Age of Insects

**Produkt Rapport**

Kasper Grønne

Contents

[**1. Problemformulering** 3](#_Toc130921739)

[**1.1 Hvor startede det?** 3](#_Toc130921740)

[**1.2 Hvad er problemet?** 3](#_Toc130921741)

[**1.3 Hvad er min løsning?** 3](#_Toc130921742)

[**1.4 Path Choice** 3](#_Toc130921743)

[**2. Projektbeskrivelse** 5](#_Toc130921744)

[**2.1 Komponent 1 – UI** 5](#_Toc130921745)

[**2.2 Komponent 2 – Frontend Scripts** 6](#_Toc130921746)

[**2.3 Komponent 3 – Controller** 6](#_Toc130921747)

[**2.4 Komponent 4 – DataHandler** 6](#_Toc130921748)

[**2.5 Komponent 5 – Server** 6](#_Toc130921749)

[**2.6 Komponent 6 – Backend Scripts** 6](#_Toc130921750)

[**2.7 Case flow illustration** 7](#_Toc130921751)

[**3. Applikationens Overordnet Funktionalitet** 8](#_Toc130921752)

[**3.1 Spillerens synspunkt** 8](#_Toc130921753)

[**3.2 C#** 8](#_Toc130921754)

[**3.3 Unity** 9](#_Toc130921755)

[**3.4 Riptide** 9](#_Toc130921756)

[**4. Use cases** 9](#_Toc130921757)

[**5. Konklusion og Refleksion** 11](#_Toc130921758)

# **1. Problemformulering**

## **1.1 Hvor startede det?**

Et meget populært spil som mange kender til, hedder Age of War.  
I dette spil kæmper man mod en fjende, og opgraderer sin "age" til en stærkere version.   
Eksempel: Hulemænd til Militærmænd.   
I spillet trykker du på en knap, venter nogle sekunder, og så bliver den "unit" du trykkede på skabt og sendt afsted.

Units rykker sig selv, og slår af sig selv. De går i en lige linje og angriber den forreste unit fra det andet hold.

Du kan også bygge "towers" på din base som skyder fjendtlige units der kommer nær din base.

## **1.2 Hvad er problemet?**

Problemet med dette spil er, i min mening, at det er ekstremt lineært. Du opgraderer til en ny age, men der er ingen valg i hvilken age du vil udvikler dig til. Der er ingen "skill expression" i at rank op i age. Der er heller ingen "playstyles" når det kommer til at rank op. Spillet er derfor meget "one-dimensional".

## **1.3 Hvad er min løsning?**

Age of Insects, er meget lig Age of War, men med nogle små ændringer i spillet som gør en stor forskel på spillets struktur. Og vil gøre at alle disse problemer bliver løst:  
1. Skill expression  
2. Playstyles  
3. One-dimensional  
4. Skill ceiling

*Min løsning består af et simpelt nyt system kaldet ”Path Choice”.*

## **1.4 Path Choice**

Man kan specificere *hvilken* age man ranker op til. Spilleren har valget mellem 3 ages, kaldet ”civilisationer”. Civilisationer er unikke og er meget forskellige fra hinanden. Man starter i en base civilisation, kaldet ”Larvae”. Denne civilisation introducerer spilleren til hvordan spillet fungerer, med meget normale units som ikke har nogen form for ”traits”.

Når spilleren unlocker en ny civilisation, kan de vælge mellem 3 forskellige civilisationer. For eksempel, så kan man fra Larvae civilisationen, vælge mellem Ladybug, Earthworm, og Flea civilisationen.

Ladybug civilisationen går op i beskyttelse. Deres units er dyre, men har meget liv.  
Earthworm civilisationen er en well rounded civilisationen. Den er midter valget mellem Ladybug og Flea.  
Flea civilisationen er en aggressiv civilisation, som har billige units der slår hurtigt. Men de har til gengæld ikke meget liv og dør hurtigt.

Efter spilleren har opgraderer til en ny civilisation, har de adgang til 3 nye. Disse civilisationer ændrer sig fra civilisation til civilisation. Dvs. kun bestemte civilisationer kan opgradere til nogle civilisationer.

Diagram, schematic

Description automatically generated with medium confidence  
Diagram af hvilke civilisationer kan opgradere fra og til hinanden.

Dette gør, at spillerens ”playstyle” er personlig og vil blive vist tydeligere i et spil. Hvis en spiller godt kan lide at spille aggressivt, så kan de det. Hvis en spiller godt kan lide at spille mere defensivt, så kan de det. Det tilføjer altså flere dimensioner til spillet, for begge spillere ikke længere altid har de samme ages, så du skal nok ændre den måde du spiller på, hvis din fjende vælger en anden age. Dette gør, at spillerens ”skill expression” stiger, fordi jo bedre du er til spillet, jo bedre kan du adaptere til hvad din fjende gør. Dette gør også at ”skill ceiling” på spillet bliver højrer, for der er flere variabler at lære og holde styr på.

Og til sidst, så ender spillet med ikke at være ”one-dimension”, da spillet har forskellige (altså ikke spejlvendt) gameplay, hvilket vil sige at hvert spil er anderledes.

# **2. Projektbeskrivelse**

Produktet er et spil lavet i Unity2D. Produktet bruger C# som sit primære sprog. Spillet består af flere forskellige komponenter. Projektet er lavet over 4 uger, som en del af et skoleprojekt. En spiller skal kun have internet og en computer at spille spillet på, for at kunne spille.

## **2.1 Komponent 1 – UI**

UI er en stor del af alle spil. Alle spil har noget interaktiv design som spilleren kan bruge. I dette produkt består UI af 2 menuer. StatsMenu, og MainMenu.

* StatsMenu er en simpel menu, som viser hvor meget Gold og EXP, spilleren har. Denne menu er ikke interaktiv, og den eksistens er udelukkende for at formidle information til spilleren.

Text

Description automatically generated with low confidence

* MainMenu er en lidt mere avancerede menu, som består af 3 knapper. 2 af disse knapper fører til deres egen mindre menu.   
  Units knappen fører til en indre menu, der viser de nuværende units som kan købes af spilleren. Civilisationer knappen fører til en indre menu, der viser de nuværende civilisationer som spilleren kan opgradere til.  
  Den sidste knap, som sidder oppe i højre hjørne af i alle de indre menuer, er en tilbage knap, som fører spilleren tilbage til MainMenu.  
  Logo

  Description automatically generated with medium confidence

Når man hover over en unit eller civilisation, vises der hvor meget det koster at opgradere. Det er endnu et stykke tekst, som eksisterer udelukkende for at formidle information til spilleren.

Der er også et input field når man starter spillet, som beder spilleren om at indtaste sit brugernavn. Samt en knap nedenunder som konnekter spilleren til serveren.

Graphical user interface, application

Description automatically generated

Efter du er connected til serveren kommer man ind i en lobby som kan holde op til users. Der er knapper hvor en spiller kan trykke at de er ready, samt en knap som starter spillet.

Graphical user interface, application

Description automatically generated

## **2.2 Komponent 2 – Frontend Scripts**

Frontend scripts er scripts, som reagerer på noget der sker i spillet. Det kan for eksempel være når spilleren køber en unit, eller når en unit dør. Disse scripts snakker med controlleren for at hente eller sende data til DataHandleren eller Serveren. UI hører til denne kategori.

## **2.3 Komponent 3 – Controller**

Controlleren er en individuel komponent, som ikke indeholder noget data. Controlleren er en middleware, som tager rollen af en API. Controlleren indeholder en masse metoder, som enten sender data til serveren, eller sender eller henter data fra DataHandleren. Der er strengt forbudt for controlleren at referere ud af dens eget script, med mindre det er til DataHandleren. (Gammel kode eksisterer stadig i Controlleren som refererer til Shoppen. Dette vil i fremtiden ligge i DataHandleren)

## **2.4 Komponent 4 – DataHandler**

DataHandleren er en individuel komponent, som indeholder alt dataen spillet skal bruge for at fungere. Dette inkluderer for eksempel Units, Civilisationer, Spillere, og Shops. DataHandleren har rollen af en database for spillets frontend. Den eneste komponent som må snakke med DataHandleren er Controlleren. Hvis andre komponenter refererer til DataHandleren, er der brud på arkitekturen som er blev etableret i resten af spillet.

## **2.5 Komponent 5 – Server**

Serveren er en komponent, som skal startes op i et andet instans af Unity. Serveren er lavet i Unity2D, og tager imod requests fra Controlleren. Serverens job er at holde styr på spillere, og står for at sende data til alle spillere når en spiller trigge ret event (fx en spiller køber en unit). Når serveren sender requests til spillerne, er det GameLogic scriptet som tager imod dem. Serveren gør brug af et netværks framework som hedder Riptide.

## **2.6 Komponent 6 – Backend Scripts**

Backend scripts modtager beskeder fra serveren, og kører derefter nogle events, afhængig af beskeden. Backend scripts står for eksempel for at tilføje spillerere til spillets lobby, starte spillet, spawn units og opdaterer unit targets.

## **2.7 Case flow illustration**

Diagram

Description automatically generated

Simpel illustration af arkitekturen af produktet, med eksempel på et case flow for når en spiller køber en unit.

# **3. Applikationens Overordnet Funktionalitet**

## **3.1 Spillerens synspunkt**

En spiller starter spillet, og skriver deres username ind i input fielded. De trykker derefter ”Connect”, hvor spilleren bliver tilføjet til en lobby i spillet, og applikationen derefter sender en request med en ny user til serveren. Hvis spilleren ikke kan få konneksion til serveren, bliver de smidt tilbage til den første side.

I lobbien kan spilleren se, hvilke spillere der er i lobbien, om de er klar til at spille, og en knap som starter spillet. Knappen starter kun spillet hvis der er 2 users i lobbien, og begge er klar. En lobby består af alle spillere online. Dvs. at kun 2 spillere kan spille spillet på samme tid, på samme server.

Efter en spiller har trykket start, starter spillet.

Spilleren har en base som de skal beskytte mod fjenden. Fjenden har sin egen base, som spilleren skal ødelægge. Vinderen er den, som ødelægger den andens base først.

Spilleren kan købe units, som bliver spawned ved deres base. Disse units vil så begynde at gå imod fjendens base for at ødelægge den.

Hvis spillerens units of fjendens units møder hinanden, begynder de at angribe hinanden. Det handler derfor om, at købe de rigtige units på det rigtige tidspunkt, for at vinde mod din modstander.

Spilleren kan opgradere til en ny civilisation, når de har fået nok EXP. De kan selv vælge 1 ud af 3, og når de har valgt en civilisation, kan de ikke gå tilbage.

## **3.2 C#**

C# er et programmeringssprog, som jeg har meget erfaring i. C# er objektorienteret, og er derfor ekstremt nemt at arbejde med når det kommer til at lave spil. I spil skal man hele tiden have adgang til objekter, og holde styr på, om de objekter gør som de skal. Derfor er C# et naturligt og populært sprog at bruge, når man laver spil.

## **3.3 Unity**

Unity er et meget kendt og populært engine/framework. Unity bliver for det meste brugt til at lave spil, men det er blevet meget populært at kører simulationer via Unity, da deres physics engine er nem at arbejde med, og du kan sætte ting op meget nemt og hurtigt.

Unity’s physics engine er også en af grundene til at Unity er blevet så populært mellem ny spilleudviklere. Unity’s physics engine gør det nemt for folk at lave spil, hvor objekter skal bevæge sig, hvilket meget vigtigt i de fleste spil.

Jeg har valgt at lave spillet i Unity, da jeg er meget erfaring med Unity i forvejen, og da jeg ikke havde meget tid, krævede det at jeg vidste hvad jeg arbejdede med. Unity er også ekstremt hurtigt at arbejde i, da man meget hurtigt kan lave ændringer via deres editor, hvor man kan sætte værdier uden overhovedet at skulle programmere noget.

## **3.4 Riptide**

Riptide er en open source package man kan bruge i Unity, som er meget ny og simpel. Den lader én gøre meget kompliceret ting, meget simpelt. Der er andre, mere simple og udviklet løsninger, men de lader ikke programmøren lave kompliceret ting. I hvert fald ikke uden man støder på mange problemer.

# **4. Use cases**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Event | Actor | Basic Flow |
| Use Case 1 | Spiller køber unit | Spiller | Spiller trykker på en knap, som sender et ID til shop scriptet. Shoppen sender eventet videre til controlleren, som henter den specifikke unit prefab fra DataHandleren, og derefter sender spillerID, UnitID og dens spawn time videre til serveren. Serveren kører derefter en timer på hvor lang tid det tager at spawn den specifikke unit. Derefter sender serveren ud til alle spillere, at spilleren med det specifikke ID, har købt unit med det specifikke ID. Klienten tager imod denne besked, og spawner den specifikke unit for den specifikke spiller. Den specifikke unit instans bliver derefter tilføjet til DataHandlerens playerUnit liste af dictionaries. |
| Use Case 2 | Unit dør | Spillet selv | Unit sender en besked til backend scripts, at den er død. Backend script fjerner derefter denne unit, og sender en besked til Controlleren at den skal fjerne den specifikke unit fra DataHandleren. Derefter opdaterer Backend scripts alle unit targets, så de får nye targets. |
| Use Case 3 | Spiller opgraderer civilisation | Spiller | Spiller trykker på en knap, som sender et ID til shop scriptet. Shoppen sender eventet videre til controlleren, som derefter sender spillerID og CivilisationID videre til serveren. Serveren sender derefter ud til alle spillere, at spilleren med det specifikke ID, har opgraderet civilisation med det specifikke ID. Klientens Backend scripts tager imod denne besked, og kører derefter en civilisation action, som opgraderer spillerens civilisation. Denne metode får Backend scripts fra DataHandleren, via Controllerne. Efter den er opgraderet, opdaterer Backend scripts shoppens shops, så det nu er nye units and civilisationer som spilleren kan købe. |
| Use Case 4 | Spiller konnekter til serveren med userame | Spiller | Spilleren indtaster sit username, og trykker på en knap, som åbner en lobby op. Frontend scripts sender derefter en besked til controlleren, som indeholder usernamet som spilleren har indtatset. Controlleren sender en besked til serveren med dette username. Serverer modtager denne besked, og laver en ny spiller. Den sender derefter ud til alle spillere, at en ny spillet har joined. Spilleren forbliver på lobby screen. Andre spillere kan se den nye spiller med hans username |
| Use Case 5 | Spiller konnekter til serveren uden username | Spiller | Spilleren trykker på en knap, som åbner en lobby op. Frontend scripts sender derefter en besked til controlleren, som sender en besked til serveren. Serverer modtager denne besked, og laver en ny spiller. Den sender derefter ud til alle spillere, at en ny spillet har joined. Spilleren forbliver på lobby screen, hans navn er Guest samt hans interne ID. Andre spillere kan se at Guest (ID) er i lobbien. |
| Use Case 6 | Spiller fejler at connect | Spiller | Spilleren indtaster sit username, og trykker på en knap, som åbner en lobby op. Frontend scripts sender derefter en besked til controlleren, som indeholder usernamet som spilleren har indtatset. Controlleren sender en besked til serveren med dette username. Serveren modtager ikke denne besked, og beskeden timer out. Backend scripts fortæller derefter Frontend scripts, at spilleren skal tilbage til ”Connect” menuen. |
| Use Case 7 | Spiller starter spillet med nok folk i lobbien, og alle er klar | Spiller | Frontend scripts sender en besked til Controlleren, som sender en besked til serveren. Serveren validerer denne besked, og sender en bool tilbage til alle spillere. Alle klienters backend scripts modtager denne besked, og sender en besked til Frontend scripts at de skal vise spillet. |
| User Case 8 | Spiller starter spillet, uden nok folk i lobbien | Spiller | Frontend scripts sender en besked til Controlleren, som sender en besked til serveren. Serveren validerer denne besked, og sender en bool tilbage til alle spillere. Alle klienters backend scripts modtager denne besked, og sender en besked til Frontend scripts at de ikke skal vise spillet. |
| User Case 9 | Spiller starter spillet, uden alle er klar. | Spiller | Frontend scripts sender en besked til Controlleren, som sender en besked til serveren. Serveren validerer denne besked, og sender en bool tilbage til alle spillere. Alle klienters backend scripts modtager denne besked, og sender en besked til Frontend scripts at de ikke skal vise spillet. |
| User Case 10 | Spiller trykker klar | Spiller | Frontend scripts sender en besked til Controlleren, som sender en besked til Serveren. Serveren modtager denne besked, og sender derefter beskeden videre til alle spillere i lobbien. |

# **5. Konklusion og Refleksion**

Dette er et spil jeg altid gerne har villet lave. Og jeg vidste med det samme hvad jeg ville lave, da jeg fik muligheden.

Spillet har stadig nogle balancing issues og nogle bugs. For eksempel sker der ikke noget, når en spiller vinder. Og når spilleren er i maks civilisation, fjerner shoppen ikke sin civilisations undermenu, hvilket resulterer i at de kan trykke på knapper som ikke gør noget.

Valget af C# og Unity har til gengæld været ekstremt vigtigt og godt, og har hjulpet mig med at blive så færdig som muligt meget hurtigere end hvis jeg havde valgt andre muligheder.

Valget af Riptide multiplayer, i bagklogskab, var ikke nødvendigt. Alt jeg gør er at sende simple requests, men til gengæld var det ekstremt nemt at arbejde med, og vil helt klart arbejde med det i fremtiden hvis jeg fik muligheden. Det var noget jeg var helt ny i, men det var fedt og nemt at bruge.

Jeg er stolt af hvad jeg har lavet, og det er noget jeg helt klart ville arbejde på i min fritid (hvis jeg havde den tid ☹). Jeg er faktisk meget taknemmelig for at skolen har givet mig disse 4 uger, da det har givet mig muligheden for at arbejde frit, og på noget som jeg ville lave alligevel. Det var som betalt ferie, da det er noget jeg havde tænkt mig at lave i min fritid når jeg så fik den tid.

Der var mange ups and downs, især når det kom til at tilføje multiplayer, men der var helt klart det værd. Jeg har spillet spillet flere gange med mine venner i Aarhus, og vi har det meget sjovt med det.