Jonathan Wahlund, Filip Reinholdsson, Axel Karlsson

Space Invaders - Slutrapport

A. Projektplanen

Programbeskrivning

Vi tänker göra en version av det gamla arkadspelet Space Invaders i java. Spelet går ut på att spelaren styr en kanon som alltid är längst ned på skärmen, kanonen kan glida i sidled åt höger och vänster. Från skärmens övre del kommer ett antal fientliga rymdmonster, som rör sig fram och tillbaka i sidled och sakta åker nedåt på skärmbilden.
Spelarens måste skjuta alla monster innan någon av dem hunnit ned till botten av skärmen. Desto färre fiender som återstår, desto snabbare rör de sig, dock fortfarande med samma inbördes positioner som de hade när hela formationen först visade sig. Fienderna kan ibland skjuta tillbaka, men inte sikta. I vissa varianter av spelet har spelaren skydd i form av hus, men förstörs gradvis när de blir träffade av rymdmonstren eller spelarens skott.
Vi har tänkt oss bygga en simpel version av detta spel som bas, därefter ändra på saker som vi tycker bör ändras och lägga till ”tweaks” och roliga funktionen där vi känner att det finns rum för sådant.

Användarbeskrivning

Användarna av vårt program behöver nödvändigtvis inte vara särskilt datorvana. Även om det inte är en nödvändighet vore det dock intressant om användarna i alla fall hört talas om Space Invaders tidigare så att de kan göra någon form av jämförelse mellan vår version och de äldre. I övrigt finns det inget speciellt krav som vi ställer på användare.

Användarscenarier

1. Karl-Erik sitter vid sin dator och får en länk till ett spel från sin kompis Mohammed. Han öppnar länken och spelet startar med en knapp som det står “START” på. Genom att han känner igen speltypen så vet han att han ska använda piltangenterna för att flytta sig och space för att skjuta.

2. Pelle har tråkigt en eftermiddag och bestämmer sig för att gå in på sin favoritsida jetekulspel.se. Där ser han att ett nytt spel finns utlagt som han aldrig sett förut! WOW! Han startar spelet och ser en bild på piltangenterna med indikationer för höger och vänster. Han ser även en spacebar som det står SKJUT på. Efter att han förstått kontrollerna så trycker han på start. Väl inne i spelet så förstår han snabbt vad det går ut på då spelet är väldigt simpelt.

Testplan

Då vi tänker använda eclipse så kommer det vara väldigt lätt att bara starta programmet med ett knapptryck. Eftersom vi redan vet från början hur vi vill att spelet skall fungera och att spelet är så simpelt så kommer det vara lätt för oss att snabbt hitta buggar och fel. När vi fått en fungerande bas kommer vi troligtvis prova spelet under och efter varje gång vi ändrar något eller lägger till funktioner i spelet.

När vi har ett fungerande spel så kommer vi be vänner och familj prova på vårt spel, fråga vad de tycker och om det är något som man skulle kunna lägga till. Vi kommer ta in vad de säger och diskutera om någon av dem nämnde någon rolig funktion som kan läggas till i vårt program. Om någon bra idé kommit upp så kodar vi in den i programmet!

Programdesign

Först och främst tänker vi oss en klass Space Invaders som har en metod som ritar upp hela spelet och metoder som ritar upp alla fiender, spelaren och alla skott. Dessa måste sedan ha egna klasser med metoder som ger var de befinner sig och när de skjuter. Klassen Player som hanterar spelaren måste även ha metoder som hanterar vilka knappar som är nedtryckta för att kunna uppdatera spelarens position.

Tekniska frågor

Hur hanterar man input från tangentbordet till ett javaprogram som körs?
Ska vi implementera "hus" och "power ups" i vår version?
Hur ska vi på ett effektivt sätt, hantera saker som försvinner från planen under spelets gång?
Behöver projektiler från spelaren och från fiender hanteras separat?
Vilka interfaces skulle vi kunna använda för att förbättra projektet?
Hur skulle dessa interfaces kunna se ut?

Arbetsplan

Projektet delas upp lika bland alla personer. Vi har rätt öppen tidsplan men siktar på att han en fullt fungerande simpel version uppe någon vecka innan projektet ska vara klart och sedan implementera fler funktioner när basen är klar.
Vi kommer titta mer på GitHub som tydligen skall vara bra när man jobbar med kod tillsammans.

B. Projektutförandet

Programbeskrivning

Programmet har tre olika lägen eller "states". I samtliga lägen spelas det musik i bakgrunden. Första läget som spelet befinner sig i när man kör programmet är "main menyn". Denna meny kan man alltid nå från de andra lägena genom att trycka på escape knappen. På menyn finns det två knappar som man kan välja mellan med hjälp av piltangenterna och "enter knappen". Den första knappen play startar spelet genom att gå till läget vi har kallat för SpaceInvaders och den andra knappen exit avslutar spelet. På denna meny visas även kontrollerna som används då spelet har startats.

SpaceInvaders läget som bygger upp själva spelet är väldigt likt de gamla space invaders arkadspelen. Vår version går ut på att styra ett rymdskepp som befinner sig längst ner på skärmen och endast kan förflytta sig i sidled med hjälp av piltangenterna. Från skärmens övre del kommer fientliga rymdskepp som rör sig i sidled och sakta nedåt mot botten av skärmen.

Spelaren skjuter skott då space knappen hålls in och måste med hjälp av denna funktion skjuta alla fientliga skepp innan de nuddar spelaren eller botten av skärmen för att spelet inte ska förloras. Spelaren har som utgångspunkt endast ett liv.

De fiender som befinner sig närmast botten kan också skjuta men vilken av dessa fiender som skjuter bestäms slumpmässigt. Om en fiende blir träffad av ett skått försvinner den. Det samma gäller då skått kolliderar. Om spelaren blir träffad av skått från fiender förloras ett liv och om spelaren endast har ett liv då den blir träffad förloras spelet.

När en våg av fiender har skjutits ritas en ny våg av fiender upp på skärmen. Ju fler vågor som klaras av ju svårare blir spelet i form av att antalet gånger en fiende måste skjutas för att den ska försvinna ökar. Var fjärde bana kommer en "boss" agerar som en stor fiende med mycket liv.

Spelet har även tre typer av "powerups". Rapid fire minskar tiden mellan skått för spelaren. 1up ger spelaren ett liv till. Spelaren kan som max ha tre liv. Double barrel ger spelaren ytterligare ett skått när spelaren skjuter. Explosive bullet gör så att objekt i närheten av spelarskåttens kollisionspunkter också försvinner.

Spelet har inget mål utan forsätter tills spelaren förlorar. Istället sparas spelarens poäng som blir högre ju fler fiender som har skjutits. När spelaren förlorar går spelet in i GameOver läget. I detta instrueras man att trycka på höger ctrl knapp för att starta ett nytt spel. På denna meny ser man lokalt sparade poäng av de som har spelat tidigare. Man kan även skriva in sitt namn och spara sina poäng här. Som vanligt kan man även trycka på escape knappen för att återgå till huvudmenyn.

Användarbeskrivning

Våra användare är sådana som vet vad Space Invaders är för något. Konceptet ligger klart för dem även om de kanske inte är särskilt datorvana. Äldre användare använder ofta enbart datorn för att skicka mail medan de yngre ofta har väldigt mycket datorvana sen innan och även erfarenhet av spel.

Användarscenarier

1. Karl-Erik är hemma hos Mohammed som läser Datateknik. "Testa det här spelet som vi gjort och säg vad du tycker". Han startar spelet ett dubbelklick och spelet startar direkt. Eftersom han redan spelat Space Invaders förut så vet han redan hur det går till så han hoppar direkt in i spelet. Här märker han att det skiljer sig en aning från originalet och han påpekar dessa skillnader. Han väljer att försöka hitta buggar för att han är "breaka" koden. Trots försök så hittar han inte något fel.

2. Pelle dyker upp hos Mohammed. Precis som Mohammed så blir han också ombedd att testa spelet. Han vet vad Space Invaders är men har inte spelat det tidigare. Genom att läsa informationen som kommer upp så förstår han vilka kontroller spelet har och hur han ska gå till väga för att starta spelet. Han klickar igång spelet på "enter" och börjar spela. Han spelar ett tag, ger några förslag om förbättringar, och slutar med en poäng om 5000. Han sparar sin highscore genom att skriva in sitt namn och trycka på enter.

Tester

Vi lät vänner och familj i vår närhet testa spelet när det hade nått en buggfri, spelbar nivå framförallt i syftet att få respons på vad de ansåg saknades. Innan vi lät någon testa spelet försäkrade vi oss alltid om att de kände till vad spaceinvaders var. Detta gjorde vi eftersom vi ville ha testpersoner som hade en idé om vad vi ville uppnå.

Testresultat

I överlag så fick vi mycket god respons. Många av de förslag som testpersonerna kom med var saker som antingen redan var planerade eller som det redan fanns idéer för. Testningen gav oss därför en god bild av vilka tillägg som vi skulle fokusera på. Idéen om att ha en boss efter ett visst antal vågor av fiender var en idé som kom fram under användartestning och som vi därför implementerade.

Det fanns inga buggar som krävde andra än oss själva att spela för att märka, så det var enbart själva spelupplevelsen som testades.

Programdesign

Programmets kärna är GameSetup.java som ritar upp fönstret och hanterar spelets tre states MainMenu, SpaceInvaders och GameOver. Dessa klasser bygger vidare på klassen BasicGameState. Därför har alla states metoden init() som initierar objekt, metoden render() som ritar upp objekt på fönstret och metoden update() som uppdaterar värden allteftersom programmet körs. Alla states använder sig också av klasserna Textures som hanterar de bilder som ska ritas upp på fönstret och Sounds som hanterar musiken och ljuden som spelas.

MainMenu använder sig av klassen Button som håller reda på de knappar som ska ritas upp och alterneras mellan. Klasserna Sounds och Textures används även av alla tre states.

SpaceInvaders använder sig av klasserna ExplosionAnimation, Player, Powerup och Projectile och Enemies som i sin tur använder sig av klassen Enemy,. Alla dessa klasser hanterar de olika objekten som ska ritas upp på fönstret. Det som måste förtydligas är att klassen Enemies skapar objekt av typen Enemy. Projektet är uppbyggt på de sättet eftersom fienderna alltid förflyttar sig tillsammans som en entitet. Därför har klassen Enemies exempelvis metoden move() som förflyttar alla fienderna.

GameOver hanterar sparandet av poäng och använder sig av klassen highscore för att skapa ett nytt highscore.

Arbetsplan

Projektet har delats upp lika bland alla personer. Vi har följt vår tidsplan och hade som planerat en fungerande förstaversion (utan grafik) mer än en vecka innan slutdatumet och de tillägg som vi hade planerat har implementerats i slutversionen.

C. Jämförelse mellan plan och utförande

Programbeskrivning

Den största skillnaden mellan det som var planerat och hur slutresultatet blev är implementationen av StateBasedGame som vi valde att göra. Eftersom vi inte visste att detta verktyg fanns när vi planerade projektet var det svårt att förutse att projektet skulle se ut som det nu gör.
Bortsett oplanerade sätt att implementera olika menyer i spelet är spelet i sig väldigt likt det som var planerat. En skillnad är att vi inte implementerade hus eftersom spelet inte uppnådde en sådan svårighetsgrad att de blev nödvändiga. Till skillnad från planerat och till skillnad från originalspelet ökar inte fiendernas hastighet när de blir färre. Vi struntade i det tillägget eftersom svårigheten i senare vågor blir tillräckligt stor då fienderna får mer liv per våg.

En annan sak som vi inte hade förespått var hur många klasser vi skulle ha. Vi kände att det var behagligt att dela upp spelet i många klasser för att lättare hantera alla de olika elementen.

Användarbeskrivning

Ingen noterbar skillnad mellan beskrivningen och den faktiska då den var väldigt öppen från början.

Användarscenarier

De tänkta scenariona och hur det faktiskt (ungefär) gick till skiljer sig främst då inga tester gick till genom att en länk skickades till någon hemsida. Koden finns inte upplagd på något sådant sätt och testerna gick till på våra egna datorer. Detta för att vi skulle kunna snabbt ändra i koden vid behov och inte lägga ner tid på att försöka exportera ett ofärdigt program. I övrigt så gick det ungefär till som tänkt.

Tester

Tester och felsökning i de tidiga stadierna gick som planerat då man tidigt märkte vid körning av programmet om saker funkade som planerat eller inte. Vi utförde användartestningen vid ett senare stadie än planerat, därför hjälpte inte användartestningen till i felsökningssyfte. Däremot var den hjälpsam för urval av vilka tillägg som skulle utvecklas härnäst.

Programdesign

Bortsett från implementationen av StateBasedGame stämmer grunderna av resultatet bra överens med det som var planerat. Det är dock ett flertal klasser som har lagts till i efterhand för att förbättra programmet som går långt utöver det som var planerat. Exempel på detta är hur uppdelandet av klasserna Enemies och Enemy vilket inte var planerat underlättade mycket. Andra klasser som lades till som inte var planerade för var Textures och Sound.

En annan skillnad i planering och resultat var hur metoder som hanterar vilka tangenter som var nedtryckta inte tillhörde Player klassen utan istället tillhörde SpaceInvaders. Anledningen till att denna ändring gjordes var att vi inte kände till hur input från tangentbordet hanterades i slick och hur mycket smidigare det blev att uppdatera sådana aspekter i den inbyggda update metoden.

Tekniska frågor

Hur exporterar vi vårt slick2d spel så den går att köra utanför eclipse?

Arbetsplan

Den enda skillnaden i arbetsplanen är hur vi använt Dropbox istället för GitHub. Detta val av metod känns i efterhand något felaktigt då GitHub anses vara ett mer använt och bättre anpassat verktyg för våra behov. Anledningen till att det blev Dropbox var att vi kom igång med själva programmet så snabbt att det skulle vara konstigt att avbryta vårt arbete för att istället lära oss GitHub som inte skulle förbättra vår kodarupplevelse så mycket ändå. Vi valde att fokusera på projektet istället för hur vi synkade vår kod.

Sammanfattning

Den största lärdomen som kan dras från projektet är förmågan att arbeta i grupp inom programmering på sätt som inte gjorts tidigare. Sådana kunskaper som vi fått av projektet är kommunikation om vem som gör vad. Man har även sätt syftet och anledningen till att skriva väldokumenterad och lättläslig kod i praktiken.

Mer konkreta programmeringsrelaterade kunskaper som vi fått av projektet är mycket större förståelse för hur programmering för 2D grafik eller GUI i allmänhet kan se ut i java. Vi har också lärt oss en mängd nya metoder att hantera bilder, ljud och input från tangentbord.

Trots våra stora framsteg ser vi fortfarande stora utvecklingsmöjligheter för vårt program. Bortsett från de tillägg som vi hade planerat från början som inte kom med(hus och fiender som ökar i hastighet) finns det stor utvecklingsmöjlighet inom exempelvis powerups. Några lätta powerups att implementera är hastighetsökning hos spelaren, hastighetsminskning hos fiender eller större projektiler. En annan utvecklingsmöjlighet är ett tillägg av “downgrades” alltså motsatsen till powerups som gör spelet svårare för spelaren.