|  |
| --- |
| Rinmangymnasiet |
| Sänka Skepp |
| En projektrapport för slutuppgiften i Programmering 1 |
|  |
| **Patrik Olsson TE16D** |
| **2018-08-20** |

|  |
| --- |
|  |



Innehållsförteckning

[1 Bakgrund 2](#_Toc522559370)

[1.1 Mål 2](#_Toc522559371)

[1.2 Syfte 2](#_Toc522559372)

[1.3 Metod 2](#_Toc522559373)

[2 Problembeskrivning 2](#_Toc522559374)

[2.1 Kravspecifikation 2](#_Toc522559375)

[2.1.1 Funktionella krav 2](#_Toc522559376)

[2.1.2 Icke funktionella krav 2](#_Toc522559377)

[2.2 Begränsningar 2](#_Toc522559378)

[3 Design 3](#_Toc522559379)

[3.1 Användargränssnitt 3](#_Toc522559380)

[3.2 Pseudokod 3](#_Toc522559381)

[4 Avslutande diskussion 3](#_Toc522559382)

[5 Källförteckning 3](#_Toc522559383)

# Bakgrund

Mitt slutprojekt kommer på grund av brist på tid och framför allt motivation, att bli ett enkelt sänka skepp-spel.

## Mål

Programmet ska uppfylla kraven under rubriken **2.1.1 Funktionella krav**.

## Syfte

Syftet med projektet är att skapa ett projekt som kan återspegla elevens kunskaper som denne har fått under kursens gång, och att eleven ska få pröva på hur det är att utveckla en produkt med samma metoder som ute på företagen.

## Metod

Eftersom jag var ensam i detta projekt, så finns det nästan inga anteckningar om planering, mer än det jag skrev i detta dokument innan projektets början. Dokumentation skedde på två fronter samtidigt. Dels i detta dokument, där all formalia skrevs ner på svenska, dels på GitHub där all kod med kommentarer finns. På GitHub går det även att spåra alla framsteg som gjordes med programmet. Alla kommentarer i koden och på GitHub skrivs på engelska.

<https://github.com/TNTPata/Battleship>

# Problembeskrivning

Som reglerna i sänka medger, så skjuter en spelare i taget ett skott på en inte tidigare beskjuten ruta på motståndarens spelbräde. Om det inte finns någon del av ett skepp på rutan, så är det en miss, och rutan markeras med vit färg. Om någon del av ett skepp finns på rutan, så är det en träff, och rutan markeras med gul färg. När alla rutor av ett skepp har blivit träffade, så ska motspelaren få veta att detta har skett, genom att ”Skepp sänkt” utropas. När alla skepp på en spelares planhalva är sänkta, har den andra spelaren vunnit spelet.

## Kravspecifikation

Nedan beskrivs de funktioner som programmet ska ha.

### Funktionella krav

* Två spelbräden, 10 x 10 positioner.
* Enspelarläge
* Automatisk placering av skepp på båda planhalvor.
* Markerring av träff på motståndaren med gult, miss med vitt, icke beskjuten ruta med ljusblått.
* Markering av egna skepp med grönt, träffade skepp med rött, miss med vitt, icke beskjuten ruta med ljusblått.
* Vinnaren ska ropas ut när alla skepp på en sida är sänkta.

### Icke funktionella krav

* Dator med Windows 10 som operativsystem.

## Begränsningar

Uppgiften är att skapa ett program som uppfyller **2.1.1 Funktionella krav**, allt annat som inte har med det att göra är därför uteslutet.

# Design

Nedan beskrivs hur programmet är designat. Om det inte var uppenbart genom att läsa rubriken.

## Användargränssnitt

Spelet ska bestå av två ”spelbräden” som utgörs av ett 10x10 knappnät. Mellan de två spelbrädena finns en ruta med diverse statistik och kontrollknappar. Gränssnittet byggs med hjälp av Windows Forms.

## Pseudokod

1. Start.
2. Automatisk placering av skepp sker på båda spelplaner, och markeras med grön färg på egna planen.
3. Användaren trycker på en knapp på motståndarens spelbräde.
4. Om ett skepp finns på den givna positionen, markera med färg som visar träff (gul), annars färg som visar miss (vit).
5. Datorn skjuter mot användarens spelbräde.
6. Repetera steg 3 till 5 tills en spelares spelplan är helt tom på skepp.
7. Presentera vinnare.
8. Fråga om båda spelarna vill köra igen. Om ja, gå till 2.
9. Stäng programmet.
10. Slut.

# Avslutande diskussion

Det sägs att det finns två saker människor är bättre på än några andra djur på planeten. Det första är att döda levande ting, och det andra är att förutse framtiden. När jag påbörjade detta projekt, någon gång i Mars 2018, skulle detta ta högst en månad att slutföra. I extrema fall kunde det ta två. På grund av brist på motivation och buggar som har vägrat att bli lösta, blev projektet rejält uppskjutet, och redovisat någon timme innan skolavslutningen.

Det finns många saker som kunde ha utförts bättre, till exempel planeringen av projektet, med pseudokod och fysiska ritningar och skisser. Flera lösningar på problem under kodningen har lösts med ganska fula och osnygga metoder.

Två saker i projektet står ut, och får trots allt ändå ses som lyckade. Det första är den ”AI” som jag gjorde för att placera ut skepp. Den väckte mitt intresse för att programmera en mer avancerad artificiell intelligens, som mycket fort släcktes när jag insåg hur otroligt mycket jobb som skulle ligga bakom ett sådant projekt.

Det andra som har lyckats i projektet är dokumentationen via GitHub. Användningen av GitHub har hjälpt mig att mycket tillförlitligt spåra mina framsteg, och låtit mig göra större ändringar i koden utan att behöva oroa mig för att det fungerade originalet skulle sluta fungera, tack vare GitHubs grenfunktioner.

# Källförteckning

* Intet.