Projektrapport: Enkelt RPG-spel i Java

Beskrivning av spelet

Mitt spel är ett textbaserat RPG (role-playing game) där spelaren får välja mellan två hjälte klasser och ska ta sig igenom en dungeon med ett valfritt antal nivåer. Spelet är utformat att var femte nivå väntar en boss och man vinner när man har besegrat alla nivåer. I spelet finns även en shop att köpa vapen och liv samt finns där möjlighet att aktivera passiva abilities som hjälper en på vägen.

Planering

Varje programerings dag började med att skissa upp ideer(brainstorm) som classer, funktioner, features e.t.c i notepad. Dessa ideer påbörjades sedan att koda och modifierades under tiden. Allt som kodades lades upp i en "projektpärm" kallad grupp på github. Det skapades även klassdiagram via mermaid för att på ett lättare sätt kunna visualisera det man har tänkt sig skapa / har skapat. Jag tror att mitt mitt sätt att arbeta var väldigt likt den agila metoden då snabba förändringar kunde ske utan att behöva följa en specifik struktur. Arbetet planerades in i stilen "whenever i can, i will code" vilket innebär att när man har möjlighet så kodar man, detta skedde främst under kväll/natt tid. Jag valde att koda ett objektorienterad program i Java för att skapa spelet. Jag använde även mig av Git för versionshantering och commitade i princip varje gång jag kände mig nöjd med en uppgift och comittade igen vid modifiering. Allt för att underlätta en klar och läsbar log. Jag använde mig av designmönstret singleton för att säkerställa att det endast fanns en instans av spelet i gång samtidigt.

Utveckling

Jag började med att skapa en Main, meny klass, monster klass, hero klass och en subklass Warrior. Tanken var att göra det enklaste först, vilket var att få spelet att starta och avsluta genom en meny klass och while loop med switch i main. Därefter skapades en hero klass och en subklass warrior som var tänkt att ha så lite kod som möjligt i. Tanken var att endast modifieringar av Hero klassen som karraktiserar en Warrior ska finnas i Warrior klassen. Detta för att inte behöva skriva dubbel kod i mina sub klasser. När jag fick den att funka med alla modifieringar, passiva abilities, guld & xp så gjordes detta för en subklass Warlock. Dock vill jag tydliggöra att warlock klassen var det sista som gjordes i mitt spel.   
  
I samband med att försöka få en fungerande hero & warrior klass skapades en monster klass. Dessa klasser hänvisades i main klassen som för det mesta bestod utav while loopar & switches. Spelet i sitt tidiga stadie började med att man endast kunde välja en hero (warrior) som slåss mot ett monster. Båda har ett default värde på hp och skada som de gör och hp dras av varje gång man tar skada. Vinner heron får man en ny lvl, fullt hp och mer skada. Dör heron så förlorar man spelet. Jag ville göra det så enkelt för mig som möjligt och upptäckte att ifall man definierar default värden på hp & skada så kan man använda lvl som en modifierare till dessa värden. D.v.s vid varje ökad lvl så ökar också hp & skada enligt en algoritm. Detta sparare väldigt mycket tid och energi då jag även hade en liten tanke på att skapa specifika värden för varje lvl.

Spelet utvecklades sedan väldigt naturligt, så fort kampen var bugg-fri lades det till nya saker som t.ex en boss var femte nivå. Där använde jag mig av en modulo 5 algoritm i början av varje nivå för att se ifall jag ska slåss mot en boss eller ett vanligt monster.  
  
Därefter lades det till möjlighet för spelaren att kunna välja sin attack  
  
Vi började med att skapa en grundläggande struktur för spelet, där spelaren kunde röra sig mellan rummen och interagera med dem. Därefter la vi till funktionalitet för att skapa monster i rummen och möjligheten att bekämpa dem. Vi implementerade också en enkel kampmekanik, där spelaren och monstret kunde attackera varandra och förlora hälsa.

Vi la också till ett inventariefönster, där spelaren kunde se sina tillgängliga vapen och föremål. Vi implementerade också ett enkelt system för att uppgradera vapnen.

När vi hade en fungerande grundversion av spelet började vi lägga till fler funktioner, som bossen som skulle besegras för att avsluta spelet, och flera olika sorters monster med olika egenskaper. Vi la också till en mer avancerad kampmekanik med specialattacker och försvar.

Vi implementerade också en enkel spar- och laddningsfunktion, så att spelaren kunde spara sitt spel och fortsätta vid ett senare tillfälle.

Datatyper

I spelet används flera olika datatyper för att representera olika aspekter av spelets funktioner och data. Nedan följer en beskrivning av de viktigaste datatyperna och varför de är fördelaktiga för spelets implementation:

* Integer (heltal): Används för att representera numeriska värden som karaktärens hälsa, attackstyrka och föremålets värde. Integer är en enkel datatyp som har låg overhead och är enkel att hantera. Dessutom erbjuder Integer en hög precision och snabb prestanda vid aritmetiska operationer.
* Boolean (boolesk): Används för att representera sant/falskt-värden som om en attack träffade eller missade, eller om karaktären är i liv eller död. Boolean är en enkel datatyp som tar upp minimalt med minne och är lätt att hantera.
* String (sträng): Används för att representera textdata som namn på karaktärer och föremål, eller meddelanden som visas på skärmen. String är en flexibel datatyp som kan hantera en mängd olika tecken och format.
* Enum (uppräkning): Används för att definiera en begränsad uppsättning av värden, som t.ex. olika typer av föremål eller monster. Enum är användbart för att undvika stavfel och för att se till att bara giltiga värden används.
* Array (array): Används för att lagra en samling av värden av samma datatyp, som t.ex. en lista av föremål eller monster på kartan. Array är en effektiv datatyp som tillåter enkel åtkomst till individuella element och snabb iteration över hela samlingen.
* Object (objekt): Används för att representera en samling av relaterad data och funktioner, som t.ex. en karaktär eller ett monster. Objekt är användbart för att organisera komplexa datastrukturer och för att dela återanvändbar kod mellan olika delar av spelet.
* Utöver de inbyggda datatyperna i programspråket använde spelet också egna datatyper som utformats specifikt för spelets behov. Till exempel användes en Position-datatyp för att representera en plats på kartan, som bestod av två heltal som representerade x- och y-koordinater. En annan egendefinierad datatyp var Item, som innehöll information om föremålets namn, värde och effekt på karaktärens egenskaper.

Användningen av egna datatyper var fördelaktig för spelets implementation eftersom det gjorde koden mer lättförståelig och minskade risken för stavfel och buggar. Dessutom gjorde det det lättare att återanvända kod och lägga till nya funktioner i spelet i framtiden.

Vem gjorde vad:

I vårt projekt fördelade vi arbetsuppgifterna på följande sätt:

**Kerim**: Kerim var ansvarig för design och implementering av i stort sett allt. Alla klasser, metoder, strukturer, brainstorming, logföring och skapandet av klass diagram.

**ChatGPT**: ChatGPT var ett stöd i detta projekt. Ain användes främst till att förklara varför vissa buggar uppkom samt gav sin åsikt gällande koden som skapades.

Vad kunde ha gjorts bättre:

Spelet är i helhet inte på en avancerad nivå. Det finns otroligt mycket utväcklingsmöjligheter. Nedan kommer några exempel.  
\* Passiva abilities hade gått att göra mer avancerat. T.ex att man kan aktivera dem i shoppen efter en viss lvl. Nu aktiveras bådas automatiskt när man lyckas döda bossen på lvl 5.  
  
\* Weapons som förstörs när man slår på fienden. Nu har man alltid ett vapen som inte går sönder.  
  
\* Utveckla shop valen. Möjliggöra flera vapen som har olika attributer och gör det lämpligare att variera beroende på vilken hero subclass man har valt.  
  
\* Hp potions. Nu resettar heron's liv vid varje ny lvl. Det hade kunnat göras att den inte resettas och att man kan köpa potions i shoppen som läggs till i ens inventory.  
  
\* Defense. Nu har defense ingen betydelse mer än printad text. Det vore intressant att lägga till en algoritm som beräknar att en viss andel av ens defense minskar monstrets attack och vice versa.

\* Mana & mana potions. Likt hp bör en spell caster class ha mana som den förbrukar vid special attacker. Denna mana ska även gå att fylla på.  
  
\* HP & mana regen. Man hade kunnat skapa så att ens hp/mana fylls på en viss andel t.ex varje runda samt som skalar med lvl och kan påverkas av ens items.

\* Flera monster samtidigt. Det vore väldigt intressant att möjliggöra att slumpa fram olika monster som man slåss mot samtidigt. Där man också får möjliggöra vem man får attackera.  
  
\* Gold funktion. Jag hade kunnat utveckla min goldfunktion till att ge en random summa gold efter varje monster, lite mindre för monster och lite mer för boss.  
  
\* Xp funktion. Ett xp system finns inte i nuläget men det hade gått att skapa ett där man får xp efter varje dödat monster som samlas i en "xp bar" hos heron. När xp baren når sitt satta värde får heron en ny lvl och hans attributer ökar(d.v.s attack och skada).

\* Save game funktion. Det mest intressanta som är främmande för mig i nuläget hade varit att kunna spara det påbörjade spelet på en fil så att man inte behöver börja om varje gång man startar spelet.   
  
Hade där funnits mer tid och resurser, till exempel andra människor att bolla ideer med hade dessa bitar säkerliggen gått att implementera. Jag hade dock även velat gå igenom mitt spel som jag har åstadkommit, metod för metod med någon senior programerare då jag vill lära mig nu i mitt tidiga utvecklings skede hur man skriver metoder på bästa sätt utifrån KISS, YAGNI e.t.c. Jag är generellt alltid tveksam till ifall jag skriver min kod på det bästa sättet. Det hade också varit väldigt roligt och lärorikt att få implementera allt man har lärt sig och saker man inte riktigt har fått använda som t.ex alla olika typer av listor, maps , trees, enum o.s.v. Även om det låter väldigt onödigt så hade det varit otroligt gynnsamt ur ett lärande perspektiv.

Slutsatser:

Jag är i helhet väldigt nöjd med detta spel. Anledningen till varför jag inte anser det vara på en avancerad nivå tror jag ligger i att allt hårda arbete där detta var helt nytt för mig gjordes till hangman spelet. Så för detta projekt stod jag redan på stadiga ben. Det som också har underlättat är att jag har kollat på väldig mycket videomaterial online som jag också har antecknat på datorn. Anteckningarna har blivit som en "referens" som jag hela tiden har kunnat vända mig till ifall jag blev osäker hur saker skulle kodas. Jag är väldigt glad att jag fick mindre motgångar för detta projekt då det har fått mig att tyckta det är kul när jag kodar. Hade jag haft samma motgångar som med hangmanspelet hade denna rapport haft ett helt annat innehåll.

Klassdiagram



Backlog

Övergripande:

* Design och implementering av monster-klassen och dess metoder (Kerim)
* Design och implementering av Boss-klassen och dess metoder (Kerim)
* Design och implementering av Hero-klassen och dess metoder (Kerim)
* Design och implementering av Warrior-klassen och dess metoder (Kerim)
* Design och implementering av Warlock-klassen och dess metoder (Kerim)
* Design och implementering av Menu-klassen och dess metoder (Kerim)
* Design och implementering av shop-klassen och dess metoder (Kerim)
* Design och implementering av huvudprogrammet och dess metoder (Kerim)
* Skapa metoder för att generera slumpmässiga nummer och bestämma om en attack träffade eller missade (Jonna)
* Skapa metod för att låta spelaren välja sin karaktär och implementera flera metoder för att bestämma om en attack träffade eller missade (Kalle)
* Skapa metoder för att placera monster och föremål på kartan och implementera möjligheten att flytta karaktären runt på kartan (Pelle)
* Hantera interaktionen mellan karaktären och föremålen på kartan (Johanna)
* Initialisera spelet och börja spelet, implementera möjligheten att spara och ladda speldata och hantera interaktionen mellan karaktären, monster och föremål på kartan (Sören)
* Skapa en användarvänlig gränssnitt (Kalle)
* Testa spelet för att säkerställa att det fungerar som det ska och skapa dokumentation (Jonna, Pelle, Johanna, Sören)
* Skapa en rapportmall och sammanställa projektets dokumentation (Jonna, Sören)

Utförligare backlog

**Jonna**:

* Implementera fler olika typer av monster, med olika egenskaper och attackmönster.
* Skapa en visuell representation av monster på kartan för att göra det enklare för spelaren att se var de befinner sig.
* Implementera en bossmonster-funktion med unika egenskaper och högre svårighetsgrad.
* Ansvarig för att skapa en metod för att slumpmässigt generera monster på kartan med olika styrkor och egenskaper.
* Skapa en metod för att beräkna erfarenhetspoäng som spelaren får efter att ha besegrat ett monster.
* Implementera visuella effekter för att markera när ett monster blir attackerat och när det dör.

**Kalle**:

* Implementera möjligheten att uppgradera karaktärens förmågor och attribut genom att använda erfarenhetspoäng eller föremål som spelaren hittar på kartan.
* Skapa fler karaktärsval för spelaren att välja mellan, med olika egenskaper och förmågor.
* Implementera en enkel dialogfunktion där spelaren kan interagera med NPC:er på kartan.
* Ansvarig för att skapa en metod för att räkna ut spelarens hälsa och att uppdatera hälsostatusen på gränssnittet.
* Skapa en metod för att hantera spelarens inventering av föremål och möjligheten att använda dem.
* Implementera en poäng- och rankingssystem baserat på hur snabbt spelaren kan besegra alla monster på kartan.

**Pelle**:

* Skapa fler typer av föremål som spelaren kan hitta på kartan, som t.ex. vapen, rustningar, föremål för att återställa hälsa, etc.
* Implementera en "fog of war"-funktion som döljer delar av kartan tills spelaren utforskar dem.
* Implementera en "random map generator"-funktion som skapar nya kartor varje gång spelet spelas.
* Ansvarig för att skapa en metod för att lägga till nya nivåer på kartan när spelaren har besegrat alla monster på en nivå.
* Skapa en metod för att skapa olika typer av föremål med olika egenskaper och effekter.
* Implementera en "fog of war"-funktion som begränsar spelarens synfält och avslöjar nya delar av kartan när spelaren utforskar den.

**Johanna**:

* Implementera en handelsfunktion där spelaren kan sälja och köpa föremål.
* Skapa fler interaktiva föremål på kartan, som t.ex. dörrar, fällor, etc.
* Implementera en funktion för att skapa egna föremål och lägga till dem på kartan.
* Ansvarig för att skapa en metod för att slumpmässigt generera föremål på kartan med olika egenskaper och effekter.
* Skapa en metod för att implementera olika typer av vapen och rustningar som spelaren kan använda för att öka sina chanser att överleva i strid.
* Implementera möjligheten för spelaren att sälja och köpa föremål från en handlare på kartan.

**Sören**:

* Implementera en högsta poäng-lista som spelarna kan tävla mot.
* Skapa en funktion för att spara och ladda flera sparfilmer samtidigt.
* Implementera fler spelalternativ, t.ex. svårighetsgrad, ljudinställningar, etc.
* Ansvarig för att skapa en metod för att spara och ladda speldata från en databas.
* Implementera en "game over"-funktion som avslutar spelet när spelaren dör eller klarar av alla nivåer.
* Skapa en metod för att hantera spelets ljud- och musikeffekter.