Requirements and Analysis Document for Hive

Hanna Adenholm, Lisa Qwinth, Stina Hansson

2021-10-24

Version 1

1. Introduktion

Hive är ett brädspel för två spelare där hexagon-formade pjäser som föreställer insekter utgör brädet. Målet med detta projekt är att omvandla brädspelet Hive till en mobilapplikation, och därigenom göra det mer tillgängligt för folk att spela. Spelet som har tagits fram består av fem olika pjäser, biet, gräshoppan, myran, spindeln och skalbaggen. Dessa pjäser rör sig på olika sätt på brädet. Biet kan, likt kungen i schack, endast gå ett steg i valfri riktning. Gräshoppan kan hoppa över brädet, men endast i raka linjer. Myran kan gå hur många steg som helst runt brädet. Spindeln är som myran, men kan endast gå tre steg. Skalbaggen är som biet, men kan också hoppa upp på andra pjäser. Målet med spelet är att med hjälp av pjäserna omringa sin motståndares bi, men samtidigt se upp så att ens eget bi inte blir omringat. Spelet är gjort för att spelas på samma mobil, där spelarna får turas om att göra sitt drag.

Alla regler: https://www.ultraboardgames.com/hive/game-rules.php

Applikationen Hive är ett alternativ till de som föredrar att spela mobilspel framför brädspel, eller för de som inte vill lägga pengar eller har råd med ett brädspel. Applikationen har fördelar som det fysiska brädspelet inte har i och med att man alltid har tillgång till det via mobilen och man inte är beroende av en slät yta att lägga ut pjäserna på.

1.1 Definitioner

- **Activity:** En skärm som användaren kan interagera med där gränssnittet visas
- **GUI:** Grafiskt användargränssnitt
- **Hashmap:** En lista av par, där paret består av en nyckel och ett värde
- **Enumeration:** En sorts klass i java som håller en grupp av konstanter, vilket är variabler som inte kommer förändras.
- **Fragment:** En del av gränssnittet som läggs till på en activity
- **Java:** Ett objektorienterat programspråk
- **JUnit:** Ett ramverk för enhetstestning, för Java
- **LiveData:** En klass som finns i Androids bibliotek som notifierar de klasser som observerar viss data så fort denna data ändras
- **User Story:** En förklaring av en funktion i en applikation, som är skriven från perspektivet av en användare.

2. Krav

2.1 User Stories

De user stories som projektet är byggt på listas nedan. Alla user stories är implementerade förutom STK006.

User Story

ID : STK001
Namn: Rutnät med pjäser

Beskrivning

Som användare vill jag kunna se rutnätet med de spelade pjäserna så att jag kan spela spelet.

Bekräftelse

Funktionell

- Kan jag se rutnätet med de spelade pjäserna?
- Kan jag se alla pjäser på en gång?

User Story

ID : STK002
Namn: Möjliga drag

Beskrivning

Som användare vill jag kunna se möjliga drag för en specifik pjäs så att jag kan veta var den kan flyttas.

Bekräftelse

Funktionell

- Kan jag klicka på en spelad pjäs och se de möjliga dragen markerade på rutnätet?
- Är den valda pjäsen markerad?
- Kan jag ändra mig och klicka på en annan pjäs?

User Story

ID : STK003
Namn: Ospelade pjäser

Beskrivning

Som användare vill jag kunna se de ospelade pjäserna, så att jag kan veta hur jag ska göra mitt drag.

Bekräftelse

Funktionell

- Kan jag se alla pjäser som inte är spelade än?
- Kan jag se hur många som är kvar av varje typ?

User Story

ID : STK004
Namn: Spela en ny pjäs

Beskrivning

Som användare vill jag kunna lägga ut en ny pjäs, så att jag kan spela spelet.

Bekräftelse

Funktionell

- Kan jag klicka på en pjäs i min hand och se de möjliga dragen markerade?
- Kan jag klicka på en av de möjliga dragen och se att min pjäs flyttas dit?

User Story

ID : STK005
Namn: Nuvarande spelare

Beskrivning

Som användare vill jag kunna se vems tur det är, så att jag vet när det är min tur.

Bekräftelse

Funktionell

- Ändras färgtemat på skärmen beroende på vems tur det är?
- Byts spelarhanden till den nuvarande spelarens?
- Står det vems tur det är överst på skärmen?

User Story

ID : STK006
Namn: Regelfönster

Beskrivning

Som användare vill jag ha en knapp som visar reglerna för spelet så att det blir enklare att lära sig spelet.

Bekräftelse

Funktionell

- Finns det en knapp lättillgänglig med en symbol som förmedlar att den visar reglerna?
- Kommer reglerna upp när man tryckt på knappen?

User Story

ID : STK007
Namn: Starta om spelet

Beskrivning

Som användare vill jag ha en knapp som startar om spelet så att jag kan starta ett nytt spel när jag vill.

Bekräftelse

Funktionell

- Finns det en knapp lättillgänglig med en symbol som förmedlar att den startar om spelet?
- Ligger knappen så man inte kan komma åt den av misstagen medan man spelar?
- Startas spelet om när jag klickar på knappen?

User Story

ID : STK008
Namn: Spelregler

Beskrivning

Som användare vill jag att en pjäs inte flyttas från en position till en annan om den inte får det enligt spelreglerna

Bekräftelse

Funktionell

- Är endast tillåtna drag markerade som möjliga drag när jag klickar på en pjäs?

User Story

ID : STK009
Namn: Ny skärm vid vinst

Beskrivning

Som användare vill jag se en ny skärm med vilken spelare som vann när någons drottning är omringad, för att veta när spelet är över och vad resultatet blev.

Bekräftelse

Funktionell

- Kommer det upp en ny skärm när spelet är över?
- Visar skärmen vem det är som vunnit?

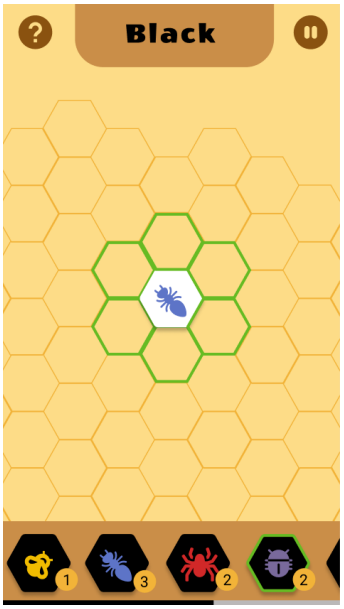
2.2 Definition of Done

För att säkerställa att hela gruppen är överens om när en user story anses som klar, har gruppen satt upp en Definition of Done-lista.

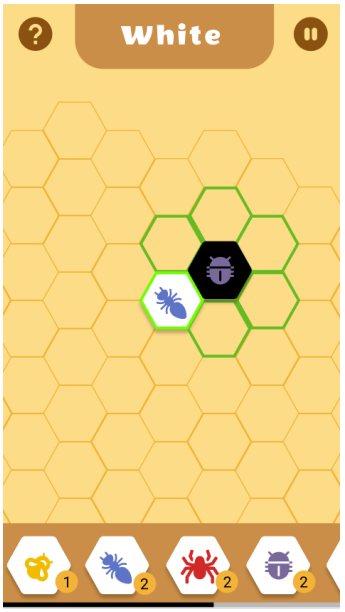
- Koden ska genomföra det den förväntas göra, utan komplikationer
- Koden ska vara bra ur ett objektorienterat perspektiv
- Kraven för den aktuella User Storyn ska vara mött
- Test som testar aktuella metoder ska finnas och bli godkända
- All kod ska vara dokumenterad

2.3 Användargränssnitt

I figur 1 och 2 syns prototypen som gjordes i Figma där man kan se hur GUI:et hade sett ut om applikationen utvecklats vidare och funktionalitet såsom en hjälp-knapp med regler hade implementerats.

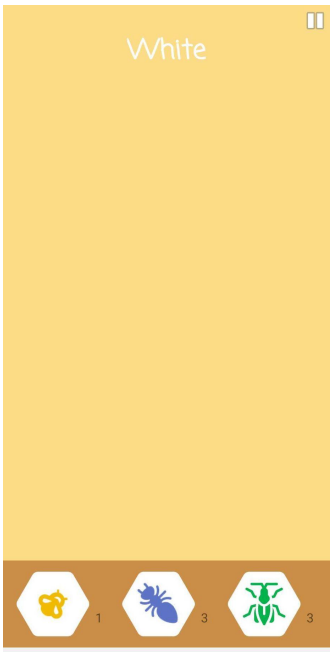


Figur 1: Figmaprototyp för ett drag

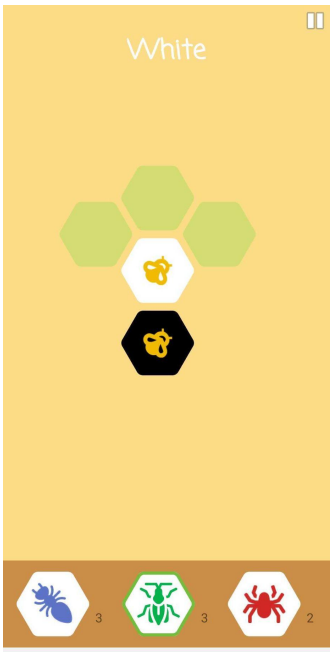


Figur 2: Figmaprototyp för ett drag

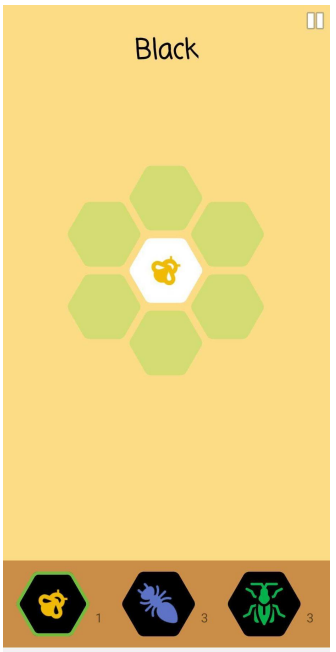
När applikationen sätts igång kommer man till huvudsidan (se figur 3). Ett nytt spel skapas och startas automatiskt. Genom att klicka på pjäserna och sedan på de upplysta platserna kan man flytta och placera ut pjäser. Vyn för hur det ser ut när den vita respektive svarta spelaren ska göra ett drag kan ses i figur 4 och 5.



Figur 3: Huvudsida



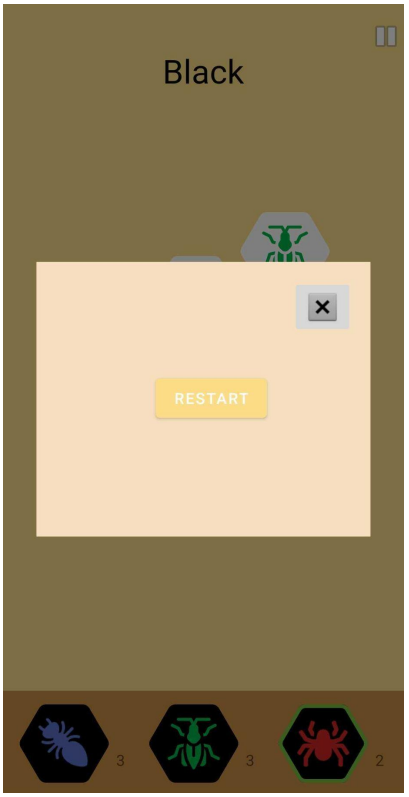
Figur 4: Vy vid vita spelarens tur



Figur 5: Vy vid svarta spelarens tur

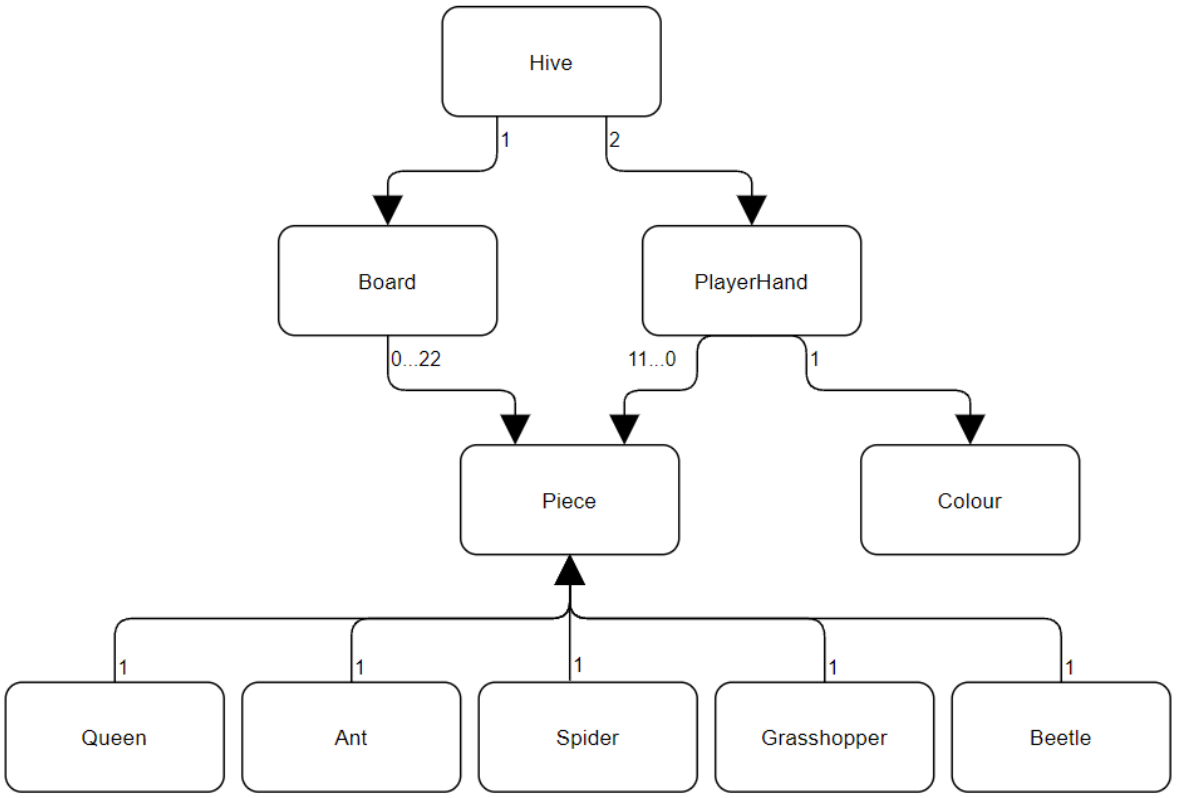
När en spelare vinner dyker en ruta upp (se figur 6) som meddelar vem som vunnit, och ger möjlighet att starta ett nytt spel. Högst upp i högra hörnet finns en pausknapp som tar fram en ruta (se figur 7) där man kan välja att starta om spelet.

Figur 6: Vinstruta



Figur 7: Pausmeny

3. Domänmodell



3.1 Beskrivning av klasserna

Hive
Representerar spelet. Innehåller en board och två playerHands och håller koll på vems tur det är, vilken piece som är vald och om det är någon piece vald.

Board
Håller koll på pjäserna och deras position på brädet, där brädet representeras av en hashmap. Brädet har en variabel för varje drottning och håller koll på när en drottning blivit omringad och spelet då är slut.

PlayerHand
Har koll på vilka pjäser som befinner sig i respektive spelares hand. När ett objekt av PlayerHand skapas, skapas pjäserna som varje spelare ska ha från början.

Colour
En Enumeration som håller de två färgerna som en spelare kan ha, svart och vit.

Piece
I Piece finns de regler för hur pjäserna rör sig som är gemensamma för alla pjäser.

Queen
Ansvarar för hur drottningen kan röra sig. Drottningen kan enbart röra sig ett steg i valfri riktning, så länge det inte redan ligger en pjäs där.

Ant
Ansvarar för hur myran kan röra sig. Myran kan röra sig valfritt antal steg i valfri riktning längs med de utlagda pjäserna.

Grasshopper
Ansvarar för hur gräshoppan kan röra sig. Gräshoppan kan hoppa över andra pjäser i raka linjer, men inte över tomma platser.

Beetle
Ansvarar för hur skalbaggen kan röra sig. Skalbaggen kan röra sig ett steg i valfri riktning, och har förmågan att krypa upp på andra pjäser.

Spider
Ansvarar för hur spindeln kan röra sig. Spindeln kan röra sig exakt tre steg längst med de utlagda pjäserna, i valfri riktning.

4. Referenser

6.1 Verktyg

- **Android Studio** IDE:n vi gjorde projektet med.
- **Draw.io** Ett verkyg som användes för att måla UML-diagram.
- **Figma** Ett verktyg som användes för att ta fram den grafiska designen.
- **GitHub** Lagring och versionshantering för Git av projektet.
- **Google Drive** Samlad lagring av samtliga dokument rörande projektet.
- **Slack** Kommunikationstjänst som användes för kommunikation om projektet
- **Trello** Organiseringsverktyg för de User Storys vi jobbat utifrån i projektet
- **Zoom** Kommunikationsverktyg för videosamtal för samtal om projektet.

6.2 Bibliotek

- **JUnit** används för att testa programmet.