Projektdelen i kursen EITA15, Digital system

Gruppdeltagare

Oskar Elfström, oelfstrom@gmail.com
Richard Nilsson, ri5728ni-s@student.lu.se
Simon Thall, si7608jo-s@student.lu.se
Victor Paulsen, victor_kai_oscar.paulsen.8685@student.lu.se

Applikationsbeskrivning

Applikationen som vi tänkt utveckla är spelet sänka skepp. Den funktionalitet som ska utvecklas är framförallt Startmeny, Spelplan och en datorstyrd motståndare. Händelser som ska registreras i spelet är bland annat:

- Var man släpper bomben
- Om bomben träffar
- Om en hel båt blivit träffad
- Om alla båtar är träffade
- Vilken spelare som vunnit

Vid ett vunnet spel så ska ett meddelande visas och spelet kan startas om. Mitt i spelets gång ska det kunna pausas och avslutas så att man kommer tillbaka till startskärmen.

Användargränssnitt

Input till spelet görs genom ett tangentbord kopplat till USB-porten på FPGA. Genom att trycka på piltangenterna ska spelaren kunna navigera runt i menyer och spelplan och sedan kunna godkänna/OK genom att trycka på Enter.

Outputs sker främst genom en skärm kopplad till VGA-kontakten på FPGA. Kan även ske att displayer på FPGA används för teknisk information (ex. Vilken knapp som är nedtryckt eller antal interrupts) under utvecklingen. VGA-styrenheten programmerad på FPGA kanske modifieras för att få en snyggare rendering av pixlar på skärmen.

Spelplan

Spelplanen ska vara uppdelad i två rutnät där vänster spelplan är spelarens och höger är datorns. Varje ruta i rutnätet ska ha 4 tillstånd, Neutral, bombad, bombad med träff, båt och sänkt.

Neutral
Alla rutor är neutrala från början. Rent grafiskt så ska det se ut som att rutan innehåller vatten. En neutral ruta kan bytas till bombad, bombad med träff och båt.

Bombad

När spelaren eller datorn markerat en ruta och väljer att bomba den så kommer den ändras till en bombad ruta om ingen båt finns på den platsen. Kan grafiskt visas som ett rött kryss eller vågor som representerar att missilen endast åkte ner i vattnet.

Bombad med träff
Om en bomb träffar en båt så ändras den neutrala rutan till en träffad. Grafiskt visas detta genom att en explosion finns på den rutan. En träffad ruta kan omvandlas till en sänkt båt om alla rutor båten står på blir träffade.

Båt
När en båt placeras på en neutral ruta så ändras den till en båt. Det är endast de rutor med en båt på som kan bytas till en *bombad med träff* eller *sänkt*. Båtarna finns i olika storlekar 1, 2, 3 och 5 som kan roteras. De får inte placeras på intilliggande rutor (även diagonaler). Grafiskt så bör båten se ut som en båt uppifrån som sträcker sig över flera rutor. Man ska däremot inte kunna se sin motståndares båtar.

Sänkt
När alla rutor som en *båt* tar upp träffats så blir båten sänkt. Detta innebär att alla intilliggande rutor också träffas (det går ej placera en båt där från början) och att ett liv försvinner. Motståndaren kommer då se den sänkta båten på spelplanen. Grafiskt så bör en sänkt båt ritas som en båt under vatten.

Varje ruta görs som ett objekt som sedan ligger i en nested array (2d-array). Arrayerna blir då som ett koordinatsystem där man kan hämta objekt ex. ruta[0,4] (rutan i raden längst till vänster 5 rader ner) genom att göra map1[0][4]. Blir även enkelt att modifiera genom get- och set-metoder: map1[0][4].tileType(2); Exemplet skulle då sätta typen på rutan till *Bombad med träff*.

Motståndare

Spelet ska börja med att motståndaren är datorstyrd. Anledningen till en datorstyrd motståndare är att det är det mest logiska på ett spel där man inte ska se motståndarens spelplan. Vilket skulle vara svårt att undvika när man spelar på samma skärm. Idén finns däremot att man ska kunna spela mot en riktig spelare genom att den som inte spelar får kolla bort från skärmen.

Funktionaliteten på motståndaren ska vara att datorn gissar på olika rutor tills en träff sker. Vid träff ska den avgöra om en båt sänktes eller om det kan finnas fler delar av samma båt på de omkringliggande rutorna. Om fler delar finns så ska den försöka att gissa sig fram tills hela båten är sänkt.

Uppdelning av arbete

Applikationens olika delar delas upp till mindre för att alla ska kunna utveckla samtidigt. Vilken del var och en får beror på tidigare erfarenheter samt vad man är intresserad av

Oskar - Backend, spelfunktioner
Richard - Datormotstånd, AI
Simon - Grafiskt gränssnitt; meny, ritning av spelplan
Victor - Backend; spelfunktioner, koordinatsystem, objekt

Uppdelningen är endast för att tydligare visa vem som ska ansvara för vilken del av applikationen. Vid behov kan uppgifter ändras för att kunna samarbeta till en lösning.