

Projektbeskrivning

<Escape the Dungeon>

2024-05-15

Projektmedlemmar:

Albaraa Alsaleh <albal207@student.liu.se>
Mohannad Alaya <mohal172@student.liu.se>

Handledare:

Morgan Nordberg <morno368@student.liu.se>

Innehåll

1. Introduktion till projektet.....	2
2. Ytterligare bakgrundsinformation.....	2
3. Milstolpar.....	2
4. Övriga implementationsförberedelser.....	4
5. Utveckling och samarbete.....	4
6. Implementationsbeskrivning.....	6
6.1. Milstolpar.....	6
6.2. Dokumentation för programstruktur, med UML-diagram.....	6
7. Användarmanual.....	7

Projektplan

1. Introduktion till projektet

Vi planerar att utveckla ett plattformsspel med titeln "Escape the Dungeon". Spelet handlar om en person som måste rymma från en grotta. Det finns dock en del hinder som spelaren måste gå igenom. Spelaren måste ta upp nyckeln och sedan försöka hitta vägen ut. Efter ett antal utmaningar vinner spelaren spelet.

Spelet kommer att ha en enkel design med en grottlik bakgrund och objekt som ska representera djur och monster. Spelaren kommer att styra huvudspelaren med piltangenterna.

Vi har inspirerats av klassiska plattformsspel som "Shiren the Wanderer: The Tower of Fortune and the Dice of Fate", men vi planerar att lägga till en egen twist på spelet för att göra det mer unikt. Vi hoppas att detta spel kommer att vara roligt och utmanande för spelare.

2. Ytterligare bakgrundsinformation

Vårt spel kräver mycket objektorienterad programmering. Man behöver hålla koll på olika klasser och måste hitta ett smart sätt att koppla ihop dem. Det kräver även en liten mängd animeringar, men vi satsar därför på att vara originella. När det kommer till programmering så bör vi lära oss mer om plattform video games och förstå deras funktioner bättre.

3. Milstolpar

#	Beskrivning
1	Till att börja med behöver vi en karta. Detta gör vi genom att skapa grafiskt fönster. Vi ska rita vår första design för bakgrunden som kommer vara enkelt och definitivt utvecklas med tiden.
2	I andra steget planerar vi att utveckla miljön mer. Spelaren ska kunna förflytta sig inom kartan. Dessa behöver en del algoritmer och klass programmering för att kunna utveckla det. Här ska vi även börja försöka introducera principen av mål, hursomhelst spelet bör utvecklas lite mer först.
3	Nu är vi klara med kartan, och då är det dags att ha spelfigurer. Vi ska fokusera huvudsakligen på huvudspelare, men vi ska även börja med möjliga fiender, hinder och föremål. När det kommer till utseende, så kommer det att vara mycket enkelt och utvecklas mer i nästa steg.
4	Här kommer vi till rörelsen av spelaren. Rörelsen av spelaren är väldigt enkel, han ska kunna hoppa, samt gå vänster och höger.
5	Här introduceras fysiken av olika föremål och hinder och hur de kan påverka huvudspelaren, med andra ord kollision. Om ett monster rör spelaren bör

spelaren dö. Samtidigt ska ett hinder kunna stoppa spelaren.

6 Allmän test!

7 Nu programmeras nyckel och dörr logiken. Detta innebär att vi behöver lägga olika funktioner för att kunna överföras mellan kartorna till en annan.

8 Nu utvecklas olika sorters fiender och deras rörelse. Ett monster ska kunna döda spelar. Vi funderar på att lägga fladdermus, om spelaren rör under medan han försöker rymma, då saktas spelaren ner. Olika bossar rör sig annorlunda och har olika svårighetsgrader och funktioner.

9 Bättre visualisering.

10 En AI som påverkar fiendens funktioner.

11 Ett fungerande spel.

12

13

14

15

16

17

18

19

20

21

22

23

...

4. Övriga implementationsförberedelser

Planen är att börja med att skapa filen "Escape" med en main-klass. Därefter kommer en bakgrundsklass att läggas till i samma fil. I denna bakgrundsklass definieras alla nödvändiga klasser för att visa bakgrunden. I bakgrundsfilerna skapas först en fil för startmenyn som innehåller flera klasser. För att skapa kartan används Java Swing för att skapa ett grafiskt fönster.

En annan klass krävs för kartan, i den implementeras fysiken. Tanken är att en spelaren ska kunna förflytta sig mellan olika rum, dessa rum ska vara kopplade ihop.

Vi tänkte skapa flera nya klasser för monster som rör sig på olika sätt och har olika svårighetsgrader och funktioner genom att implementera AI-funktioner för fiendens rörelse.

För att implementera slå-funktionen behöver vi först bestämma vilken knapp eller tangent som ska aktivera den. Därefter behöver vi lägga till en ny objekttyp för spelarens föremål (yxan) och dess rörelse. När spelaren trycker på knappen eller tangenten, då bör det döda mönstren om spelaren är tillräckligt nära.

För att skapa spelfigurer tänkte vi använda Java Swing och Image-klasser för att skapa huvudspelaren, fiender, hinder och föremål. Vi behöver sedan rita dessa figurer på kartan.

För att implementera fysiken av olika föremål och hinder och deras effekter på huvudspelaren tänkte vi skapa kollisionssystem. Genom att använda Java Swing för att skapa kollisionzoner för objekten och sedan använda en kollisionsdetektion algoritm för att upptäcka kollisioner mellan objekten.

Avslutningsvis tänkte vi lägga en menyskärm som låter användaren välja att starta spelet eller avsluta det, men även lägga till ett game over-skärm som visas när huvudspelaren är död.

5. Utveckling och samarbete

Vi delar samma ambition och siktar på att slutföra de första sex stegen i projektet senast den 13 april. För att uppnå detta är kommunikation viktigt för oss. Vi håller regelbundna möten för att diskutera framsteg, potentiella vägsärrar och nästa steg på vägen. Genom öppen och tydlig kommunikation säkerställer vi att alla i teamet arbetar mot samma mål och att eventuella frågor hanteras effektivt och snabbt.

Projektrapport

6. Implementationsbeskrivning

I implementeringen av spelet använder vi olika algoritmer och metoder för att skapa en välstrukturerad och effektiv kodbas. Tillståndslogik (Enum) tillämpas för att hantera spelets olika tillstånd på ett organiserat sätt. Genom att använda det här mönstret kan vi enkelt växla mellan olika tillstånd, som när spelet startar, när spelaren spelar, när spelaren vinner eller förlorar eller när spelet slutar.

Vi använder också olika tillståndsvariabler för att spåra spelets tillstånd vid varje given tidpunkt. Dessa variabler styr om spelet pågår, om spelaren vinner eller förlorar, om spelaren attackerar eller om fienderna är aktiva.

Flyttals Matematik används för att hantera spelarens och fiendens rörelser. Detta gör att du kan exakt och effektivt beräkna positioner och kollisioner, vilket är nödvändigt för att spelet ska fungera smidigt och korrekt.

6.1. Milstolpar

Vi har totalt 11 milstolpar och vi har genomfört mestadels majoriteten av milstolparna! Det var tack vare ett härligt samarbete och bra kommunikation mellan oss.

Milstolpe 1: Det var att skapa en bakgrund, milstolpen genomfördes.

Milstolpe 2: Det var att beskriva funktioner för spelarens rörelse, milstolpen genomfördes.

Milstolpe 3: Det var att avsluta kartan och påbörja med fienderna, milstolpen genomfördes.

Milstolpe 4: Det var att avsluta spelarens rörelse och börja med hoppfunktionen, milstolpen genomfördes.

Milstolpe 5: Det var att introducera kollision, milstolpen genomfördes.

Milstolpe 6: Det var en allmän test, milstolpen genomfördes.

Milstolpe 7: Det var att implementera dörr och nyckellogiken, milstolpen genomfördes.

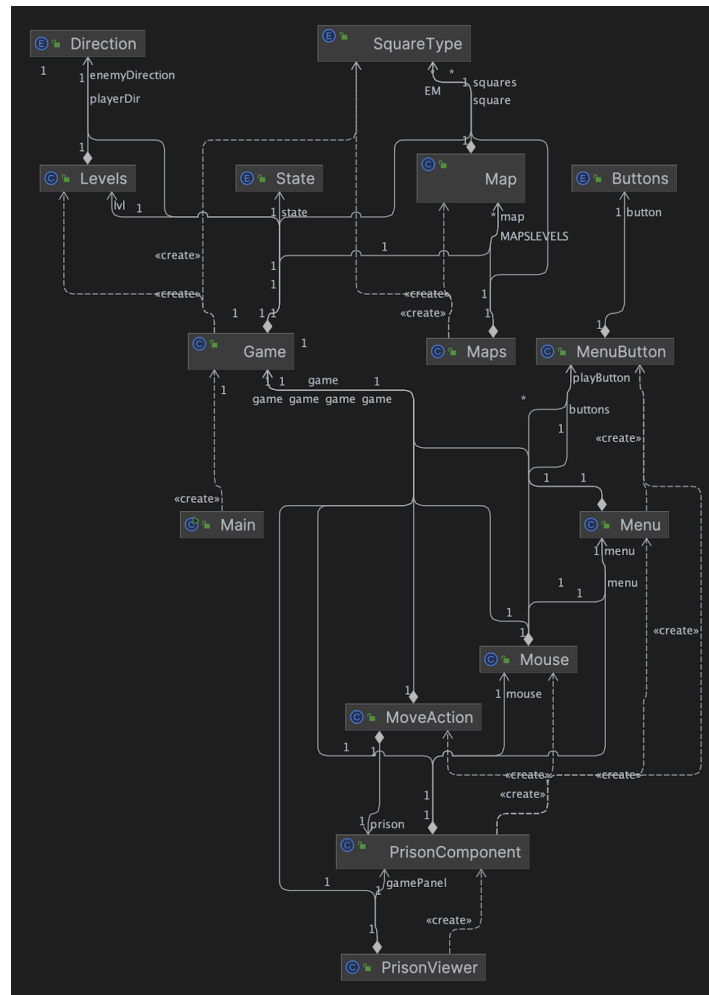
Milstolpe 8: Det var att utveckla fiendens logik, det var planerat att låta fladdermösset sakta ner spelaren, hursomhelst vi har bestämt att låta fladdermösset döda spelaren istället. Milstolpen är delvis genomförd.

Milstolpe 9: Det var att ha bättre visualisering, milstolpen genomfördes

Milstolpe 10: Det var att ha en AI som påverkar fiendens rörelse, milstolpen genomfördes.

Milstolpe 11: Sista milstolpen var att ha ett fungerande spel, milstolpen genomfördes.

6.2. Dokumentation för programstruktur, med UML-diagram



Figur 1. UML-diagram

Figur 1 visar ett strukturdiagram av spelets kod, som ofta kallas för UML-diagram. Spelet består bland annat av följande klasser: Game, MovieAction, PrisonComponent, PrisonViewer, Mouse, Menu, MenuButton, Buttons, Map, Maps, SquareType, State, Levels, Direction och Main.

Game-klassen ansvarar för att hantera spelets logik och tillstånd. Här finns funktionen “tick” som körs igenom spelet, och för spelet framåt! Funktionen skickas till klassen “PrisonViewer”, där det kallas hela tiden genom en “while-loop”. Andra viktiga funktioner är “hasCollision” som kallas i “tick” och ansvarar för att se om spelaren har kolliderat med ett objekt eller ej.

PrisonViewer-klassen tar emot “tick-funktionen” och kör spelet, vilket görs med hjälp av Java klassen “JPanel”. Vi har även implementerat funktioner som hanterar FPS med hjälp av Java klassen “Runnable”.

Direction-enum har olika rörelsetyper såsom vänster och höger. Dessa används sedan i “MovieAction-klassen” för att ändra spelarens rörelse.

MovieAction-klassen ansvarar för tangentbordets inmatningar. Detta handlar främst om vänster- och högerpilen. Samt Mellanslagstangent och “B”-knappen. Här implementeras "KeyListener", vilket är en klass som redan finns i Java. “KeyListener-klassen” är intresserad av att bearbeta en tangentbordshändelse och har egna funktioner som används här, såsom “keyPressed” och “keyReleased”. I dessa

funktioner kallar vi på "setPlayerDir" som befinner sig i "Game-klassen", och påverkar spelets rörelse med hjälp av "Direction-enum".

PrisonComponent-klassen ansvarar för spelets ritning. Här ritas spelet, baserat på "Direction-enum", vilket ändrar sig i "MovieAction-klassen" och baserat på det ritas spelarens rörelse. Vi har även en annan "enum" som kallas för "SquareType". I "Maps-klassen" bestäms vilken typ varje objekt ska ha, vilket kännetecknas i "PrisonComponent-klassen". Detta sker eftersom vi skickar det skapade spelet "game" från "Main-klassen" hit. "PrisonComponent-klassen" sträcker sig "extends" till klassen "JComponent" som har olika funktioner som ritar spelet.

State-enum innehåller olika tillstånd för spelet, som pågående spel, meny-tillstånd och liknande.

Mouse-klassen ansvarar för datormus rörelset. Här lyssnar klassen på datormusens rörelse med hjälp av klasserna "MouseListener" och "MouseMotionListener". Därefter skickas avlysningar till "Game-klassen" genom att ändra på "State" vilket ändrar spelet från menyn till pågående och vice versa.

Buttons-enum har olika typer av knappar som menyn kan rita.

MenuButton-klassen ansvarar för att sätta bilderna på olika knappar som "Menu-klassen" ska rita sen.

Menu-klassen ansvarar för ritning av menyn. Den använder funktionerna som befinner sig i "MenuButton-klassen" och med hjälp av en "Buttons-enum" så kan det avgöra vilken bild som ska ritas.

SquareType-enum har olika typer som varje block ska ha.

Map-klassen ansvarar för att skapa en lista som sedan ska användas i "Maps-klassen" för att konstruera en karta.

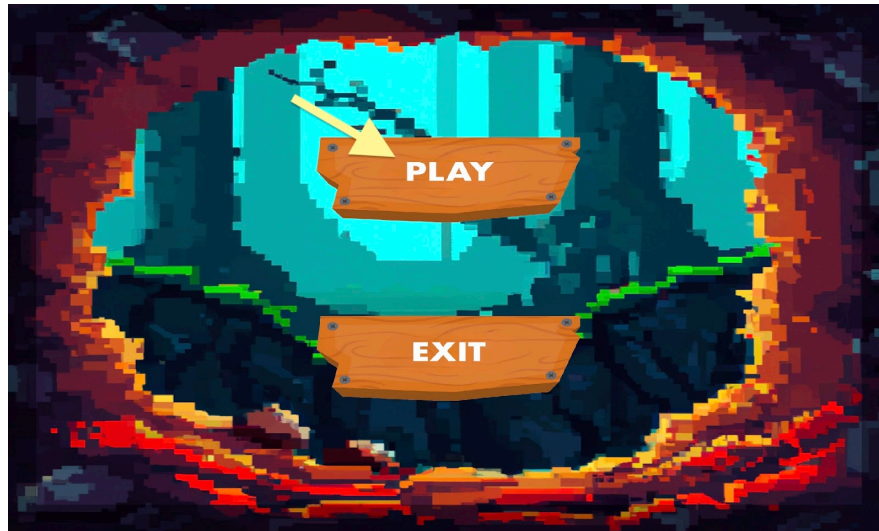
Maps-klassen ansvarar för att konstruera olika kartor. Dessa kartor används sedan i klassen "PrisonComponent" för att rita bakgrunden. Kartorna skapas med hjälp av "SquareType-enum", som bestämmer vilken typ varje block ska ha. Denna process sker i Game-klassen.

Levels-enum har olika nivåer som sedan används i spelet för att hantera nivåskiftningarna.

Main-klassen ansvarar för skapandet av spelet med hjälp av "Game-klassen". Det går att bestämma vilken nivå som ska påbörjas genom att modifiera de värden som skickas till "Game"-konstruktorn.

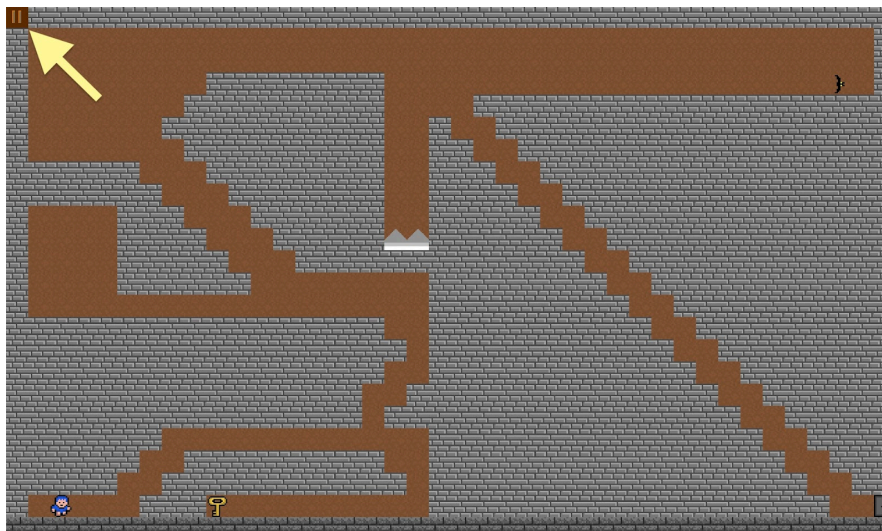
7. Användarmanual

Du börjar spelet genom att trycka på knappen “PLAY”, som visas på figur 2!



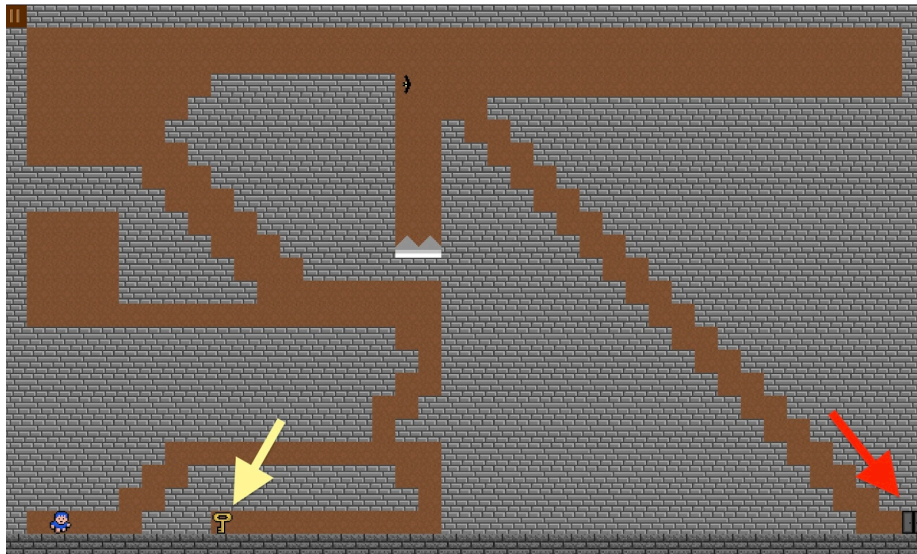
Figur 2. Meny-skärmen

Därefter överförs du till spelet. Använd höger- och vänsterpilarna för att flytta spelaren. Tryck på mellanslagstangenten för att hoppa. För att pausa spelet, tryck på knappen längst upp till vänster på skärmen.



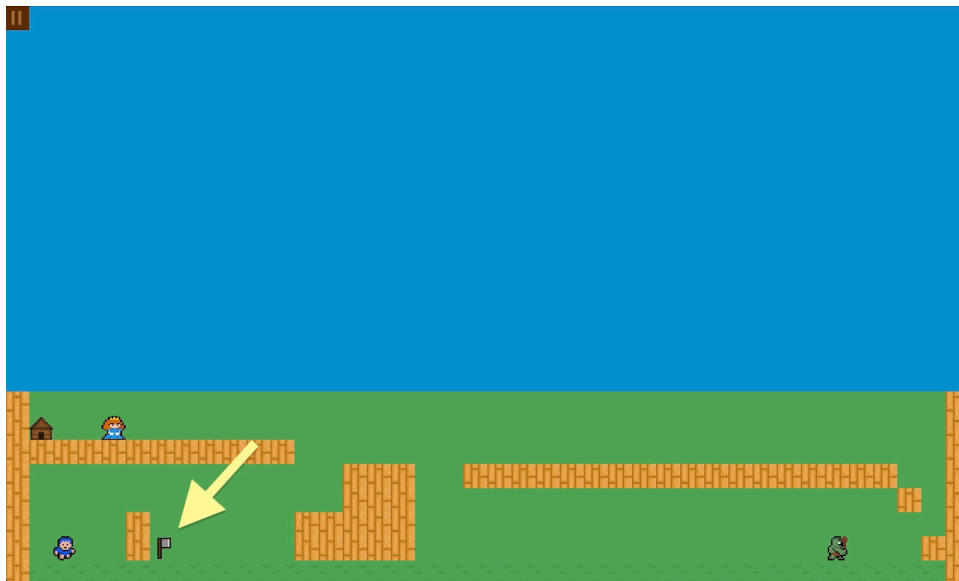
Figur 3. Pausknappen.

Målet är att ta upp nyckeln och sedan gå till dörren utan att dö (nyckeln och dörren visas i figur 4). När du har uppfyllt detta överförs du till nästa nivå. Samma princip gäller på de resterande nivåerna, förutom den sista. Därför är strategin som du bör sträva efter att undvika alla fiender och sikta på nyckeln. När du har tagit upp nyckeln ska du gå mot dörren.



Figur 4. Den gula pilen siktar på nyckeln, medan den röda pilen siktar på dörren.

På den sista nivån finns ingen nyckel, men ett monster måste besegras för att avsluta nivån. För att göra detta, plocka upp yxan och tryck på "B"-knappen för att attackera monstret (yxan visas i figur 5). När monstret är besegrat, gå till det bruna huset för att vinna spelet.



Figur 5. Yxan.