Projektbeskrivning

Plattformsspel

2032-03-22

Projektmedlemmar:
Johannes Eriksson joeri765@student.liu.se
Yousef Drgham youdr728@student.liu.se

Handledare:
Isak Horvath isho220@student.liu.se

Innehåll

Projektplan

1. Introduktion till projektet

Vi har valt att göra ett platformer spel vilket är ett spel som går ut på att spelaren ska ta sig igenom en bana som består av plattformar, hinder och fiender. För att ta sig fram springer och hoppar spelaren mellan de olika plattformarna och undviker fienderna.

Vår idé är att spelaren ska under en begränsad tid ta sig igenom banan av hinder och plattformar får att nå ett mål som tar sig till nästa bana, om man inte lyckas ta sig till målet inom tiden ska man skickas tillbaka till starten och behöver då börja om från början. Man ska även kunna hitta power ups som antingen kan förlänga tiden som krävs för att komma i mål eller under en kort tid ökar din hastighet eller hopplängd. Vi har även tänk att lägga till fiender och även en boss om man lyckas komma till slutet av spelet.

2. Ytterligare bakgrundsinformation

Vi vet inte riktigt vad vi ska svara på den här frågan

3. Milstolpar

#	Beskrivning
1	**Fönster:** Vi börjar med fönster så att man sedan kan lägga till dom andra delarna till fönstret när dom är implementerade så att vi kan se om vi har gjort det på rätt sätt.
2	**Board/Background:** Sen ska vi göra en board/background som kommer att användas för att skapa bakgrunden till spelet, alltså platformarna
3	**Sprite:** Här ska vi lägga till sprite i form av plattformar
4	**Rita ut bakgrunden:** Vi ska här se till att det som gjorts tidigare kan ritas ut i Fönstret som vi skapade innan.
5	**Skapa spelare:** Här ska vi lägga till en spelare i mappen
6	**Kollision mellan spelare och plattformar:** Här ska vi lägga till så att en spelare kan kollidera med en plattform
7	**Styrning av spelare:** Här ska vi lägga till att spelaren kan styras av spelaren. Det vi vill lägga till här är att gå fram och tillbaka samt att hoppa.
8	**Mål i slutet av banan:** Här vill vi lägga till ett mål i slutet av banan som senare i projektet kommer att bli en dörr eller portal som för en till nästa bana.
9	**Göra det spelbart:** Här ska vi se till att allt som har gjorts tidigare

	fungerar tillsammans och att spelet är spelbart
10	**Lägga till hinder:** Här ska vi lägga till hinder i banan
11	**Kollision med Hinder:** Här ska vi lägga till kollision mellan spelaren och hinder
12	**Spelarens död och respawn:** Här ska vi lägga till så att spelaren kan dö vid kollision av hinder och att spelaren då respawnar i början av banan
13	**Timer för att nå målet:** Här ska vi lägga till en timer som gör att om spelaren inte når målet inom tiden så kommer spelaren att behöva starta om spelet från början.
14	**Power ups 1 Öka tiden:** Här ska vi lägga till en power up som gör att tiden som krävs för att nå målet ökar
15	**Power ups 2 Öka hasigheten:** Här ska vi lägga till en power up som gör att spelaren hastighet ökar under en kort stund
16	**Power ups 3 Öka hopplängden:** Här ska vi lägga till en power up som ökar hopplängden för spelaren under en kort stund
17	**Poäng/Collectibles:** Här ska vi lägga till någon typ av poäng eller collectibles som spelaren ska kunna plocka upp i banan.
18	**Highscore list:** Här ska vi lägga till en highscore list så att spelaren ska kunna se hur många poäng samt vilken tid det tog att klara av en bana
19	**Lägga till fiender:** Här ska vi lägga till fiender som spelaren undvika för att målet
20	**Kollision med fiender:** Här ska vi lägga till kollisioner mellan spelaren och fiender samt kollision mellan spelaren och fiendens attack
21	**Lägga till nya banor:** Här ska vi lägga till nya banor med ökande svårhetsgrad som man ska komma till genom att gå igenom målet från tidigare
22	**Lägga till en boss:** Här ska vi lägga till en boss battle innan slutet av spelet
23	**Slut på spelet:** Här ska vi set till att allt som har gjorts tidigare fungerar och att spelet är spelbart
24	**Extra saker om vi har tid:** Här ska vi lägga till extra saker om vi har tid som designelement, extra power ups eller andra saker vi kommer på under projektets gång.

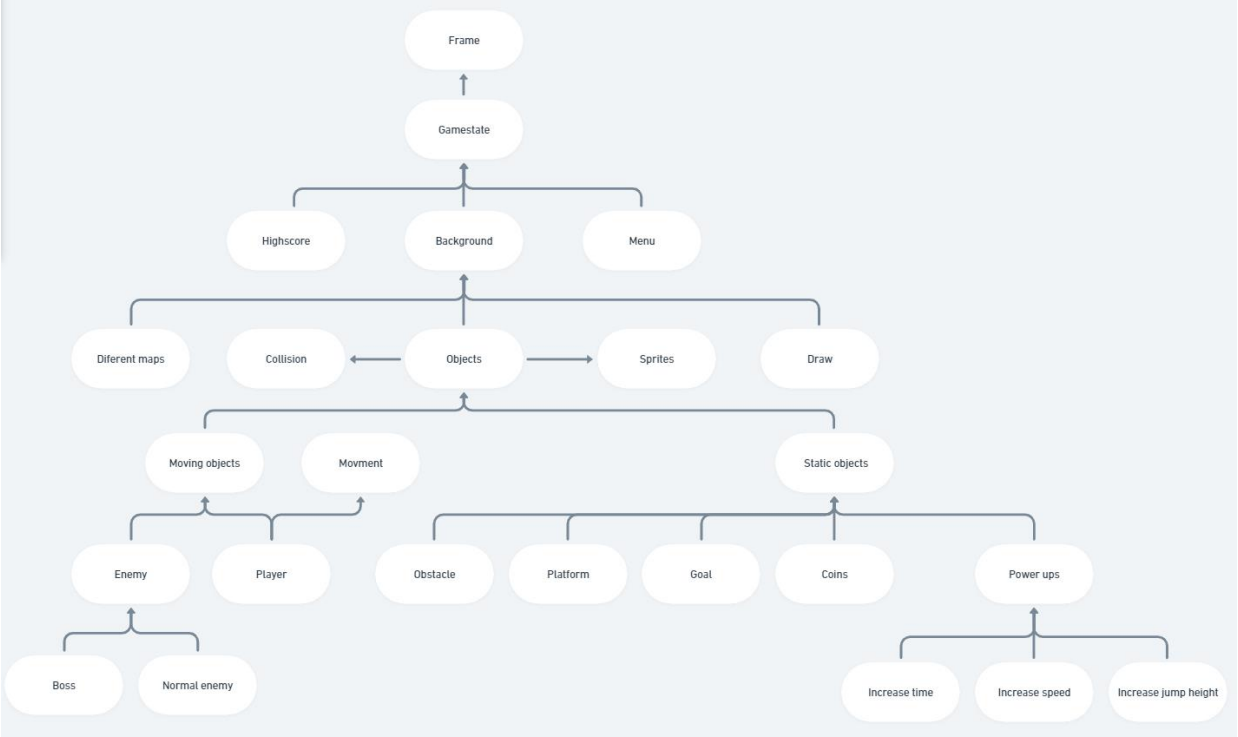
4. Övriga implementationsförberedelser

Vi tänkte att vi ska ha en background klass där allt som ska vara i bakgrunden ska skapas genom att kalla på andra klasser. Hur bakgrunden ska se ut kommer att skickas in som en fil som klassen kan läsa av och sedan skapa, för att göra det mer generellt och tillåta att man senare kan lägga till nya banor om man vill.

Vi ska göra en object interface som beskriver vad ett object är och en abstrakt klass som ska implementera alla funktioner som ett object ska ha och sedan ska en plattform-, fiende-, hinder- eller målklass extenda ifrån den och lägga till sina egna funktioner. Sen är det bakgrunds klassen som ska kunna skapa ett object genom att kalla på en createobject klass som sen kommer att få sina egenskaper från klasserna som extendar från den abstrakta object klassen och kallar på en sprite klass som laddar in rätt sprite.

Sen ska vi ha en generell kollisions klass som ska fungera för alla typer av kollisioner.

Vi tänkte göra en power ups interface som sedan dom olika power ups klassenar kan implementera.



5. Utveckling och samarbete

Projektrapport

Även om denna del inte ska lämnas in förrän projektet är klart, är det **viktigt att arbeta med den kontinuerligt** under projektets gång! Speciellt finns det några avsnitt där ni ska beskriva information som ni lätt kan glömma av när veckorna går (vilket flera tidigare studenter också har kommenterat).

Tänk på att ligga på lagom ambitionsnivå! En välskriven implementationsbeskrivning (avsnitt 6) hamnar normalt på **3-6 sidor** i det givna formatet och radavståndet, med ett par mindre UML-diagram och kanske ett par andra små illustrerande bilder vid behov. Hela projektrapporten (denna sista halva av dokumentet) behöver sällan mer än 10-12 sidor.

6. Implementationsbeskrivning

I det här avsnittet, och dess underavsnitt (6.x), beskriver ni olika aspekter av själva *implementationen*, under förutsättning att läsaren redan förstår vad *syftet* med projektet är (det har ju beskrivits tidigare).

Tänk er att någon ska vidareutveckla projektet, kanske genom att fixa eventuella buggar eller skapa utökningar. Då finns det en hel del som den personen kan behöva förstå så att man vet *var* funktionaliteten finns, *hur* den är uppdelad, och så vidare. Algoritmer och övergripande design passar också in i det här kapitlet.

Bilder, flödesdiagram, osv. är starkt rekommenderat!

Skapa gärna egna delkapitel för enskilda delar, om det underlättar. **Ta inte bort några rubriker!**

Även detta är en del av examinationen som visar att ni förstår vad ni gör!

6.1. Milstolpar

Ange för varje milstolpe om ni har genomfört den helt, delvis eller inte alls.

Detta är till för att labbhandledaren ska veta vilken funktionalitet man kan ”leta efter” i koden. Själva bedömningen beror *inte* på antalet milstolpar i sig, och inte heller på om man ”hann med” milstolparna eller inte!

6.2. Dokumentation för programstruktur, med UML-diagram

Programkod behöver dokumenteras för att man ska förstå hur den fungerar och hur allt hänger ihop. Vissa typer av dokumentation är direkt relaterad till ett enda fält, en enda metod eller en enda klass och placeras då lämpligast vid fältet, metoden eller klassen i en Javadoc-kommentar, *inte här*. Då är det både enklare att hitta dokumentationen och större chans att den faktiskt uppdateras när det sker ändringar. Annan dokumentation är mer övergripande och saknar en naturlig plats i koden. Då kan den placeras här. Det kan gälla till exempel:

- **Övergripande programstruktur**, t.ex. att man har implementerat ett spel som styrs av timer-tick n gånger per sekund där man vid varje sådant tick först

tar hand om input och gör eventuella förflyttningar för objekt av typ X, Y och Z, därefter kontrollerar kollisioner vilket sker med hjälp av klass W, och till slut uppdaterar skärmen.

- **Översikter över relaterade klasser** och hur de hänger ihop.
    - Här kan det ofta vara bra att använda **UML-diagram** för att illustrera – det finns även i betygskraven. Fundera då först på vilka grupper av klasser det är ni vill beskriva, och skapa sedan ett UML-diagram för varje grupp av klasser.
    - Notera att det sällan är särskilt användbart att lägga in hela projektet i ett enda gigantiskt diagram (vad är det då man fokuserar på?). Hitta intressanta delstrukturer och visa dem. Ni behöver normalt inte ha med fält eller metoder i diagrammen.
    - **Skriv sedan en textbeskrivning av vad det är ni illustrerar med UML-diagrammet**. Texten är den huvudsakliga dokumentationen medan UML-diagrammet hjälper läsaren att förstå texten och få en översikt.
    - IDEA kan hjälpa till att göra klassdiagram som ni sedan kan klippa och klistra in i dokumentet. Högerklicka i en editor och välj Diagrams / Show Diagram. Ni kan sedan lägga till och ta bort klasser med högerklicksmenyn. Exportera till bildfil med högerklick / Export to File.

I det här avsnittet har ni också en möjlighet att visa upp era kunskapen genom att diskutera koden i objektorienterade termer. Ni kan till exempel diskutera hur ni använder och har nytta av (åtminstone en del av) objekt/klasser, konstruktorer, typhierarkier, interface, ärvning, overriding, abstrakta klasser, subtypspolymorfism, och inkapsling (accessnivåer).

Labbhandledaren och examinatorn kommer bland annat att använda dokumentationen i det här avsnittet för att förstå programmet vid bedömningen. Ni kan också tänka er att ni själva ska vidareutveckla projektet efter att en annan grupp har utvecklat grunden. Vad skulle ni själva vilja veta i det läget?

När ni pratar om klasser och metoder ska deras namn anges tydligt (inte bara ”vår timerklass” eller ”utritningsmetoden”).

Framhäv gärna det ni själva tycker är **bra/intressanta lösningar** eller annat som handledaren borde titta på vid den senare genomgången av programkoden.

Vi räknar med att de flesta projekt behöver runt **3-6 sidor** för det här avsnittet.

7. Användarmanual

När ni har implementerat ett program krävs det också en manual som förklarar hur programmet fungerar. Ni ska beskriva programmet tillräckligt mycket för att en labbhandledare själv ska kunna *starta det, testa det och förstå hur det används*.

Inkludera flera (**minst 3**) **skärmdumpar** som visar hur programmet ser ut! Dessa ska vara ”inline” i detta dokument, inte i separata filer. Sikta på att visa de relevanta delarna av programmet för någon som *inte* startar det själv, utan bara läser manualen!

(Glöm inte att ta bort våra instruktioner, och exportera till PDF-format med korrekt namn enligt websidorna, innan ni skickar in!)