen och resultatet av utforskningen i fri form. Och till sist ges mina slutsatser angående projektet i sin helhet.

4.1 Kodningsprocessen

Utveckling av spel är en ny kunskap för mig, och på så vis har jag under denna process lärt mig en hel del som kan hjälpa mig i kommande projekt. För att komma igång med arbetet använde jag mig av en färdig bas i form av en handledning för hur man kodar ett liknande spel. Utan detta skulle utvecklandet av själva spelet tagit mig mycket längre tid. Handledningen lärde mig hur jag bättre kunde använda mig av GameMakers inbyggda funktioner och visade hur man lätt kunde navigera GameMaker för att snabbare hitta vad man söker. Med kunskaperna jag lärde mig, kunde jag efter handledningen enklare implementera nya egna idéer. Att använda sig av andras spel/kod som bas för ens egna projekt är något jag varmt kan rekommendera och definitivt kommer att använda mig av i framtiden. Ju mer spel man utvecklar desto fler baser har man dessutom att arbeta med när man vill göra något nytt.

När det kommer till implementationen av gravitation kunde jag ha gjort saker i en bättre ordning. En bra ordning för implementationen kunde vara, forskning om gravitation, letande efter spel som använder sig av egen gravitation, att lära sig mera om de inbyggda funktionerna och verktygen i GameMaker och till sist implementation av egen kod. Men efter att jag hade spelets bas klar förivrade jag mig och antog rätt kaxigt att jag bara kunde kasta in formeln för Newtons lag om universell gravitation på några objekt, och så är allt klart. Men på grund av mina brister i kunskap om hur GameMaker fungerar gjorde jag en hel del misstag som jag senare måste åtgärda. Till exempel hade jag till en början lagt min gravitations funktion rakt på varje objekt skilt. Detta innebar att varje ändring jag gjorde måste göras om på 4 andra ställen. Efter att jag lärde mig hur man gör skript i GameMaker behövde jag bara göra ändringarna en gång och på ett ställe. Detta skulle ha försnabbat min kodning i början en hel del. Andra liknande misstag jag gjorde kostade mig mycket tid som jag kunde ha använt smartare genom att först utforska mera om vad GameMaker erbjuder för inbyggda funktioner och verktyg.

För implementeringen av formeln från Newtons lag om universell gravitation borde jag likaså lärt mig mera förrän jag påbörjade kodningen. Till exempel förstod jag inte vad konstanten G gjorde, och hade alldeles för höga värden på den i början. Efter att jag läst mera om gravitation och de olika delarna som bygger upp formeln, kunde jag bättre välja

massorna av mina objekt och värdena på G samt min *scaler* variabel. Den kunskapen jag lärde mig kunde också ha sparat tid för mig i tidigare skeden.

Slutligen underestimerade jag mängden tid som går in i att planera, rita, designa och koda alla andra delar av spelet. Kodningen av gravitation var mest baserad på kunskap om gravitation och matematik, saker jag tycker är intressanta och anser mig kunnig i. Däremot kunde jag till exempel ingenting om hur man gör startmenyer och hur man får kameran att fungera som man vill inom spelmiljön. Det gick minst lika mycket tid till att implementera sådana delar till projektet som det gick till att implementera gravitationen. Kodningen som tillåter spelaren att spara sin framgång och fortsätta var man lämnat spelet senast är ett bra exempel. Det var något jag mot slutet av spelet ville implementera, och antog att det går lätt. Men den implementationen tog minst en vecka. Alla sådana brister i min kunskap som jag inte visste att jag inte hade, var saker som jag borde ha utforskat mera på förhand.

Själva processen av att utveckla spelet anser jag dock att var mycket roligt. Jag lät min fantasi flöda och implementerade en hel del extra saker för att få spelet att ge en bättre upplevelse. För varje version av den upplevelsebaserade gravitationen, lade jag också till en hel del nya saker som gjorde spelet roligare i sin helhet. Till exempel lade jag till stjärnor i bakgrunden av spelmiljön för att ge mera liv och bättre kunna se hur asteroiderna rör sig. Jag lade också till en butik med varor man kunde köpa för att göra spelkaraktären snabbare eller mera skyddad. Sammanfattat är jag mycket nöjd med spelet jag lyckades producera och anser att jag nått mitt mål för arbetet.

4.2 Tankar och reflektioner om arbetets resultat

Enligt mitt mål för projektet är det sammanfattade resultatet att jag lyckats producera ett 2D spel, i två versioner, som innehåller självgjord gravitation helt enligt målets krav och utöver det undersöka vilkendera av versionerna spelarna föredrar. Det är klart att båda versionerna av spelet har fungerande gravitation och att de fungerar på olika sätt. Spelen innehåller minst sex objekt som påverkas av gravitationen vilket också är i enlighet med kravens minst 4 objekt. Bristerna här är att den verklighetsbaserade versionens kalkyleringar tog så pass mycket mera bearbetningskraft av min dator att jag inte kunde hantera en tillräckligt stor mängd objekt inom spelmiljön. Skalan kunde inte heller vara tillräckligt

stor för att korrekt kunna simulera gravitationen utan en bättre dator. Därpå utvecklade jag spelet med upplevelsen konstant som baktanke, vilket betydde att grunden för spelet var bättre passande för upplevelsebaserade versionen. Dessutom kom mina artistiska kunskaper emot när det kommer till att skapa grafiken för objekten, vilket ytterligare dämpar verklighets känslan av spelet. Med mera tid och kunskap om spelutveckling kunde jag vidareutveckla spelet och producera något som bättre blandar ihop verklighet med upplevelse. Resultatet för spelutvecklingen är dock enligt mig nått, baserat på att jag lyckats producera egen gravitation i en 2D spelmiljö.

Jag skulle också ta reda på vilkendera versionen är mer engagerande för spelaren. Resultatet var att den upplevelsebaserade versionen var klart bättre än den andra enligt testgruppens tycke. Testgruppens uppsättning kunde dock ha ändrat på resultatet men högst antagligen inte drastiskt i detta fall. Den verklighetsbaserade versionen är klart sämre på grund av brister i mitt kunnande inom gravitation samt att båda versionerna använder sig av en bas som bättre passar upplevelsebaserade modellen. Detta var mitt första försök på att implementera egen gravitation i ett spel. Och att hoppa rakt in i djupa endan och försöka utveckla en verklighetstrogen gravitation är oftast inte så lätt. Undersökningen som gjordes för att finna svaret till vilkendera versionen som enligt spelaren är bättre, kunde också har varit bredare och längre. Bristerna här var att min kapacitet för att nå testare inte var så stor, och därmed var testgruppen relativt liten och bestod av för det mesta likasinnade personer. Dessutom kunde jag ha gjort flera iterationer på den verklighetsbaserade gravitationen på samma vis som till den andra modellen. Jag använde mig av en *scaler* konstant, som jag kunde har ändrat på för att göra flera versioner. Med flera versioner av båda två gravitations modellerna, kunde undersökningen ha gjorts på nya versioner av båda modellerna vid varje test. På så vis kunde jag har utvecklat båda versionerna sida vid sida, och basera ändringarna på responsen jag skulle har fått vid varje testskede. Det kunde ha lätt till att undersökningen möjligen fått andra resultat. Dock anser jag att fastän resultatet kan vara partiskt, är det i enlighet med vad jag förväntade mig att få för svar.

4.3 Slutsatser

Under arbetets gång har min kunskap om både gravitation och spelutveckling blivit bättre. Jag har producerat ett fullständigt 2D spel, med egen gravitation, olika spelrum, en butik, möjligheten att spara sin framgång samt mekanismer för att samla ihop poäng. Spelet i sin helhet har en del brister när det kommer till vad ett spel oftast innehåller för basfunktioner. Men, båda två gravitationsmodellerna fungerar och spelet har en början och ett slut. Det betyder enligt mig att jag har lyckats slutföra det som var planerat enligt mitt mål. Dessutom har jag genom respons från en testgrupp på 30 personer, också lyckats formulera ett svar på vilkendera av mina gravitationsmodeller som ger bättre spelupplevelse. På så vis har jag kommit fram till dessa två slutsatser:

1. Målet för projektet, att skapa ett 2D spel med självgjord gravitation har uppnåtts, även om det fanns vissa brister och begränsningar i implementeringen.
2. Enligt testgruppens feedback var den upplevelsebaserade versionen av spelet mer engagerande än den verklighetsbaserade versionen, men undersökningen kunde ha gjorts mer omfattande och breddats för att ge mer tillförlitliga resultat.

KÄLLOR

Cohen, L. & Manion, L. & Morrison, K. (2017). Research Methods in Education (8th edition.). Taylor & Francis

GameMaker. (2023). https://GameMaker.io/en

GameMaker. (2021). *Space Rocks*. https://GameMaker.io/en/tutorials/space-rocks-gml

Khan Academy. (2023). *Introduction to gravity.* https://www.khanacademy.org/science/hs-physics/x215e29cb31244fa1:types-of-interactions/x215e29cb31244fa1:newton-s-law-of-universal-gravitation/v/introduction-to-gravity

Space. (2022). *What is the gravitational constant?* https://www.space.com/what-is-the-gravitational-constant#:~:text=The%20gravitational%20pull%20between%20two%20objects%20can%20be%20calculated%20with,of%20the%20distance%20between%20the

Unity. (2023). *Unity documentation.* https://docs.unity.com/

Unreal Engine. (2023). *Unreal Engine documentation.* https://docs.unrealengine.com/5.1/en-US/

Yoyogames. (2023). *GameMaker manual.* https://manual.yoyogames.com/

BILAGOR

Kodexempel 1. Verklighetsbaserad gravitations skript

```
// Gravity script
function scr_gravity(){

	//Variables to create list of objects to inflict gravity upon
	x1 = x;
	y1 = y;
	rad = mass/300;
	obj = obj_gravityParent;
	prec = false;
	notme = false;
	list = ds_list_create();
	ordered = true;
	collisions = collision_circle_list(x1, y1, rad, obj, prec, notme, list,
	ordered);

	if(collisions > 0){
	  for(var i = 0; i < collisions; ++i;){
	       with(list[| i]){
	           dist = point_distance(x,y,other.x,other.y);
	           dir = point_direction(x,y,other.x,other.y);

	           // Calculate the pulling force (gravity) towards the other
	           object
	           if(dist < 400){dist = 400}
	           G = 0.1;
	           scaler = 100;
	           pullForce = (G*mass*other.mass)/sqr(dist*scaler);
	           show_debug_message(pullForce)
	           if(pullForce > 1){pullForce=0.11}
	           motion_add(dir,pullForce);
	       }
	  }
	}
	ds_list_destroy(list);
}
```

Kodexempel 2. Upplevelsebaserade gravitations funktioner, liten asteroid.

```
if(instance_exists(obj_asteroid_med)){
	astMed = instance_nearest(x,y,obj_asteroid_med);
	astMedDist = point_distance(x,y,astMed.x,astMed.y);
	astMedDir = point_direction(x,y,astMed.x,astMed.y);

	if(astMedDist < 300){ motion_add(astMedDir,0.005); }
}

if(instance_exists(obj_asteroid_big)){
	astBig = instance_nearest(x,y,obj_asteroid_big);
	astDist = point_distance(x,y,astBig.x,astBig.y);
	astDir = point_direction(x,y,astBig.x,astBig.y);

	if(astDist < 700){ motion_add(astDir,0.01); }
	if(astDist < 400){ motion_add(astDir,0.015); }
}

if(instance_exists(obj_planet)){
	planet = instance_nearest(x,y,obj_planet);
	planetDist = point_distance(x,y,planet.x,planet.y);
	planetDir = point_direction(x,y,planet.x,planet.y);

	if(planetDist < 1000){ motion_add(planetDir,0.01); }
	if(planetDist < 700){ motion_add(planetDir,0.015); }
	if(planetDist < 400){ motion_add(planetDir,0.02); }
}

if(instance_exists(sun)){
	sunDist = point_distance(x,y,sun.x,sun.y);
	sunDir = point_direction(x,y,sun.x,sun.y);

	if(sunDist < 1000){ motion_add(sunDir, 0.04); }
	if(sunDist < 700){ motion_add(sunDir,0.05); }
	if(sunDist < 500){ motion_add(sunDir,0.1); }
	if(sunDist > 1000){ motion_add(sunDir,0.01); }
}
```

Anonymiserad Excel data för varje enskild version, poängskala 1–5.

Deltagare	Verklighetsbaserad	U Version 1	U Version 2	U Version 3	U Version 4	U Version 5
1	2	3	4	4	5	5
2	1	2	3	4	5	5
3	2	3	3	4	5	5
4	2	3	4	4	4	5
5	2	3	4	5	5	5
6	1	3	4	5	5	5
7	1	3	3	4	4	4
8	1	3	4	4	4	5
9	1	3	3	4	5	5
10	2	3	3	4	4	5
11	2	3	3	5	5	5
12	2	3	3	3	4	5
13	1	3	4	5	5	5
14	2	3	3	4	4	4
15	2	3	3	4	4	5
16	2	3	3	4	5	5
17	1	3	4	5	5	5
18	1	3	4	4	5	5
19	2	3	4	5	5	5
20	1	2	3	5	5	5
21	1	2	3	4	4	4
22	1	2	3	4	5	5
23	1	2	3	4	5	5
24	1	2	3	3	4	4
25	2	3	3	3	4	5
26	2	3	4	4	5	5
27	1	3	3	4	4	4
28	1	3	4	4	4	5
29	2	3	4	4	4	5
30	1	2	2	4	4	5
Medeltal	1,47	2,77	3,37	4,13	4,53	4,83