

Planetsimulator

Johannes Olegård

Anton Olsson

14 maj 2013

1 Programbeskrivning

1.1 A

Vi ska skriva ett program som simulerar ett 2D-universum. Användaren kan skapa planeter i olika storlekar, massor, färger, hastigheter, accelerationer, etc. och planeter kommer sedan interagera med varandra (t.ex. skapa omloppsbanor).

1.2 B

Vi har skrivit ett program som simulerar ett 2D-universum. Programmet kan ses som ett s.k. sandbox-spel. Användaren kan skapa planeter i olika storlekar, massor, färger, och hastigheter. och planeter interagera med varandra med gravitation (acceleration) och kollisioner.

Projektet finns på github: <https://github.com/jesajx/inda12-proj>

1.3 C

Programiden ändrades inte ändrats så mycket, men har till viss del byggts vidare på (t.ex. med kollisioner och ljud).

2 Användarbeskrivning

2.1 A

Applikationens användargrupp är seende mellan ca 13 och 90 år med åtminstone lite datorvana som är intresserade av att experimentera med fysik. Användaren förutsätts ha viss kunskap i relevanta fysikaliska begrepp (t.ex. vet ungefär vad massa och radie är) då programmet ej förklarar dessa. Vidare förutsätts användaren förstå engelska då text i programmet är på engelska.

Programmet skulle exempelvis kunna användas av högstadiesbarn för att med experiment lära sig hur gravitation, etc. påverkas av massa, hastighet, etc.

2.2 B

Applikationens användargrupp är egentligen vem som än vill experimentera med fysik. Användaren behöver ha minimala kunskaper i fysiska begrepp (främst massa och radie) och förstå engelska.

Applikationen använder skärm, mus och tangentbord, så användaren behöver förstås kunna se och ha åtminstone en fungerande arm. Programmet spelar upp ljud vid vissa handlingar, men hörsel är inget måste.

Programmet skulle exempelvis kunna användas av högstadiesbarn för att med experiment lära sig hur gravitation, etc. påverkas av massa, hastighet, etc.

2.3 C

Inte så mycket har ändrats över uppfattningen av användargruppen. Eventuellt att färre fysik kunskaper troligtvis krävs och att användarna är tänkta att experimentera sig fram.

3 Användarscenarier

3.1 A

3.1.1 Scenario 1

Användaren är en datalog på kth som går första året. Fysikkunskaperna är goda. Användaren vill simulera ett solsystem med 3 planeter i omloppsbana runt en stjärna.

Programmet startas med ett fönster. Vänstra delen av skärmen består av en meny. Längst ner i mitten av skärmen är en tickande klocka (en visare men inga siffror). Resten av skärmen är svart.

I menyn finns parametrar som t.ex. massa och radie att ställa in. Användaren skriver in stjärnans egenskaper och klickar sedan i mitten av skärmen där det då skapas en ifylld cirkel.

Användaren ställer in nästa planet och håller inne mus-knappen där planeten ska skapas och drar sedan pekaren åt det hållet initial hastighet ska vara. Medans användaren drar med musen så syns en pil från planeten till pekaren med text brevid om hur hög hastigheten blir när användaren väl släpper.

En streckad ellips visar hur omloppsbanan kommer bli. När användaren släpper så börjar den nyskapade planeten röra sig i en ellips runt stjärnan. Användaren repeterar mönstret för 2–3 planeter men märker denna gång att när hen håller på att placera (musknapp nertryckt och vektorn för hastighet synes) en ny planet så pausas simulationen för att underlätta placeringen.

3.1.2 Scenario 2

Användaren har viss erfarenhet av datorer (bl.a. spel) och fysikkunskaperna är minimala. Programmet startar likt beskrivet i Scenario 1. Även denna användare ska skapa ett solsystem med en stjärna och några planter i omloppsbana runt den.

Användaren testar att klicka ut lite planeter utan att ändra några parametrar. De rör sig inte så användaren försöker dra i en av dem. Planeterna och klockan nere i mitten pausas samtidigt, planeten markeras och en hastighetspil visas från den till muspekaren. Användaren släpper musknappen och planeten får fart - ut från skärmen. Användaren försöker få skärmen att följa efter och upptäcker att det går att pausa med space-knappen och att zooma med mushjulet.

Planten ges nu lite fart tillbaka - lite lägre denna gång.

Användaren provar nu att ändra parameterarna på planeten. Ökad storlek verkar inte ge gravitation, men massa verkar göra det.

Användaren ökar massan rejält planeten och alla andra planter börjar röra sig snabbare och snabbare mot den.

Vissa planeter missar den stora planeten; då får större hastighet när de kommer nära och slungas på så sätt förbi. Andra planeter åker rakt in i planeten och så att de smälter samman. Användaren noterar att massan hos den stora (fortfarande markerade) planeten ökar i menyn till vänster.

Användaren stannar den stora planeten genom att sätta dess hastighet till noll i menyn till vänster och börjar sätta ut planeter en sträcka ifrån den - med olika hastigheter, åt olika håll och med låga massor.

Efter några försök har användaren lyckats få några planeter att åka i en elliptisk bana runt den stor planeten.

3.2 B

3.2.1 Scenario 1.1

Användaren är en datalog på kth som går första året. Fysikkunskaperna är goda. Användaren vill simulera ett solsystem med 3 planeter i omloppsbana runt en stjärna.

Programmet startas med ett fönster. Vänstra delen av skärmen består av en grå meny. Resten av skärmen är svart, bortsett från lite vit text.

I menyn finns parametrar som t.ex. massa och radie att ställa in. Användaren skriver in stjärnans egenskaper och klickar sedan i mitten av skärmen där det då skapas en ifylld cirkel.

Användaren ställer in och placerar nästa planet. Användaren klickar och drar sedan i planeten med höger musknapp åt det hållet initial hastighet ska vara. Medans användaren drar med musen så syns en pil från planeten till pekaren med text bredvid om hur hög hastigheten blir när användaren väl släpper.

En streckad ellips visar hur omloppsbanan kommer bli. När användaren släpper så börjar den nyskapade planeten röra sig i en ellips runt stjärnan. Användaren repeterar mönstret för 2-3 planeter men märker denna gång att när hen håller på att placera (musknapp nertryckt och vektorn för hastighet synes) en ny planet så pausas simulationen för att underlätta placeringen. Alla planeter börjar inte gå i omloppsbana som planterat, men med några försök lyckas användaren.

3.2.2 Scenario 2.1

Användaren har viss erfarenhet av datorer (bl.a. spel) och fysikkunskaperna är minimala. Programmet startar likt beskrivet i Scenario 1.1. Även denna användare ska skapa ett solsystem med en stjärna och några planter i omloppsbana runt den.

Användaren testar att klicka ut lite planeter utan att ändra några parametrar. De rör sig inte så användaren försöker klicka och dra i en av dem. Ett snabbt vänsterklick på planeten markerar den och ändrar i menyn till vänster.

Drag med vänster musknapp flyttar planetens position. En cirkel visar hur planeten hamnar efter flyttningen och en linje dras till den från planetens nuvarande position. Vidare visas den nya positionens koordinater bredvid cirkeln. Efter en planet flyttats börjar den däremot inte röra sig.

Drag med höger musknapp ger däremot en hastigheten i motsvarande riktning. Alla planeter pausas medans användaren drar. Från den högerklicks-dragna planeten visas en pil till muspekaren och bredvid visas en siffra. Ju längre pilen är, ju större blir siffran - alltså måste det vara hastigheten. Användaren släpper höger musknappen och planeten får fart - ut från skärmen. Användaren försöker få skärmen att följa efter och upptäcker att det går att pausa med space-knappen, att zooma med mushjulet och att förflytta skärmen med piltangenterna.

Planeten ges nu på samma sätt lite fart tillbaka - lite lägre denna gång. Användaren måste dra flera gånger för att få planeten att åka åt rätt håll - högerklicks-drag lägger till hastighet, snarare än ställer in en hastigheten.

Användaren provar nu att markera planeten och ändra dessa parameterar i menyn till vänster. Ökad radie verkar inte öka gravitationen, men massa verkar göra det.

Användaren ökar planetens massan rejält och alla andra planter börjar röra sig snabbare och snabbare mot den. Användaren märker att en linje ritas från den markerade planeten i den riktning den har rör sig.

Vissa planeter missar den stora planeten; de får större hastighet när de kommer nära och slungas på så sätt in bakom planeten och iväg. Andra planeter åker rakt in i planeten och med ett pling-ljud studsar mot den och flyger iväg.

Användaren stannar den stora planeten genom att i menyn sätta dess hastighet till noll. Användaren testar därefter metodiskt att sätta ut planeter på olika avstånd och hastigheter från den tunga planeten. Planeterna sätts ut med låga massor.

Användaren märker att linjen som ritas från markerade planter i rörelse ibland böjer sig - den visar alltså hur planten kommer att röra sig i framtiden. Med hjälp av detta lyckas användaren efter några försök få några planeter att åka i elliptiska banor runt den tunga planeten.

3.2.3 Scenario 3

Scenariot börjar likt de två ovan. Användaren börjar däremot genom att klicka på "templates" i vänstermenyn. Hen får då upp ett fönster med en lista på olika förbyggda universum. Höger halva av fönstret finns ett fält med en beskrivning av markerad template. Längst ner på skärmen finns två knappar: "cancel" och "load". Användaren väljer i listan passande system för uppgiften (enkelt solsystem) och klickar på load. På skärmen skapas ett fungerande solsystem. Användaren zoomar ut med mushjulet för att få en bättre blick av systemet.

Användaren provar ladda ett nytt solsystem, dubbelstjärnigt denna gång. När användaren klickar på load tas föregående solsystem bort och ersätts med det dubbelstjärniga.

3.3 C

Inte så mycket ändrades från original-scenariorna. Främst har små ändringar i hur programmet ser ut och betéer sig lagts till.

4 Testplan

4.1 A

Vi testar programmet framförallt genom användartestning.

Användartestningen ska ske genom att ta tag i lämplig person i godtycklig kth korridor och få denne att utföra en eller flera av följande uppgifter:

1. Skapa ett solsystem med en planet och en tillhörande måne.
2. Justera omloppsbanan på en yttre planet i ett existerande solsystem med 2 orbitaler så att planeten hamnar innanför den andra.
3. Skapa ett dubbelstjärnigt solsystem eventuellt med en tillhörande planet.

4. Skapa en lite galax. Många stjärnor i mitten och fler hela solsystem som kretsar runt dessa.

De två senare uppgifterna är lite svårare och vi räknare inte med att användarna klarar dessa. Däremot vill vi gärna se hur de försöker använda programmet för att lösa problemen.

Innan användaren börjar ska vi kort förklara med ord (utan att visa) vad vi vill att hen ska försöka utföra. Vi ska också tydligt förklara att eventuella misslyckanden beror på brister i det grafiska gränssnittet och inte användarens bristande intelligens. Vi behöver tydligt anmärka att programmet är under utveckling och att det därför är det fullt möjligt att buggar gör uppgifterna omöjlig. Det bör finnas otydligheter i det grafiska gränssnittet och vi ber därför användaren att säga högt vad hen tänker så att vi kan förstå vad som behöver ändras/förtydligas för en användarvänlig design.

4.2 B

Vi testar programmet när det blivit stabilt och de flesta funktioner är fungerande genom användartestning.

Användartestningen skedde genom att ta tag i lämplig kth student efter en föreläsning och försöka få denne att utföra följande uppgifter:

1. Skapa ett solsystem med en planet och en tillhörande måne.
2. Skapa ett dubbelstjärnigt solsystem eventuellt med en tillhörande planet.

Vi tog en student från inda12 som har goda fysik kunskaper och passar in på användar scenario 1. Innan användaren började förklarade vi kort med ord (utan att visa) vad vi ville att hen skulle försöka utföra. Vi förklarade att det grafiska gränssnittet ännu inte är användarvänligt, vilket är anledningen till att vi utför testet, och att eventuella misslyckanden inte beror på användaren.

Vi anmärkte tydligt att programmet är under utveckling och att det därför är det fullt möjligt att buggar gör uppgifterna omöjliga. Vi bad användaren att säga högt vad hen tänker så att vi kan förstå vad som behöver ändras/förtydligas för en användarvänlig design.

Efter ca 1 minut så blev det uppenbart att kontrollerna inte var intuitiva nog då användaren inte förstod hur han skulle göra. Vi förklarade kort vilka mus- och tangent-knappar som gör vad och lät användaren fortsätta.

TODO för stor massa och det blev otydlig infinite bug. framtidsyn slutade fungera korrekt, visade endast nuvarande hastighet.

Vi upptäckte bl.a. att även fast användaren hade goda fysik kunskaper så var det svårt att uppskatta vilka storleksordningar på massor som är lagom för planeter på olika avstånd och med olika hastigheter.

4.3 C

Vid testerna märkte vi att det kanske inte var rimligt med de svårare uppgifterna. P.g.a. tidsbrist hann vi bara med en person och då programmet fortfarande inte var helt klart upptäckte vi en del buggar (dvs. inte bara design-fel).

5 Programdesign

5.1 A

Programmet ska använda opengl för det grafiska med hjälp av biblioteket libgdx¹. Artemis entity system² kommer användas för att lättare separera koduppgifter i mindre klasser där varje klass gör en (1) enskild sak.

Artemis använder sig av komponenter och system, där komponenter håller data och systemen bearbetar dom. Ett system använder endast några få komponenter och arbetar endast på objekt som har dom komponenterna.

Planeterna ska ha följande komponenter:

- Position
- Hastighet
- Acceleration
- Massa
- Storlek
- Färg

Det kommer behövas system för bland annat:

Gravitation vilket behöver använda *position*, *massa* och *acceleration* för att räkna ut gravitations krafter mellan planeterna.

Acceleration vilket behöver *acceleration* och *hastighet* för att förändra hastigheten.

Hastighet vilket behöver *position* och *hastighet* för att uppdatera positionen.

Ritande vilket behöver *position* och *storlek* för att rita planeterna på skärmen.

5.2 B

Programmet använder de utomstående biblioteken libgdx och artemis.

Planeterna representeras av entities med följande komponenter.

- Position (vektor)
- Hastighet (vektor)
- Acceleration (vektor)
- Massa
- Radie
- Färg (RGBA)

¹<http://libgdx.badlogicgames.com/features.html>

²<http://gamadu.com/artemis/index.html>

Fysiksystemen är:

Gravitationssystemet som beräknar gravitationsaccelerationen mellan planeter och summerar dessa för varje planet. Systemet uppdaterar alltså accelerationskomponenten för planeterna.

Hastighetssystemet uppdaterar hastighetskomponenten hos planeter med accelerationskomponenten.

Kollisionsystemet undersöker om och när planeter kommer kollidera inom närmaste tick och hanterar dessa som elastiska stötar. Systemet är ansvarigt för att uppdatera positionskomponenten hos planeter.

Framtidssystemet beräknar hur en markerad planet kommer röra sig inom närmsta framtid och ritar detta på skärmen.

Positionssystemet används av framtidssystemet för att uppdatera planeters positioner med deras hastigheter. Kollisionsystemet är nämligen för långsamt för att framtidssystemet ska vara användbart.

Användargränssnitt-systemen är:

Inputsystemet hanterar markeringar, dragningar, zoomning och liknande. Systemet hanterar interaktion med simulator-världen.

Planet-ritar-systemet ritar planeter i simulatorvärlden.

Gränssnittssystemet hanterar vänstermenyn.

Template-gränssnittssystemet hanterar template-menyn.

Hjälp-gränssnittssystemet hanterar hjälp-menyn.

5.3 C

Det blev ganska många ändringar i hur programmet ser ut inuti.

Vissa system behövdes inte heller, som t.ex. accelerationsystemet vars uppgift utfördes av gravitationsystemet. Kollisionsystemet ersatte Positionssystemet, men det senare blev ändå kvar för att användas av framtidssystemet.

Vidare delades ritningssystemet upp i flera system för att bl.a. enklare hantera interagering från användaren (t.ex. tangentryckningar).

6 Tekniska frågor

6.1 A

1. Vilken algoritm ska användas för att justera planeters position, hastighet och acceleration? Ska t.ex. alla planeter jämföras med alla andra planeter i universumet, eller ska algoritmen vara selektiv och inte räkna planeter för långt borta?
2. Hur hanteras kollisioner? Studsar, spricker eller smälts planeter samman? Hur kommer kollisionssystemet påverka resten av programmet? Behövs systemet snabbas upp?

3. Behöver programmet snabbas upp med parallellisering? I så fall, på vilket sätt?
4. Bör programmet kunna förutse hur planeter kommer röra sig (för att t.ex. visa användaren omloppsbanor). Kan detta implementeras så att det inte kräver för mycket resurser?

6.2 B

Hur vi svarade på ovanstående frågor:

1. Förflyttningen av planeter har delats upp i system och komponenter som beskrivet i 5 Programdesign.

Gravitationssystemet använder ett s.k. Quad Tree för att spara positioner och massor. Varje nod i trädet innehåller referenser till fyra under-träd, den totala massan av alla planeter i trädet och dess under-träd och det s.k. masscentrum av massorna och planterna. Varje planet jämförs sedan med trädet. Om en nods masscentrum är tillräckligt långt bort kan dess mass-total och -centrum användas för att beräkna noden som en planet, snarare än att dyka ner och jämföra med varje planet - vilket annars skulle göras om nodens masscentrum är nära med jämförande planeten.

Kollisionsystemet använder en s.k. *Sweep and Prune*-algoritm (SAP).

Till varje planet skapas en s.k. hastighetscirkel - den har samma position som planeten, men dess radie är planetes radie adderat med längden av planetens hastighetvektor.

HastighetsCirkeln passas sedan in i en fyrkant - dvs. max och min värden beräknas i x- och y-led. Planeterna sorteras sedan efter deras hastighetscirkels x-min.

Det är sedan lätt att kontrollera vilka cirklar som överlappar i x-led.

Om två cirklar överlappar i x-led, kontrolleras de i y-led. Om de även där överlappar kontrolleras om och hur planeterna faktiskt kolliderar - genom beräkna tiderna de kolliderar.

2. Kollisioner hanteras som elastiska stötar. Kollisionsystemet har snabbats upp som beskrivet ovan. Detta är en stor förbättring från att alla planeter jämförs med alla ($O(n^2)$), även om programmet ändå blir långsamt med många aktiva planeter.
3. Parallellisering har endast används i framtidssystemet.
4. Framtidssystemet har implementerats, men utan kollisioner, då dessa gjorde det för långsamt.

6.3 C

Det var inga större problem med de tekniska frågorna. Kollisionsystemet blev inte så snabbt som förhoppats, men fungerar ändå för ett någorlunda stort antal planeter.

7 Arbetsplan

7.1 A

Arbetet pågår under vecka 17 t.o.m. 20. Under denna tid är det 3 övningstillfällen: fre 26 april (v17), fre 3 maj (v18) och ons 15 maj (v20).

Projektplanen har deadline vid den första övningen. Slutrapporten och programmetkoden har deadline vid den sista övningen. Vid varje övningstillfälle ges en muntlig lägesrapport.

Vecka 17 spenderas på att skapa ett git-repository med ett kodskelett och att skriva projektplanen. Om det finns tid över läggs den på att till viss del att börja fylla kodskelettet.

Vecka 18 ska användas till att skriva så mycket av programmet som möjligt. Om det finns tid över börjar användartestningarna v18 och fullgörs annars i början av v19.

Vi räknar med ett fullt fungerande, debuggat och användartestat program i mitten av v19, ons 1 maj. Slutrapporten ska skrivas så mycket som möjligt parallellt med resten av arbetet, men framförallt slutet av vecka 19 används till rapporten. Om det finns tid över kan det användas till debugging och finslipning av programmet.

Vi räknar med att slutrapporten är klar i slutet av vecka 19, fre 11 maj.

Vecka 20 (totalt 2 dagar innan inlämning) används endast om något moment tog längre tid än förväntat.

Vi har ingen specifik arbetsfördelning utan skriver båda i koden och rapporten - däremot kanske inte i samma filer samtidigt. Vi håller kontakt via internet och kan snabbt komma överens om vad som behöver göras för tillfället och vem som gör vad.

7.2 B

Planen lämnades in i tid, men programmet blev fungerande ganska snart, men kodandet fortsatte in i v20.

Första användartestet gjordes onsdag v19. Rapporten påbörjades först lördag v19.

Arbetsfördelning varierade med i sin helhet skrev Johannes större delen av fysiksystemen och Anton större delen av grafiken och programgrunden samt framtidssystemet. Båda skrev dock lite på allt.

Vi skrev ungefär lika mycket på planen och rapporten.

7.3 C

Ett fungerande program var klart ganska tidigt, men att lägga till och förbättra tog tid.

Rapporten börjades därför på efter dess tänkte deadline och vi hann bara med ett användartest.

8 Sammanfattning