Terningespil – Alkoholskrig



Navn og Klasse: Tobias de Lasson 2.z

Vejleder: Hans-Christian

emne: Underholdning

Projektperiode: D. 2 Marts 2022 til 16 Maj 2022

rapportens indhold ikke må gengives uden aftale med skolen

Indholdsfortegnelse

[Kort abstract 3](#_Toc98934829)

[Problemformulering 3](#_Toc98934830)

[Kravspecifikation 3](#_Toc98934831)

[Funktionsbeskrivelse 3](#_Toc98934832)

[Dokumentation af selve programmet 3](#_Toc98934833)

[Test af programmet 3](#_Toc98934834)

[Konklusion 3](#_Toc98934835)

[Kildeliste 3](#_Toc98934836)

[Bilag 3](#_Toc98934837)

# Kort abstract

I denne opgave har jeg produceret et spil, som har til formål at underholde. Målgruppen med spillet er primært unge mennesker, da spillet hovedsageligt tager form af et drukspil. Jeg har produceret en prototype til spillet. Hvis man skulle udgive spillet, ville jeg nok gøre det i form af en app på mobilen, hvor man skulle konkurrere online.

# Problemformulering

Jeg skal producere et program, som har til formål at underholde. Jeg vil løse problemet ved at producere et nyt drukspil, kaldt alkoholkrig.

I Alkoholskrig er der 2 spillere. Hver spiller starter med 10 liv, og målet er at reducere modstanderens liv til 0, mens der sker forskellige ting på vejen. Man reducerer modstanderens liv ved at slå et par terninger, og den der har slået lavest, mister et liv. Når modstanderen har ramt 0 liv, skal der komme en slutskærm, hvor taberne får at vide de skal bunde hvad end de drikker (helst alkohol), som også er grunden til navnet ”Alkoholskrig”

I spillet skal der være forskellige former for ”power ups”, som skal ændre udfaldet af terningerne slået. Fx kunne der være en egenskab som gør at man slår med 3 terninger i stedet for 2. Udover det, skal der ske noget forskelligt ved hvert terningeslag. Fx kunne det være at man selv skulle drikke 2 slurke hvis man slår 7

# Kravspecifikation

Da programmet hovedsageligt er designet til at kunne simulere terninger, og fungere som et spil, skal man selvfølgelig tage udgangspunkt i det når man laver krav.

|  |
| --- |
| Krav |
| Spillet skal kunne simulere et tilfældigt slag mellem 2 til 12. |
| Spillerne skal miste et liv efter hver runde |
| Spillet skal slutte når en spiller når 0 liv |
| Spillet skal kunne genstartes i en slutskærm |

Selvfølgelig skal spillet også have basale funktioner som at slutte når en spiller når 0 liv, og skal være i stand til at kunne genstartes, da et drukspil, helst skal have nem gentagelighed.

# Funktionsbeskrivelse

Spillerne skal være i stand til at bruge enten tastaturet eller musen til at vælge ”power-ups” og vælge at slå terningerne.

Spillet skal bestå af en skærm delt i 2 menuer, spiller 1’s menu og spiller 2. Hver spiller har hver sin side, men deres terningeslag skal være fremvist i midten af skærmen. Deres liv står i hvert deres hjørne.

Skærmlayoutet er skitseret nedenunder:



Eftersom der er 2 spillere, skal de have hver deres sæt knapper som de kan bruge når de vil bruge power-ups og slå deres terning. Undervejs vil jeg derfor eksperimentere for at se hvilke knapper der fungerer bedst. Keyboardet skal deles op i 2, så hver spiller har hver deres side.

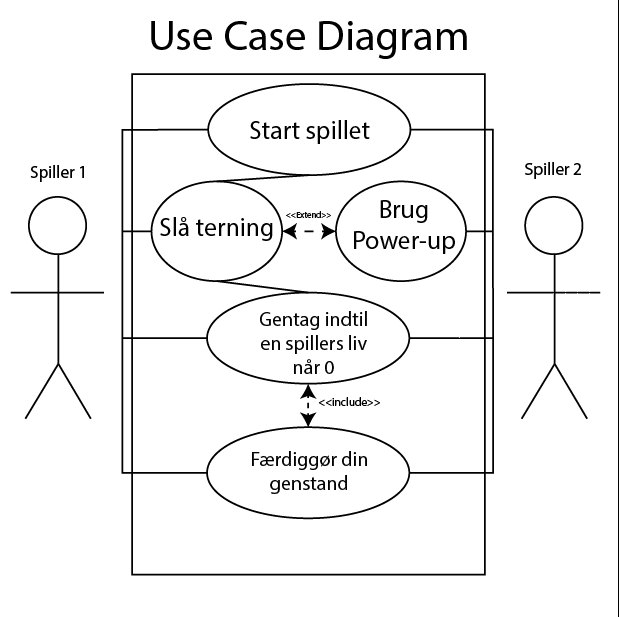
# Dokumentation af selve programmet

Programmets har hovedsageligt 3 stadier: En introskærm, et spil, og til sidst en slutskærm, som kan bringe en til introskærmen.

Spillet er det der er hovedfokus i mit program, og er også der hvor de fleste processer sker.

Der er 2 spillere, og hver runde kaster hver spiller en terning, som simuleres ved hjælp af ’random’ funktionen i Processing, den svære del kommer ind i billedet når de vil til at bruge power-ups, som bliver tjekket ved hjælp af forskellige booleans.

Som man kan se på Use Case Diagrammet under, er der i alt 5 processer, som spillerne skal tage stilling til. De starter selvfølgelig først spillet, hvorefter de slår terninger, hvor de kan vælge at bruge power-up. De gentager så denne proces indtil en af dem når 0 liv, hvorefter taberen skal ’bunde deres genstand’

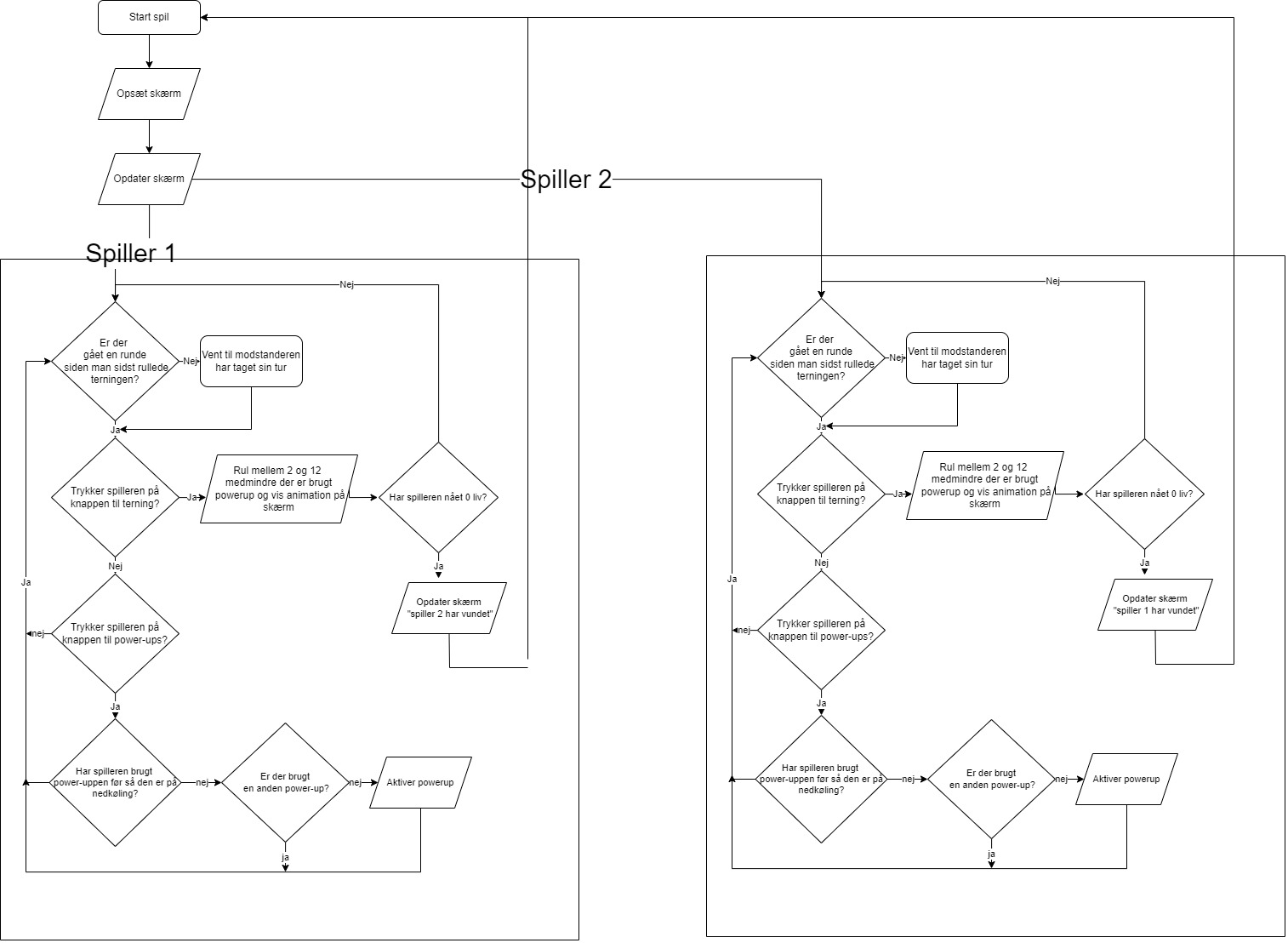


Tilstandsdiagram

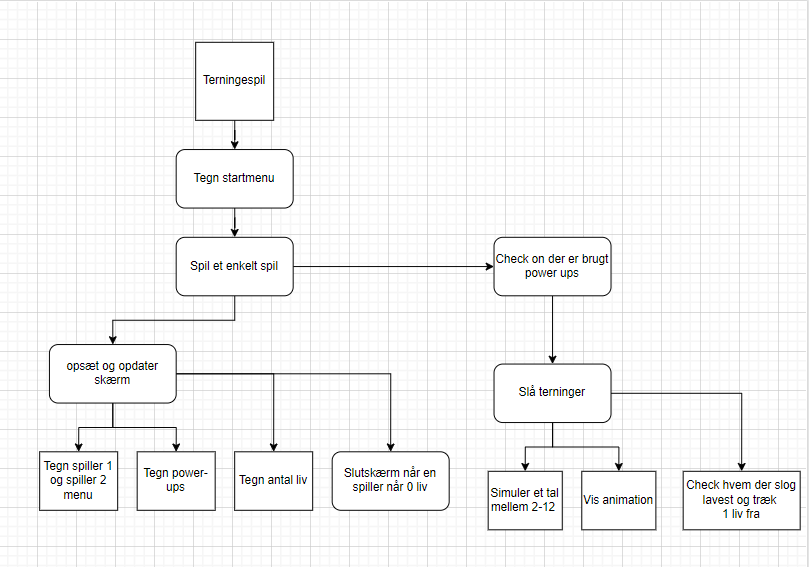
Et billede, der indeholder tekst, atletik

Automatisk genereret beskrivelse

Tilstandsdiagrammet minder meget om Use Case diagrammet, den største forskel er at vi kan se hvordan det kører i en slags løkke, hvis man vælger at genstarte spillet.



Strukturel programmering – her kan vi se hvordan programmet kommer til at fungere fra et mere kodenært synspunkt. Hver spiller kommer sådan set til at have hver deres ”kodeblok”, som man hvis man ville lave til klasser, og så have dem nedarve samme egenskaber, da hver spiller skal være i stand til at gøre de samme ting.



# Test af programmet

|  |  |
| --- | --- |
| Spillet skal kunne simulere et tilfældigt slag mellem 2 til 12. | Kan man slå en terning, som har en tilfældig værdi mellem 2 og 12?  Programmet blev ændret til kun at inkludere 1 terning, men det blev til godt at være i stand til at simulere et terningeslag. |
| Spillerne skal miste et liv efter hver runde | Efter hver runde taber en spiller et liv, medmindre spillerne har slået det samme. |
| Spillet skal slutte når en spiller når 0 liv | Når en spiller når 0 liv, kommer der en slutskærm, hvor der står hvem der har vundet og 2 muligheder; at slukke spillet, eller spille igen |
| Spillet skal kunne genstartes i en slutskærm | Der er en virkende knap til at genstarte spillet ved slutskærmen. |

# Konklusion

Under produktionen af spillet blev det ændret til 1 terning, da 1 terning giver mere mening fra et design-synspunkt. Det gør det en del nemmere for spillerne at sammenligne deres terningekast, som selvfølgelig er en stor del af dette spil.

Da

# Kildeliste

Har ikke brugt nogen kilder til min kode

# Bilag