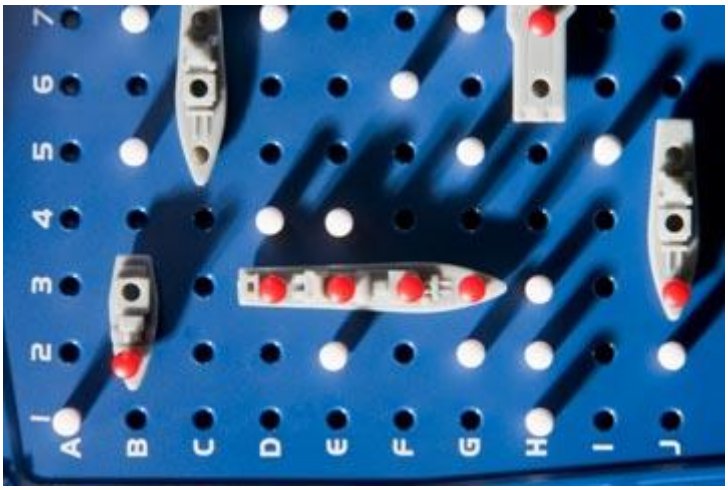
Rapport Nätverksprogrammering LTH

EDAF65

VT 2018

BATTLESHIP



https://entertainment.howstuffworks.com/leisure/brain-games/battleship-game1.htm

Daniel Andersson soc13dan
Isabelle Andersson mat13ian
Petter Haglund vov11pha
Erik Kronberg lan14ekr
Otto Sörnäs dic15oso

2018-03-02

Bakgrund

Denna rapport avser till att beskriva projektet: *BATTLESHIP* som har genomförts i kursen EDAF65. Syftet med projektet var att sammanväva de viktigaste momenten och se hur de kan användas i praktiken.

Anledningen till att vi valde att göra *Battleship* är att det är ett relativt enkelt spel med två spelare och ett fåtal definitiva regler, vilket lämpar sig utmärkt i detta sammanhang.

Kravspecifikation

Till grund för projektet hade vi vissa riktlinjer. Dessa var:

1. Spelet ska funka för två olika spelare på samma nätverk.
2. Det ska finnas en ”ready” funktion.
3. Ha båtar i olika storlek
4. Placera båtar med ”Drag and drop”
5. Event vid ”träff” och ”miss”

Modell

Vi valde att kommunicera via UDP. Anledningen var att UDP är snabbt och enkelt att implementera. Vårt spel har inte behov av den säkerhet som TCP kan bidra med, dessutom är UDP standard för de flesta onlinespel.

Klass: OnlineActivities

Denna klass utgör spelet. Klassen håller koll på brädet, båtarnas placering, antalet skepp m.m. Brädet är egentligen en vektor med Square-objekt. Square-klassen innehåller ett antal booleans för att hålla koll på vad som finns i rutan samt om den har blivit beskjuten eller inte.

Klass: Monitor

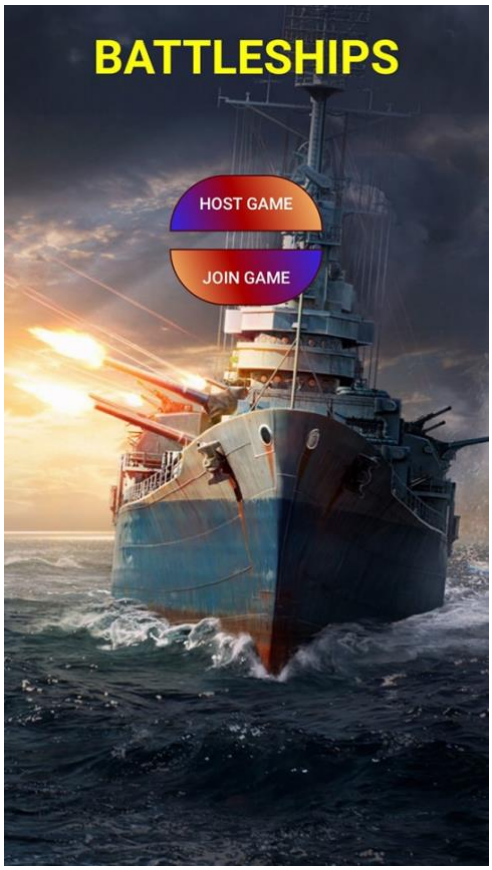
Monitorns funktion är att hålla koll på vems tur i ordningen det är. Annan funktionalitet som att hämta in motståndarens positioner och skicka sina egna. Skickar också info om var man har skjutit.

Klass: ClientThread, ServerThread

Klient och Server klasserna ärver från thread och kan ses som arbetstrådar. Deras jobb är att fixa en uppkoppling och skicka data. Uppbyggnaden av dessa två klasser är väldigt lika. De består inledningsvis av en uppkopplingsfas, setupfas (utsättning av båtar) och slutligen spelfasen som avslutas först när någon har vunnit.

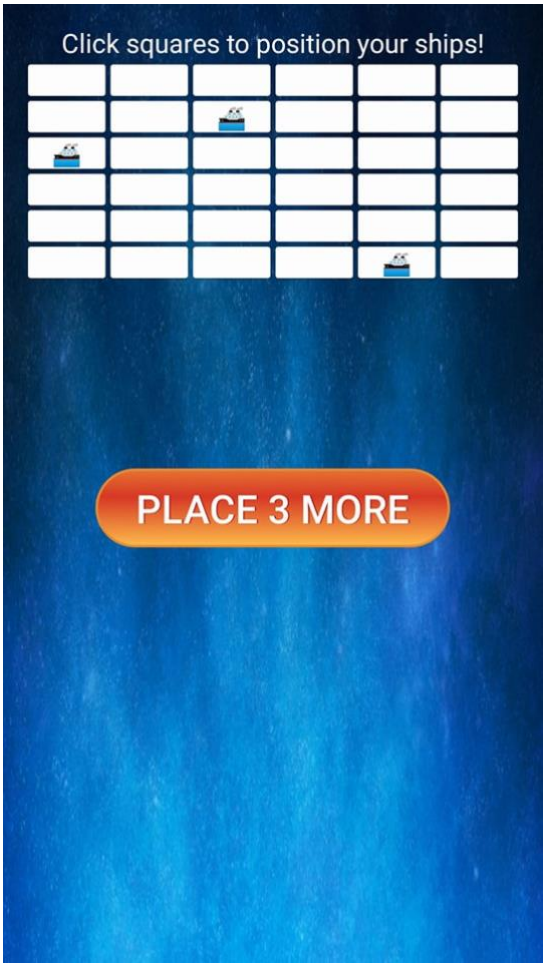
Användarhandledning

Spelarna måste vara på samma nätverk (wifi). En användare skapar ett spel (host game) vilket möjliggör för en motståndare att ansluta (join).



Figur 1: spelets startsida. Välj om du vill Hosta eller joina.

Nästa steg är att placera ut skeppen. Detta görs genom att klicka på valfria rutor. När man är redo klickar man på ”ready”.



Figur 1 Utplacering av skepp. Här syns även spelplanen med 36 rutor.

När båda spelare har klickar ”ready” börjar spelet. Spelarna kommer inte längre att kunna se sina egna båtar, utan ser en tom plan. I själva verket är det motståndares plan man ser fast båtarna är gömda, d.v.s. spelarnas bräden byts ut. Detta gör att man inte behöver skicka en fråga till motståndaren varje gång man skjuter om det är en hit eller miss.

Den som anslöt sig till spelet har första skottet. Man skjuter genom att trycka på valfri ruta.

När en spelare har träffat alla 6 båtar är spelet över.

Utvärdering

Vi är nöjda med spelets utformning och funktion, dock gjorde tidsbristen att vi inte hann med att lägga in alla funktioner vi ville ha från början. Spelet är i alla fall fullt fungerande. Det finns potential att utveckla spelet ytterligare med t.ex. mer innovativ layout, drag and drop samt olika storlekar på båtarna. Man kan också tänka sig att ha en score-lista.

Vi i projektgruppen hade ingen tidigare nämnvärd erfarenhet av Android Studios. Detta gav upphov till en del oförutsedda svårigheter, vilket var en av anledningarna till att spelet blev något avskalat. Vi valde också att prioritera funktionalitet framför prestanda vilket gör att koden kan uppfattas något ”stökig”.

Programlistor

https://isabelleandersson.github.io/