inda13 - Projekt
Javaga

Gustav Dänsel
Lukas Lundmark

16 maj 2014

1 Programbeskrivning

1.1 Programbeskrivning A

Vi siktar på att skapa en Galaga-klon i Java. Det ska alltså vara en top-down 2D arkadspel. Vi planerar att använda oss av libGDX-biblioteket.

Spelet fungerar så att man kontrollerar ett rymdskepp som ska skjuta ned utomjordingar som kommer från fönstrets övre kant och försöker röra sig nedåt. Målet är att förstöra dem innan de försvinner ut skärmen. Se http://en.wikipedia.org/wiki/Galaga

1.2 Programbeskrivning B

Programmet är utformat som ett klassiskt arkadspel som i huvudsak ska efterlikna det klassiska spelet Galaga med top-down view där man styr ett rymdskepp och ska skjuta ner fiendeskepp som kommer ner ifrån toppen av skärmen och försöker döda en. Vi har implementerat spelet med hjälp av programbiblioteket libGDX som har hjälpt oss med renderingen och inputhanteringen.

1.3 Programbeskrivning C

Programmet blev ungefär som vi inledningsvis föreställde oss, att kalla det galaga-klon är dock kanske inte helt korrekt längre då vi har implementerat många saker i spelet som inte direkt är associerat med de ursprungliga galaga-spelen. Spelet innehåller dock de grundläggande aspekterna med rymdskepp och fiender och är fortfarande i ungefär samma stil som de klassiska arkadspelen som vi försökt efterlikna.

2 Användarbeskrivning

2.1 Användarbeskrivning A

Vi tänker oss att personer som är sugna på att spela klassiska retrospel utan att köpa en arkadmaskin kan vara en potentiell målgrupp. Annars ingen, tyvärr.

2.2 Användarbeskrivning B

Vi föreställer oss att potentiella användare av vårt program skulle kunna vara vilken människa som helst som söker att slösa bort någon minut på att spel ett enkelt men underhållande spel i klassisk arkadspels-anda.

2.3 Användarbeskrivning C

Vi har insett att det nog inte enbart är gamla arkad-nördar som kan se tjusningen med vårt spel utan vilken Svensson som helst skulle nog kunna se det roliga i vårt spel. Även ovana spelare ska kunna spela spelet vilket gör det viktigt att man har enkla kontroller och enkla menyer och ger användaren instruktioner som är lätta att följa, även för den ovane.

3 Användarscenarie

3.1 Scenarie A1

Sent på kvällen, dagen innan tentamen för SF1626, pluggångesten liger tungt över Kalle. Han känner att han behöver ta en paus från att inte plugga och göra något annat och får ett tips om den supercoola Galagaklonen Javaga. Han laddar ned spelet från thepiratebay och börjar prokrastinera hårt. Han styr sin rymdhjälte med piltangenterna och skriker avfyra! med mellanslag. Vilken upplevelse det är! Han har aldrig varit med om något liknande! Kalle sitter uppe och spelar hela natten och missar tyvärr tentan, mycket sorgligt. Based on a true story.

3.2 Scenarie A2

Pappa Per är på väg hem från en lång och slitsam dag på jobbet. På vägen hem ser han reklamen för Game On 2.0 på Tekniska Museet. Han kommer och tänka på alla fantastiska kvällar med sina vänner i arkadhallen när han var liten. Han tänker på Galaga och hur kul det var att spela. När han väl kommer hem så bingar han fram en lista på Galagakloner och den som ligger högst upp och har bäst omdömen är ju givetvis Javaga. Per laddar ned spelet och börjar tidsresan till barndomen. Och vilken resa det är! Sicken resa, Mycket fräck! Han känner sina mossiga gamla färdigheter komma tillbaks och upplever sann eufori. Fantastiskt.

3.3 Scenarie B1

En vacker eftermiddag sitter Sven och försöker programmera det sista på sitt inda-projekt när plötsligt en mycket upphetsad och ivrig liten krabat kommer fram och vill att Sven ska testa hans nya spel, Javaga. Krabaten i fråga förklarar att spelet är baserat på det klassiska arkadspelet galaga, Sven har aldrig hört talas om detta spela men beslutar sig ändå för att testa det. Han får spel-filen och prövar spelet. Utan några större svårigheter inser Sven att han ska styra skeppet på skärmen med hjälp av pil-tangenterna och och skjuter med hjälp av mellanslag eftersom spelet tydligt informerar honom om det. Därefter spenderar han en lång stund med att skjuta ner rymdskepp men frustreras en smula över att han inte vet hur långt han har kvar tills han har klarat spelet eller tills han dör. Efter en stund så går hans virtuella avatar sin oundvikliga död tillmötes och Sven är nöjd med spelandet och återgår till det han gjorde tidigare.

3.4 Scenarie B2

En mycket regnig lördagsförmiddag sitter Per-Åke i sitt vardagsrum och ser på TV. Plötsligt kommer hans son in i rummet och vill att han ska testa ett nytt roligt spel som han har fått ifrån en vän i skolan. Per-Åke som inte har något bättre för sig följer med sin son för att testa spektaklet. Per-Åke som inte är en van spelare av dataspel är lite orolig hur detta ska gå men finner sig snabbt tillrätta då de enkla kontrollerna förklaras för honom i spelets början och det kommer ett litet antal fiender på skärmen, han finner att detta var ganska så underhållande och fortsätter att skjuta ner utomjordingar tills han oundvikligen förlorar.

3.5 Scenarie C

Eftersom vi nu var lite mer medvetna om hur spelaren i fråga skulle interagera med spelet gjorde vi lite ändringar i användarscenarierna och lät de fokusera lite mer på interaktionen mellan användare och program.

4 Testplan

4.1 Testplan A

Vi planerar att muta folk till att spela spelet och ge oss bra feedback. Vi planerar att göra detta vid minst två tillfällen, kanske mer om det finns tid. En viss mängd automatiserade tester kan förekomma.

Testanvändaren förväntas spela spelet tills ögonen blöder och därefter berätta för oss hur fantastiskt det var.

4.2 Testplan B

Fem personer fick sammanlagt pröva spelet då det var i sin prototyp-fas. Alla testare fick som uppdrag att helt enkelt spela spelet medan vi observerade och sen bad vi om att få deras åsikter gällande upplevelsen.

Utifrån vad vi som observatörer kunde bedöma fanns det inga direkta svårigheter för de nya användarna att komma underfund med hur spelet fungerade, förutom en av testerna som inte var en van spelare och hade vissa svårigheter att lära sig styra sitt skepp ordentligt. Utöver det så verkade de flesta användare med lätthet förstå vad spelet gick ut på och klarade sig ganska bra.

Feedbacken kan generellt sammanställas till tre återkommande punkter:

- Avsaknaden av en poängräknare verkade störa några av testarna.
- Avsaknaden av någon form av nivå-räknare gjorde några av användarna frustrerade då de ville veta hur långt i spelet de kommit
- Svårighetsgraden på den första nivån var enligt några testare lite för svår, särskilt om man var ovan vid denna typ av spel

4.3 Testplan C

Användartesterna gav oss den feedbacken vi behövde för att vi skulle kunna notera vissa luckor i vår design, dels var avsaknaden av en nivåräknare ganska besvärlig då det gjorde det svårt att få en uppfattning om man gjorde framsteg eller inte. Dessutom fanns det en avsaknad av en en poängräknare vilket många noterade var ett måste för ett spel i klassisk arkadspels-anda. Några testare tyckte även att den första nivån var lite för svår för ovana spelare vilket vi sedan tog i åtanke när vi sedan omdesignade banorna i spelet.

5 Programdesign

5.1 Programdesign A

Tanken är att dela upp programmet i flera klasser. Primärt så tänker vi oss att vi har dessa klasser:

- Input
- Render
- Game-Logic
- Units
- File I/O

5.1.1 Input

Här finns metoder för att läsa indata från kontroller såsom tangentbord och mus, samt skicka vidare dessa till relevanta klasser.

5.1.2 Render

Här görs det tunga jobbet att rita bilden som ska visas på skärmen.

5.1.3 Game-Logic

I denna klass sköts all logik - såsom hur enheter ska förflytta sig, om skott träffar eller inte och liknande saker.

5.1.4 Units

Units innehåller beskrivningar och parametrar för alla olika typer av skeppss-om finns i spelet.

5.1.5 File I/O

För att läsa och skriva till hårddisk, för t.ex. inställningsfiler.

5.1.6 Programdesign B

I slutändan kom programmet att innehålla många klasser, men dessa fyra är de som programmet huvudsakligen är baserat på

- InputManager/GameKeys
- GameStateManager
- GameState
- SpaceObjects

5.1.7 InputManager/GameKeys

1 InputManager ansvarar för avläsningen av användaren input ifrån musen och ifrån tangentbordet och GameKeys lagrar all input så den finns tillgänglig för alla objekt som vill ha den informationen.

5.1.8 GameStateManager

GameStateManeger ansvarar för kontrollera spelets nuvarande ”state”, det vill säga om spelet ska vara i menu-läge, pause-läge eller spel-läge.

5.1.9 GameState

GameState är en abstrakt klass som beskriver ett state i spelet(exempel på implementationer innefattar PlayState, PauseState, WelcomeState mm.) Ett GameState ansvarar för all logik i den delen av spelet som den ansvarar för vilket innefattar för att rendera allting i sig som ska renderas och dessutom att uppdatera alla objekt i sig och läsa användar-input ifrån GameKeys ifall det är nödvändigt.

5.1.10 SpaceObject

SpaceObject är en abstrakt klass som beskriver alla typer av objekt som interagerar med varandra på spelets skärm(exempel på implementationer innefattar Player, Enemy, Bullet mm.), den innehåller information som beskriver objektets position, dess sprite, hur den ska uppdateras, ritas och bli borttagen.

6 Programdesign C

Designen som programmet slutligen fick är väldigt likt den ursprungliga idén, valet att ha ett spel som är ”state-baserat” med en GameStateManeger och GameStates där varje state tar hand om sin egen logik och sin egen rendering gjordes eftersom det gjorde det lättare att implementera menyer och att starta om spelet. Vi valde dessutom att inte implementera någon form av filhantering, mest på grund av att tiden inte räckte till för att implementera något sådant.

7 Tekniska frågor

7.1 Tekniska frågor A

Den största tekniska frågan ligger i hur vi implementerar biblioteket. I det problemer finns det även fler mindre problem, såsom vilken typ av input vi ska använda o.s.v.

Klassiska problem såsom animation sköter det bibliotek (libGDX) vi använder.

7.2 Tekniska frågor B

Den största tekniska frågan under projektets gång var att fundera ut hur man skulle implementera och använda sig av spelbiblioteket libGDX vilket inte visade sig alltför problematisk(även om dokumentationen ibland var lite bristande). En annan teknisk fråga under projektets gång var hur vi skulle undvika minnesläckor i spelet vilket löstes med att man låter varje state i

spelet rensa sina objekt på sprites och göra sig av med dem på korrekt sätt så de rensandes ifrån arbetsminnet.

7.3 Tekniska frågor C

Det som var den mest överraskande frågan som vi stötte på under projektet var hur vi skulle hantera och undvika minnesläckor vilket var något vi inte alls hade förrutsätt att vi skulle behöva hålla på med men blev så illa tvungna till när spelet började kunna ta upp 3 Gb RAM om man spelade länge. Detta var dock inte alltför komplicerat att lösa då det helt enkelt bara var att göra sig av med alla sprites i alla objekt på korrekt sätt enligt libGDX specifikationer och låta varje state i spelet ansvara för att rensa ut alla objekt ordentligt när den väl var klar. De flesta andra tekniska frågor gällande rendering och filhantering löstes förhållandevis enkelt med hjälp av libGDX-biblioteket.

8 Arbetsplan

8.1 Tidsplan A

- Läsa och sätta sig in i biblioteket och dess dokumentation - 2/5
- Första fungerande prototyp - 9/5
- Testning under helgen 9/5 till 12/5
- Finslipning mer test till 16/5

Planen är att använda GitHub för att samarbeta på koden. Vi tänker oss att vi delar lika på arbetsuppgifterna och arbetar tillsammans. Då projektet är relativt litet så blir det enklare så då vi båda har koll på all kod och vi slipper läsa ikapp.

8.2 Tidsplan B

- Vi satte oss in i libGDX och dess dokumentation och dess wiki - 2/5
- Vi skapade ett fungerande spel som gick att testa - 9/5
- Vi testade vår prototyp under helgen 9/5 till 12/5
- Vi lade till fler klasser och olika saker i spelet och fixade saker enligt användartesterna 16/5

För att samarbeta med koden användes GitHub vilket var till stor hjälp under projektet.

8.3 Tidsplan C

Under projektet så följdes projektplanen väldigt bra och sammarbetet med hjälp av GitHub gick också bra utöver lite tekniskt krångel.

9 Sammanfattning

Under projektets gång har vi bland annat lärt oss hur man använder GitHub för att dela och uppdatera kod vilket har varit en stor fördel under projektet och har underlättat mycket av arbetet under projektet. Vi har dessutom lärt oss hur man sätter upp spelbiblioteket libGDX och använder de resurser som finns i det för att enkelt skapa ett relativt snyggt och spelbart spel. De potentiella förbättringarna på projektet är många, exempelvis skulle det vara intressant att implementera animationer, flera typer av fiender och skott, dessutom har libGDX stöd för partikelfysik vilket skulle vara kul att experimentera med. Dessutom saknas det några ganska grundläggande delar i spelet, som en game over-skärm och en mer användbar menu där man kan sköta ljud och hoppa direkt till olika nivåer i spelet.