Projektbeskrivning

Plattformsspel

2022-03-22

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.

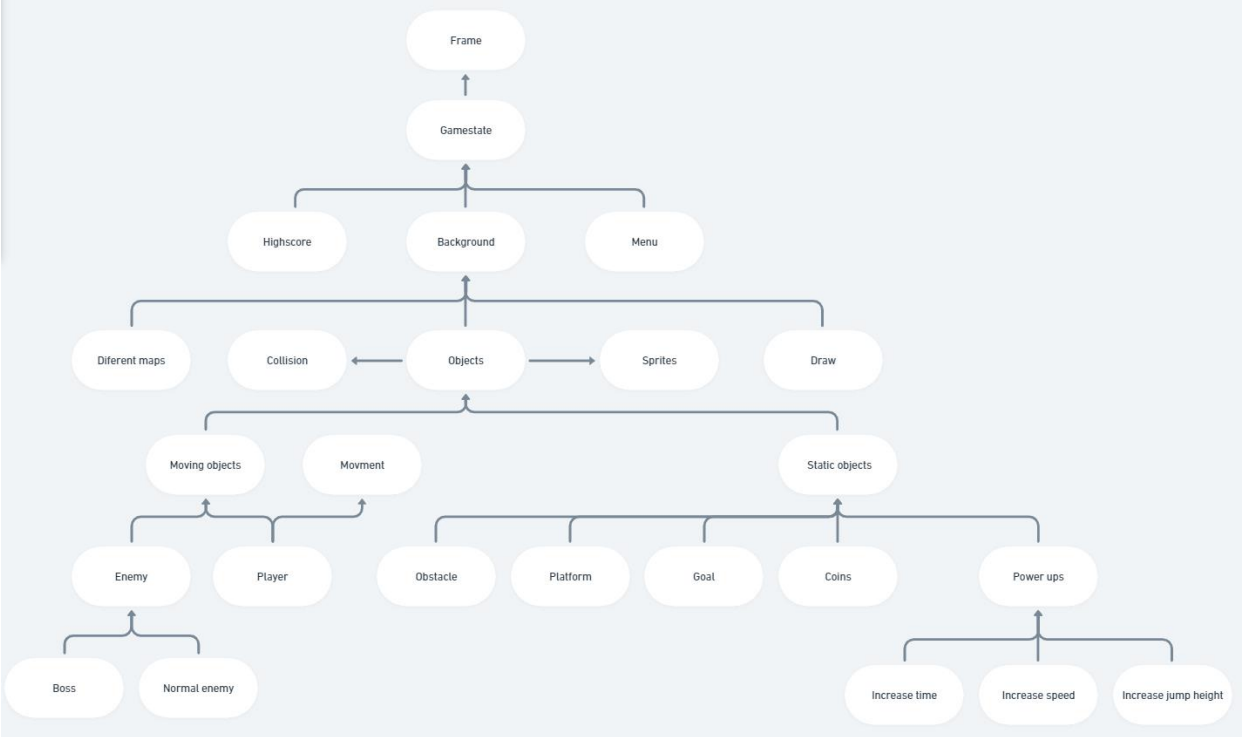
4. Övriga implementationsförberedelser

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

PROJEKT RAPPORT

6. Implementationsbeskrivning

6.1. Milstolpar

	Helt	Delvis	Inte Alls
Fönster	X		
Board/Background	X		
Sprite	X		
Rita ut bakgrunden	X		
Skapa spelare			
Kollision mellan spelare och platformar	X		
Styrning av spelare	X		
Mål i slutet av banan	X		
Göra det spelbart	X		
Lägga till hinder	X		
Kollision med hinder	X		
Spelarens död och respawn	X		
Timer för att nå målet	X		
Power ups 1 öka tiden	X		
Power ups 2 öka hastigheten	X		
Power ups 3 öka hopplängden	X		
Poäng/Collectibles	X		
Highscore list	X		
Lägga till fiender			X
Kollision med fiender			X
Lägga till nya banor	X		
Lägga till en boss	X		
Slut på spelet	X		
Extra saker om vi har tid		X	

6.2. Dokumentation för programstruktur, med UML-diagram

Övergripande porgramstruktur

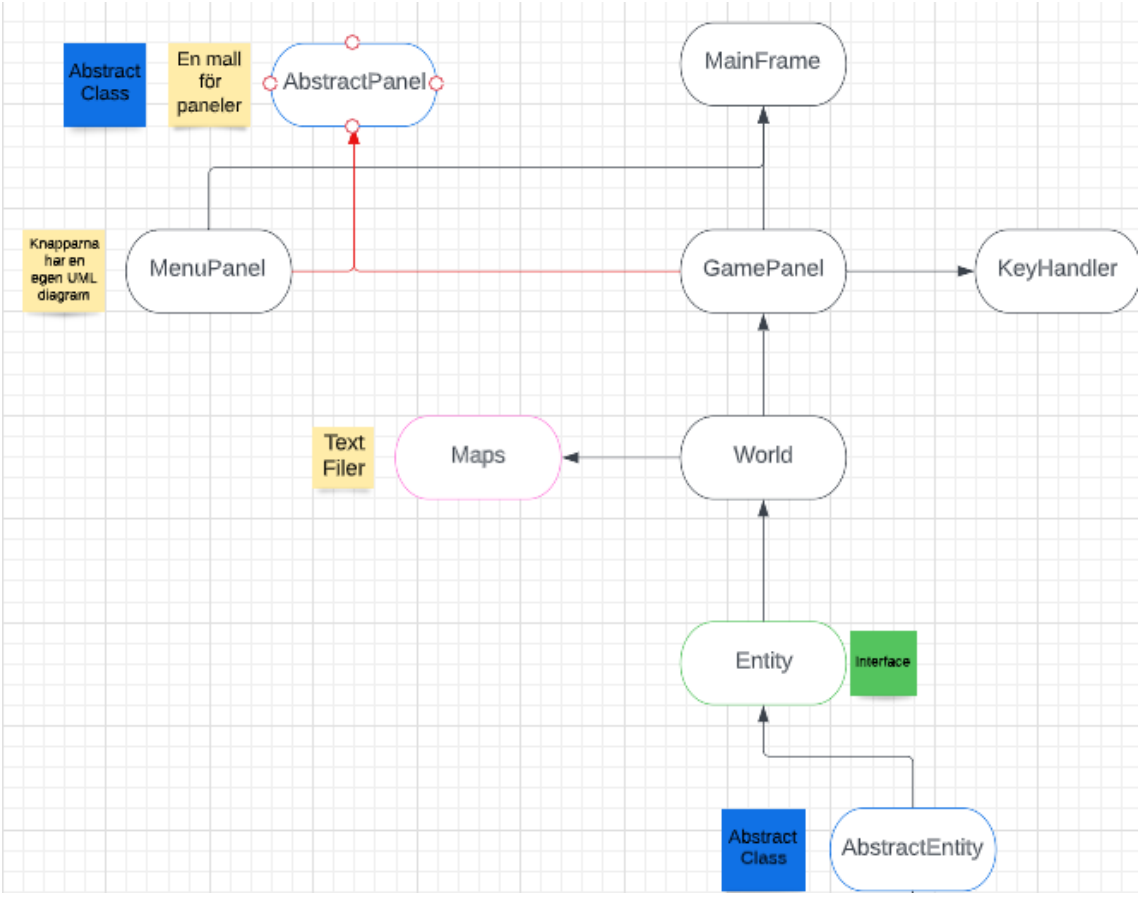
- i klassen GamePanel har vi en tick funktion run() som kontrollerar först om spelet är slut eller inte, sedan kallar på updateGameKey() som hanterar förlyttningen av spelaren. Den kallar också på updateWorld, en funktion i klassen world som hanterar timers, counters och uppdaterar alla objekt. Till slut uppdaterar den skärmen med repaint()

- Spelaren har en kollision box av datatypen Rectangle. I klassen Player har vi funktionen tryCollision() som kontrollerar om spelarens rektangel korsar en entity rektangel från entitylistan. Den kallar sedan på applyCollision() i klassen World som ser till att rätt kod utförs beroende på entitytypen.
- Vi använder Enumap för att ladda in bilder och koppla den till en entitytyp.
- Från MyButton skapar vi knappar och laddar in rätt bild från en map i klassen.

Översikter över relaterade klasser

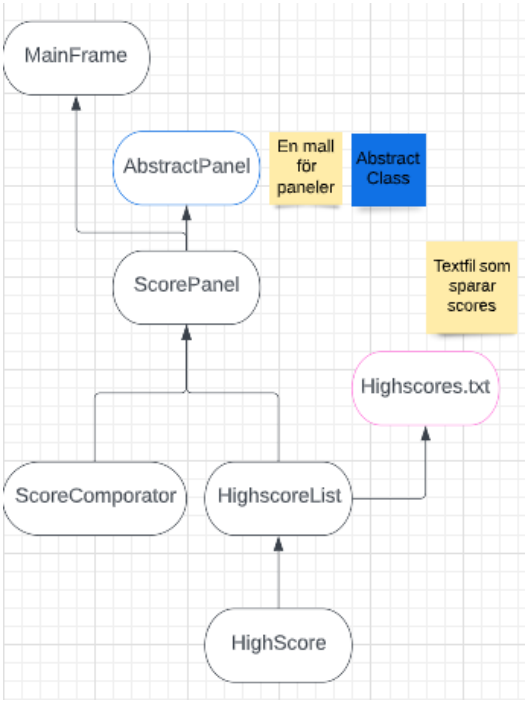
Diagrammen ger bild över hur huvudklasserna hänger ihop

- Diagrammen illustrerar klasserna från de minsta (nedersta) Entitys till den översta klassen MainFrame som visar vårt spel.

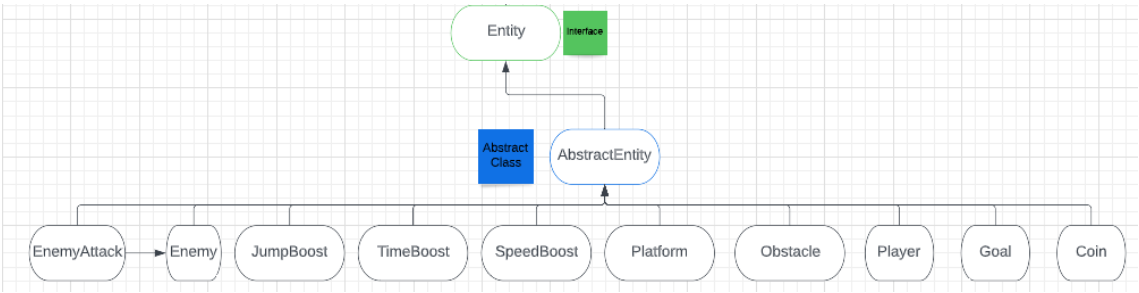


- World använder sig av Map textfiler och läser in sympol som är kopplade till varsin entity. På så sätt laddar vi in rätt objekt/bild.
- På GamePanel ritar vi ut vår spelvärld. Den använder också en keyhandler som hanterar rörelsen av spelaren, spelomstart eller spel avslut.
- Andra Paneler som kan visas på fönstret (MainFrame) är:

- MenuPanel: Extendar från vår egen AbstractPanel som är en mall för hur alla paneler ska se ut. Menu panel innehåller knappar som vi förklarar mer om i en annan del.

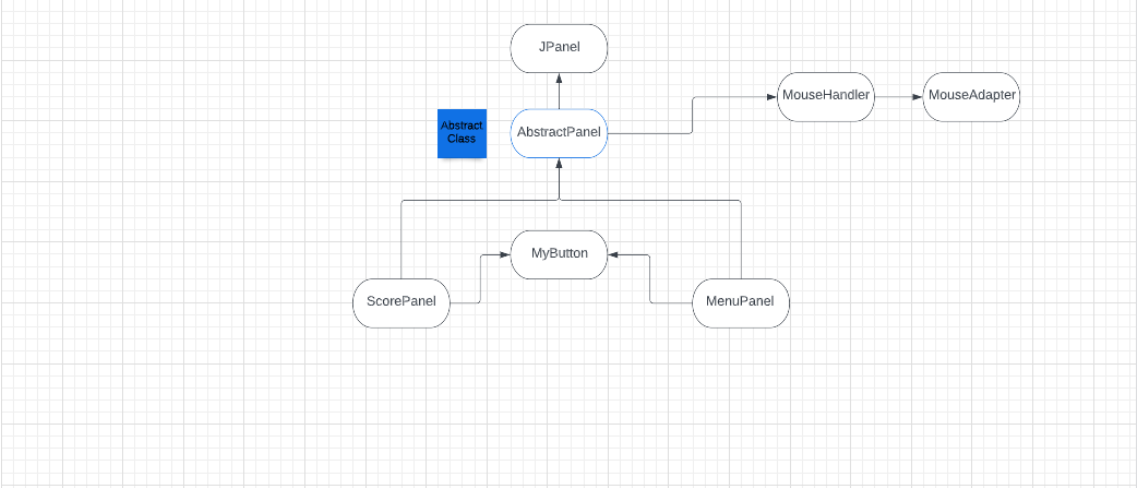


- ScorePanel: kopplad på samma sätt som MenuPanel. Utöver knappar visar ScorePanel gamla scores och kan sortera de efter time, coins eller deaths med hjälp av klassen ScoreComporator. Nya scores (Highscore objekt) läggs till en HighscoreList objekt, vilket sen sparas till en textfil som heter Highscores. Scores från textfilen kan laddas in till en HighscoreList objekt så att vi hämta listan, lägga till en ny score och sen lägga tillbaka de i textfilen.



- I World skapas alla olika typer av Entitys, vilka extendar från Abstract-Entity som i sin tur implementerar Interface Entity.

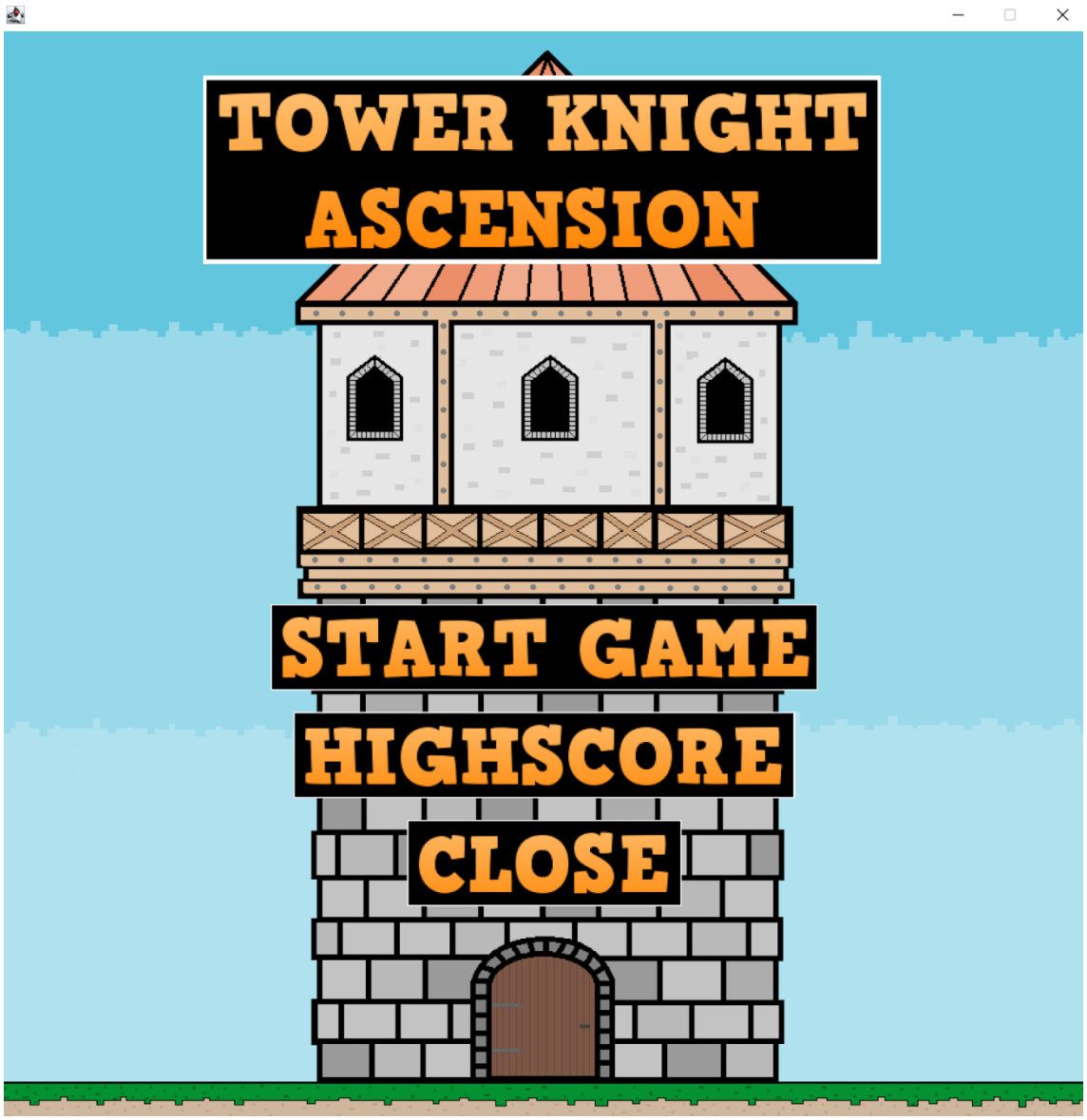
Diagram över hur mus och knapp klasserna hänger ihop



- Här illustreras hur det funkar att trycka med musen på en knapp som utför en handling.
- Vi använder oss av MouseHandler som extendar från en befintlig klass, MouseAdapter.
- MouseAdapter har två funktioner, mouseClicked() och mouseMoved(), som vi overridar och implementerar i MouseHandler. mouseClicked() aktiveras när någon trycker med musens Button1, mouseMoved() aktiveras när musen är placerad över en knapp.
- MouseHandler läggs till som Listener/MotionListener för AbstractPanel, om musen korsar eller klickar på en knapp kallar den på mouseMoved() respektiv mouseClicked() från den abstrakta klassen AbstractPanel.
- AbstractPanel funktioner ärvs och implementeras i klasser som extendar från den, alltså MenuPanel och ScorePanel. Där de utför lämliga funktioner.
- MyButton är klassen som vi skapar alla knapp objekt från. Den tar hand om att rita ut knappen och markera när musen är placerad på knappen. Klassen har en Enumap med alla olika bilder på knappar, den används för att ladda in rätt bild beroende på vilken enum typ knappen blir tilldelad när den skapas. MyButton tar in som första argument en Buttons Enum Typ. Konstruktorn kallar också på funktionen initBounds() som i sin tur- som skapar gränser till knappen (rektangel) med värdena (x, y, width och height) som skickas med och initieras i konstruktorn.

7. Användarmanual

Huvudmenyn:



1. Start game: Startar en ny game direkt.
2. HighScore: Visar spelaren tidigare försök (där man klarade av spelet), det går att sortera efter kortaste tid, minst antal deaths eller mest coins.

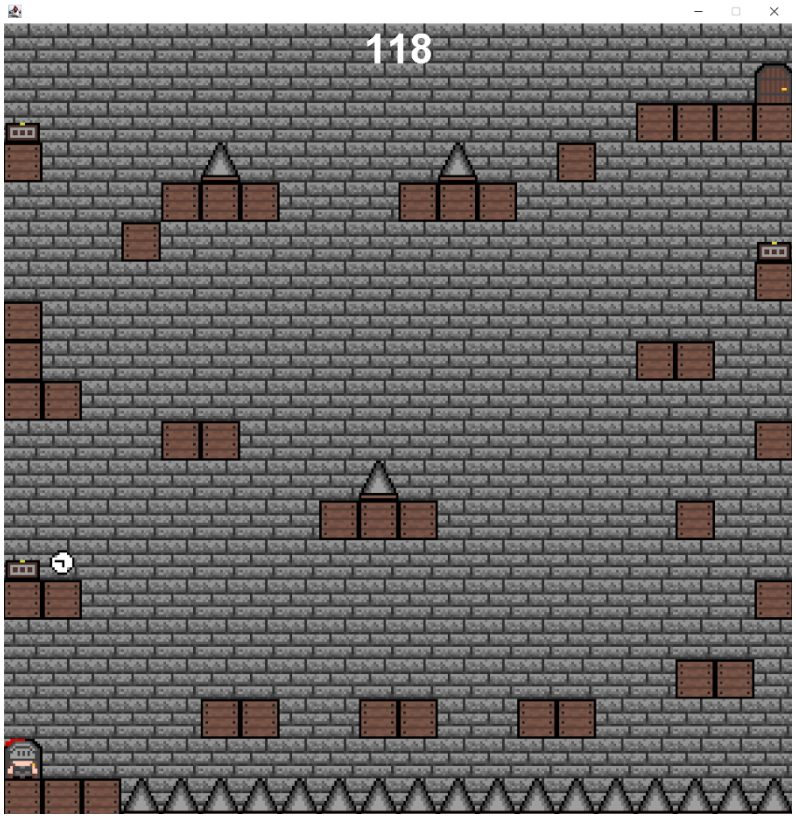
3. Close: Stänger av spelet helt.

Kontroll

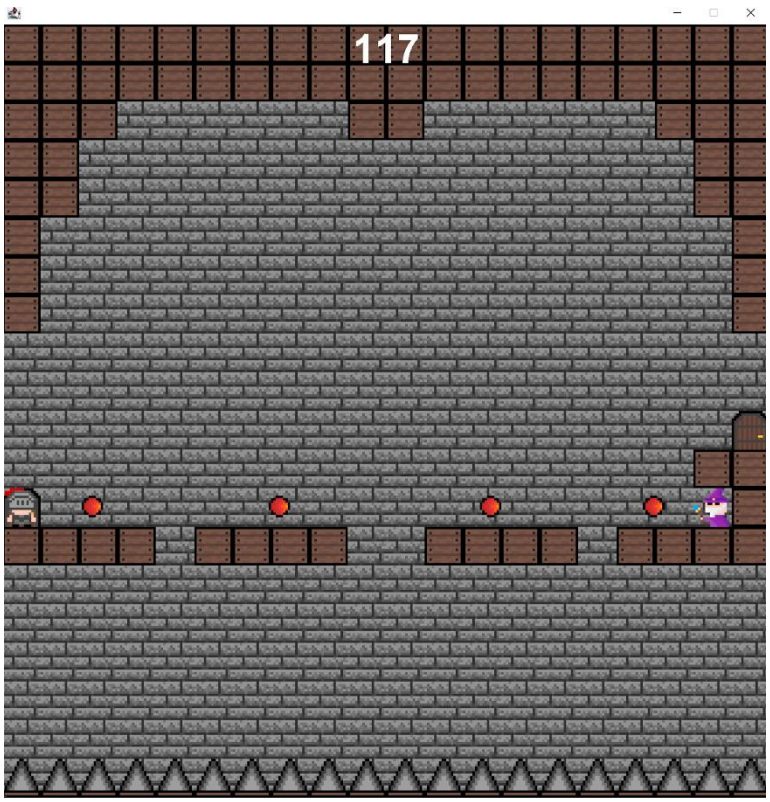
Spelet har väldigt enkel tangentbordstyrning. Man kan röra spelaren fram och tillbaka med knapparna ***A*** och ***D***. Man hoppar upp med ***Space***. Den är lätt att lära sig men kräver erfarenhet för att bli bra på.

Spelaren måste ta sig till nästa bana genom att interakta med en portal (dörr).

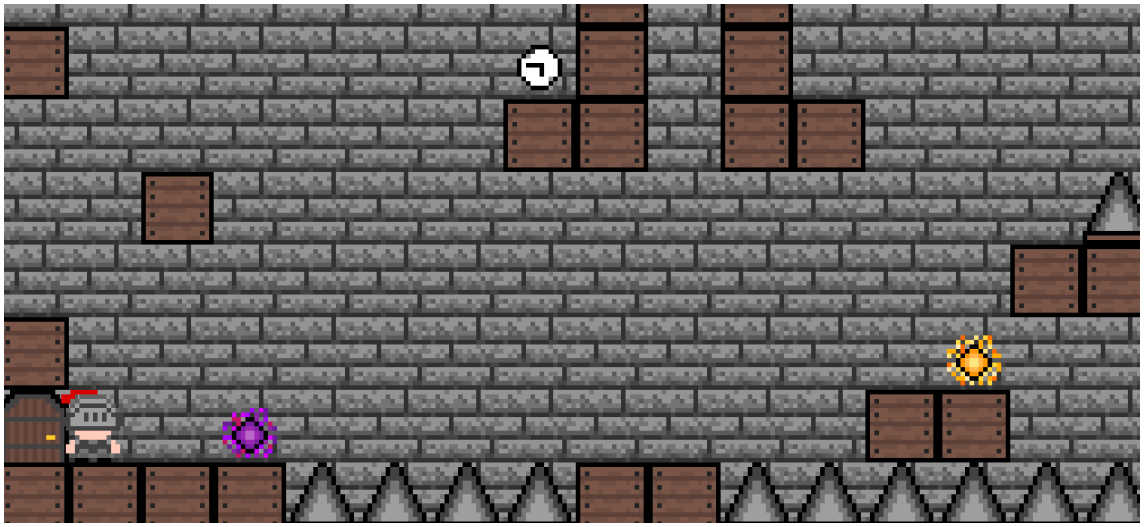
För att komma till portalen ska spelaren klara av massa hinder och ta sig upp för plattformarna.



När man har tagit sig till sista banan, möter man en boss som skjuter attack emot dig, som du måste ta dig förbi för att klara av spelet.



Powerupps/coins



Man plockar upp powerups och coins genom att kollidera med de. Coins räknas till highscore.

3 olika powersups:

- Time (Klockan): Tids tillägg för den banan du spelar på.
- Hopp (Gula klotet): hopp höjden får en kortvarig boost som är bra att uttnyttja i vissa delar av mappen för att klara av den snabbare eller komma åt en coin.

- Hastighet (lila klotet): spelarens hastighet får en kortvarig boost.