Age of Insects

**Process Rapport**

Kasper Grønne

Contents

[**1. Dag 1 – 6** 2](#_Toc130932813)

[**1.1 Dag 1** 2](#_Toc130932814)

[**1.2 Dag 2** 4](#_Toc130932815)

[**1.3 Dag 3** 4](#_Toc130932816)

[**1.4 Dag 4** 4](#_Toc130932817)

[**1.5 Dag 5** 4](#_Toc130932818)

[**1.6 Dag 6** 5](#_Toc130932819)

[**2. Dag 10 - 13** 9](#_Toc130932820)

[**2.1 Dag 10** 9](#_Toc130932821)

[**2.2 Dag 11** 9](#_Toc130932822)

[**2.3 Dag 12** 9](#_Toc130932823)

[**2.4 Dag 13** 11](#_Toc130932824)

[**3. Dag 14 – 19** 11](#_Toc130932825)

[**3.1 Dag 14** 11](#_Toc130932826)

[**3.2 Dag 15 – 19** 11](#_Toc130932827)

[**4. Dag 22 – 24** 11](#_Toc130932828)

[**5. Konklusion** 12](#_Toc130932829)

# **1. Dag 1 – 6**

## **1.1 Dag 1**

Dag 1 er nok den vigtigste dag i hele éns projektforløb. Det er denne dag, som har indflydelse på hele projektet man har foran sig.

Da jeg var alene, var det meget nemt at blive enige om hvad projektet skulle være, hvad det skulle være lavet og skrevet i, og hvad slut målet var. Jeg valgte Unity fordi jeg allerede har erfaring i det, og den bruger C# som sit primære sprog. C# er det sprog jeg har mest erfaring i, og det sprog jeg er mest komfortabel med at skrive i. Jeg ved også at Unity er godt til at bygge hurtige programmer, da det er meget hurtigt at ændre værdier uden for scripts. Unity tillader mig altså at ændre ting meget hurtigt, og i en periode af 4 uger, er det kritisk at have en program til rådighed, som Unity.

Jeg kørte ikke nogen bestemt arbejdsproces. Jeg planlagde meget fra starten af, men var villig til at ændre forventninger som projektet gik.

Jeg startede med at lave en list af Should Have, Would Have, Could Have, og Should not have;

**Should Have:**

Units, turrets, civilisationer, Gold, EXP, passiv EXP, og en shop.

**Would Have:**

Enemy AI, Healthbar, hover på knapper

**Could Have:**

Multiplayer, forskellige højder af units, forskellige sværhedsgrader

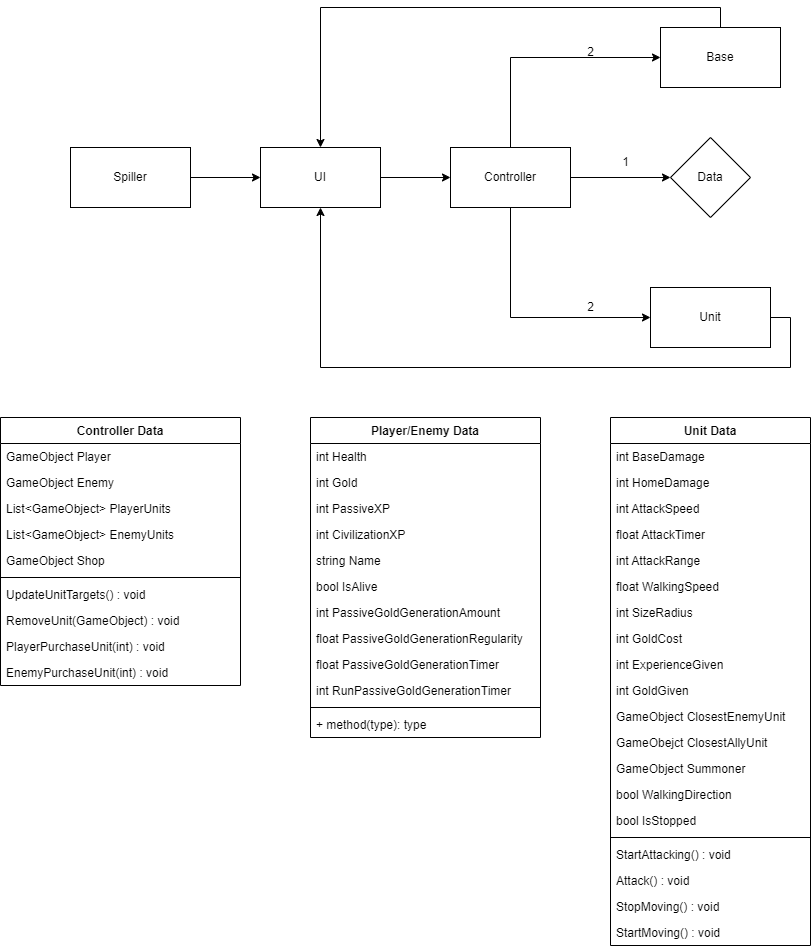
**Should Not Have:**

One-dimensional design.

Denne liste var lavet meget hurtigt, men den giver et godt indtryk i hvor jeg startede. Det meste af hvad jeg endte med at have til sidst, passer med den liste som jeg havde lavet. Men der er nogle få punkter som er meget forskellige fra det endelige produkt. Men på dag 1, var dette hvad der var planlagt.

Jeg lavede en kort graf over, hvordan jeg regnede med at min tid ville blive brugt. Som du kan se, er det delt op i 14 firkanter. Dette var med vilje. Hver firkant er det samme som 2 dage. Dvs. Jeg regnede med at UI i alt ville tage mig 4 dage, dokumentation ville tage mig 4 dage, osv. Jeg havde afsat 6 dage (ca. 20%) til error handling, for jeg ved at der altid er noget der går galt, især inde for IT og programmering.Chart, table

Description automatically generated

Jeg begyndte derefter at lave et diagram, som ville give en overordnede idé om, hvordan det skulle bygges op. Dette diagram var simpelt, men mange af de ting som grafen viser, er noget som jeg stadig har i mit nuværende produkt. Jeg må indrømme at jeg er rimelig stolt over, at jeg kunne ramme så godt når det kom til de forskellige variabler i de forskellige klasser. Der er meget få variabler i mine Player klasser og Unit klasser som jeg har i dag, som ikke var på dette diagram.

Som det sidste begyndte jeg at gøre min logbog klar. Dette gjorde jeg i Excel, og formatet så sådan her ud:



Den øverste firkant, er hvilken dag jeg arbejdede.

Den midterste firkant, er hvor godt jeg synes det gik denne dag.

* Grøn = Gik godt
* Gul = Gik okay
* Rød = Gik dårligt

Den nederste firkant, er hvor meget af min arkitektur jeg skulle ændre. Jo mere jeg skulle omskrive eller ændre, jo mørkere var den. Hver dag vil man nok kunne læse i denne process rapport, at det passer overens med denne logbog.

## **1.2 Dag 2**

Dag 2 begyndte jeg at bygge spillet. Alting gik næsten som det skulle. Indefor denne dag fik jeg gjort, så man kunne spawn en unit, og at de ville angribe spillerens modstander units.

For at teste programmet, have jeg tilføjet 2 shops til UI’en. Den ene shop spawnede units for spiller 1, og den anden shop spawnede units til spiller 2.

Det eneste problem der var, var at jeg havde bygget det meste af min app omkring generaliseret metoder. Dvs. at metoderne til mine knapper ville modtager et tal (UnitID) og et GameObject (Spiller). Men efter noget research og en masse trial and error, fandt jeg ud af, at knapper i Unity kun kan sende 1 variable. Dvs. jeg kan ikke knytte en knap til både et tal og et objekt. Derfor måtte jeg splitte næsten alle metoder op i Player og Enemy metoder. Player metoder skulle bruges af spilleren, når han trykker på knappen. Og Enemy metoderne skulle bruges af fjenden, når den spawner en unit.

(Note: På dette tidspunkt var det planlagt at spilleren skulle spille mod en computer som bare købte units mellem hver 2. og 5. sekund.)

## **1.3 Dag 3**

Dag 3 var okay. Jeg brugte det meste af dagen på at ændre unit stats (Så det var balanceret) og på at tilføje spawn timers på units. Spawn timers betyder, at efter du køber en unit, så går der lidt tid før den spawner. Stærkede units tager længere tid at summon.

## **1.4 Dag 4**

Dag 4 fik jeg playtestet en masse. Der var ikke noget storartet der skete denne dag.

## **1.5 Dag 5**

Dag 5 var starten på hvad jeg kalder ”I Should Have Done Better Planning” periode. Dog der var nogle ting, som jeg ikke kunne have vidst, vil man kunne se senere, at det meste af hvad der skete i denne periode kunne have undgået hvis jeg havde lavet bedre planlægning.

Jeg brugte hele dag 5 på at omskrive mit shop system. Mit mål var at lave shoppen generaliseret til det punkt, at jeg nemt kunne tilføje nye shops uden problemer. Hver gang man opgraderer sin civilisation, får man en ny shop. Denne shop kan indeholde mange forskellige units og civilisationer som andre ikke har.

Jeg lavede et nyt objekt, kaldet DataHandler. Denne objekt indeholdte en Singleton, som Controlleren refererede til. DataHandleren var en form for Database, som indeholdte lister af alle prefabs som spillet gør brug af. Jeg satte det op, så alle shops og units i denne DataHandleren, havde det ID som deres index i listen. Dvs. at hvis en civilisation er på index 3 i civilisation listen, er civilisationens ID 3. Dette er et meget meget meget skrøbeligt system. Dvs. at hvis jeg på noget tidspunkt vil skifte noget ud i denne liste, skal jeg være sikker på at alle civilisationer og units er på den rigtige plads. Derefter skal jeg gå ind i alle shops og ændrer knapperne, så de sender det rigtige ID.

Jeg vidste godt, da jeg lavede dette, at det var forkert. Men det var et valg jeg tog, da jeg ikke havde lang tid til mit projekt, og skulle bruge en hurtig løsning. Havde jeg sat mine mål for høje og var nød til at tage gennemveje? Nej, det vil jeg ikke mene. Men jeg vil bestemt sige at denne del af spillet er klart det mest skrøbelige og slatten kode der er. Hvis jeg havde længere tid, ville jeg havde lav et dictionaries som man kunne slå op i. Hvilket man vil se senere, jeg gør andre steder i spillet.

## **1.6 Dag 6**

Dag 6 var den hårdeste dag i hele forløbet. Jeg startede med at teste spillet, for at se om det virkede som det skulle. Og der var intet der virkede. Man kunne købe units, men de spawnede forskellige steder, de angreb ikke hinanden (Nogle gang gjorde de) og nogle gang mistede de ikke liv. Det var som om der var nogle der havde kastet en håndgranat ind i mit spil. Intet var som det skulle være.

Jeg debugger i ca. 2 timer før jeg fandt problemet. Jeg var begyndt at omskrive mange af mine metoder da jeg troede det var et problem med mit targeting system. Men efter 2 timer, lagde jeg mærke til en lille detalje; Mine prefabs’ data havde ændret sig. Så jeg valgte at køre mit spil endnu en gang, og holde øje med disse prefabs. Og jeg kunne se klart og tydeligt, at prefabsne ændrede sig mens spillet kørte.

Efter dette fandt jeg meget hurtigt ud at problemet.

Når du Instantiator et prefab, bliver der skabt en kopi af dette prefab, i from af et GameObject, som bliver tilføjet til scenen.

Problemet var bare, at jeg ikke tilføjede disse kopier til listen af player units. Jeg tilføjede det prefab, som det var en kopi af. Dvs. hver gang jeg ændrede på nogle player units, ville jeg ikke ændre på kopien. Jeg ville ændre på de prefab som lå i min editor. Chart, bubble chart

Description automatically generated

Graf af beskrevet problem ovenover

Efter jeg fik ændret så jeg tilføjede kopien af min prefab i stedet for min prefab, virkede det hele igen… Efter jeg havde ændret alt dataen på alle prefabs som var blevet ændret.

Jeg havde ligget natten før og tænkt på, hvad mine mål for dette spil var. Jeg kan ikke sige det på en anden måde end ”I had an epiphany ”. Jeg var næsten færdig med det jeg havde planlagt, og rigtig meget af det jeg manglede, skulle ud af vinduet. Derfor satte jeg mig for at lave en ny, endelig liste af krav for mit spil:

**Should Have:**

Units, civilisationer, Gold, EXP, Multiplayer, og en shop.

**Would Have:**

Healthbar, hover på knapper

**Could Have:**

En 3. mulighed i stedet for turrets, singleplayer mode, forskellige højder af units

**Should Not Have:**

One-dimensional design. Turrets.

Der er en kæmpe forskel på den liste jeg startede med, og den nye liste. Især én ting: Multiplayer.

Det var gået op for mig, at spillet ikke ville være særlig interessant hvis man kun kunne spille mod en Ai. Især fordi jeg aldrig nogensinde har lavet nogen form for PvE (Player versus Environment) spil før. Så jeg vidste ikke hvor lang tid det ville tage mig at lave en computer spiller som var intelligent nok til at man kunne spille imod dem, med forskellige sværhedsgrader. Så jeg besluttede mig for at hele spillet skulle være multiplayer.

Jeg havde også bestemt, at turrets var dårlige for spillet. Det kan være rigtig svært at balancere, da en spiller kan komme op og få stærke nok turrets til at ingen units kan komme frem til basen (Age of War har dette problem). På samme tid, hvis de skal være så dårlige så units sagtens kan komme hen til basen, selv hvis du har 4 af dem, så er spørgsmålet om de overhovedet er værd at købe. Så derfor bestemte jeg mig for at det ikke skulle være noget jeg tilføjede.

Der var dog et problem. Jeg har aldrig lavet multiplayer i Unity før. Og nu har jeg valgt at mit spil ikke fungerer uden det. Så jeg skulle i gang med at lave research.

Efter flere timers research, havde jeg kommet frem til, at jeg ville bruge en package til Unity som hedder Riptide. Denne package er open source, og lader mig tilføje komplicerede multiplayer kald, meget enkelt. Efter research, havde jeg fundet at man kunne bruge noget der hedder Mirror. Mange sagde at det var ekstremt nemt at give sig til, men hvis man ville lave noget mere kompliceret, så ville det være rigtig svært at lave.

Jeg vidste ikke hvor kompliceret de snakkede, men efter at tænke mig om, og efter jeg bedømte mine egne evner, valgte jeg Riptide. Jeg vidste ikke hvor kompliceret Riptide ville være, eller hvor kompliceret det jeg skulle bruge, skulle være. Men jeg havde nok tillid til mine evner som programmør, at jeg valgte den jeg kunne tilpasse mest. Dette var ikke nemt valg, og det var et valg som jeg vidste ville have kæmpe indflydelse på de sidste 3 uger.

Jeg begyndte at se en tutorial af ham som havde lavet Riptide, for at tilføje det til min app. Jeg fulgte tutorialen til punkt og prikke. Min controller blev delt op i 2: Controller, og GameHandler. Controlleren står for at sende og hente data fra DataHandleren, og sende data til serveren. Hvor GameHandleren står for at spawn units, ændre deres targets, osv. Før dette punkt havde min Controller gjort lidt af hvert. Dens rolle var aldrig kommet helt på plads, men det var den nu. Controlleren er det eneste der må hente data fra DataHandleren, og Controlleren må ikke referer til andet en DataHandleren eller serveren. Klienten kunne sende beskeder til serveren… Men den sendte ikke beskeder tilbage?

Jeg brugte godt og vel 1.5 timer på dette. Uanset hvor meget jeg tjekker koden igennem, så gav det hele mening.

Step 1. Åben server projektet i en Unity instans

Step 2. Åben spil projektet i en Unity instans

Step 3. Start serveren

Step 4. Start klienten og tryk ”Connect”.

Efter meget lang tid fandt jeg endelig problemet………… Hvis du ikke builder Unity projektet, kører den ikke koden før du ”tabber” ind på Unity instansen igen. Det vil altså sige at hver gang jeg sendte data til Serveren, skulle jeg manuelt klikke over på serveren igen, før den ville køre dens kode.

Der findes ingen ord for de følelser jeg havde på tidspunktet jeg opdagede det.

Jeg fandt en forum som viste hvordan man kunne få Unity til at køre selvom man ikke har buildet det. Det tog 2 sekunder at lave. 2 sekunder. Jeg havde brugt 1.5 timer på at finde ud af problemet, og så tog det 2 sekunder at fikse. Jeg er ikke sur på mig selv. Jeg er sur på Unity. Jeg har aldrig nogensinde følt mig så bedraget i hele mit liv. Jeg troede jeg kunne stole på Unity, men det kan jeg tydeligvis ikke ☹

Efter jeg fiksede problemet, kørte jeg det hele, og det så ud til at virke.

I alt havde jeg arbejdet i over 10 timer på at få tilføjet en basisk måde multiplayer struktur i mit spil. Jeg kunne nu sende og modtage data fra serveren. Jeg var nået lang, men jeg skulle stadig tilføje at den ville spawn units, opgradere civilisationer, osv.

# **2. Dag 10 - 13**

## **2.1 Dag 10**

Dag 10 gik næsten som planlagt. Jeg fik tilføjede multiplayer til resten af appen, hvilket var en meget lang proces. Jeg have lavet 2 metoder for alting i starten, med efter at have lavet multiplayer, kan alting være meget mere generaliseret. Jeg startede med at slette alle Enemy metoder, og derefter omskriv alt logikken så det var mere generaliseret. Jeg fik tilføjet mere backend logic som modtog data fra serveren og reagerede på det.

Jeg havde et problem som jeg sag og debuggede i meget mget lang tid. Af en eller anden grund, spawnede mine units de forkerte steder. Først troede jeg, at jeg kom til at ændre i prefabsne igen, men det var det ikke. Så tjekkede jeg deres andet data, og jeg kunne se at deres summoner, data som indeholdte hvem det købte den unit, var rigtig. Så jeg var vildt forvirret over, hvordan de kan spawn det forkerte sted, men med den rigtige summoner. Deres spawn lokation er nemlig direkte knyttet til deres summoner.

Efter omkring 2 timer fandt jeg endelig problemet. Jeg havde kigget igennem alle linjer kode i hele mit spil, og fandt endelig problemet….

Jeg havde byttet om på 2 variabler på serveren!

Der er nemlig en MEGET vigtig regel når man bruger Riptide: Når du sender data, skal du også modtager dem i samme rækkefølge. Dvs hvis jeg sender en int, string og ushort til serveren, skal den hive daten ud i samme rækkefølge.

Jeg var kommet til at sende UnitID og derefter PlayerID til klienten, mens ved klienten forventede jeg SpillerID og derefter UnitID.

Det forklarede hvorfor det nogle gang virkede. Nogle gang summonede jeg Units med lav nok ID til at det kunne være et PlayerID, go nogle omvendt. Efter jeg fiksede dette virkede alting som det burde.

Udover dette problem, som var meget irriterende, så gik dagen overall meget godt. Jeg fik konverteret hele applikationen om til multiplayer, fik generaliseret hele spillet og fik alting til at virke på 8 timer. Jeg havde forventet det ville tage flere dage, men heldigvis (Ikke heldigt, 100% skill 😊) så havde alt mit groundwork gjort klar til, at det kunne generaliseres hvis det var nødvendigt.

## **2.2 Dag 11**

Dag 11 gik som den skulle. Fik gjort så man kunne opgradere til en ny civilisation og fik tilføjet rigtig mange units som skulle bruges af forskellige civilisationer.

## **2.3 Dag 12**

Brugte ekstremt langt tid på at får 1 label til at opdatere. Dette var endnu et problem med prefabs. Jeg ville tilføje et label som viste prisen på units og civilisationer når du hover din mus over knapperne.

Jeg brugte omkring 5 timer på at løse problemet. Dette skulle have taget få minutter, men endte med at tage en hel dag.

Problemet ligger helt inde i, hvordan Unity som en engine virker.

Jeg havde allerede et problem hvor jeg kom til at referere til et prefab, og ikke dens kopi.

Problemet på dag 12 var, at min Shop indeholdte nogle knapper som var prefabs, og disse prefabs refererede selv til andre shops som også selv var prefabs. Dvs. at et prefab objekt indeholder nogle prefabs, og disse prefabs refererer til nogle prefabs som ligger i et prefab objekt i scenen, som refererer til nogle prefabs.

Jeg har absolut ingen anelse om hvordan jeg skal forklare det så det giver mening, eller vise det på en graf, men er er mit bedste bud:

Diagram

Description automatically generated

Som kan ses på grafen, så refererer dette label til shoppen, men da shoppen er et prefab, skal man passe på, at man ikke refererer til shoppens shops. Hvilket der var problemet. Når disse knapper refererede til mit label, vil de sende et ID. Dette ID ville ikke være ID’et på en Unit, men hvilken knap ID. Metoden ville derefter finde knappen, som i dette tilfælde var et prefab som ikke var instantieret, og vise prisen på det unit som den refererede til…

Ekstremt kompliceret, og jeg har absolut ingen anelse om hvordan jeg endte i dette scenarie. Efter jeg satte mig tilbage og kig igennem hele processen én gang til, lavede jeg noget der var 100x mere simpelt. Knapperne sender et UnitID, og dette ID bruger mit label til at hente en unit fra DataHandleren via controlleren, og så tage dens pris.

Note: Det betyder også at hver gang jeg laver en ny shop, skal jeg ind i dens prefab og sige hvilket ID den skal sende. Hvilket (Igen, meget dårlig arkitektur), at alle shop prefabs teknisk set skal vide et Units ID, selvom de egentlig bare burde sende et ”Hey, jeg er knap 1, spawn den unit, som hører til spillerens nuværende civilisation, og dens civilisations unit 1”. Men dette ville være en kæmpe omskrivning i mit shop system. Hvis jeg kunne gå tilbage i tiden, ville jeg sætte det op rigtigt. For det valg jeg tog i første uge, har gjort at alle ændringer til mine shops tager oceaner af tid. Og det er ikke sidste dag det er et problem.

## **2.4 Dag 13**

Hele dagen tilføjede jeg nye shops, units of civilisationer. Tog 8 timer, men havde derefter 3 units til alle civilisationer, og man kunne nu opgradere sin civilisation op til 4 gange.

# **3. Dag 14 – 19**

## **3.1 Dag 14**

Dag 14 lavede jeg en masse quality of life ændringer. Jeg fik tilføjet healthbar til alle units og spillere, jeg fik ændret alle units farver og deres forme, så de alle så forskellige ud og man kunne kende forskel. Dette tog meget lang tid, men spillet føles meget bedre.

## **3.2 Dag 15 – 19**

Dag 15 til 17 arbejdede jeg på at balancere alle units. Der er stadig nogen der er meget bedre, og nogle som ikke er så gode, men det føles generelt godt at spille.

# **4. Dag 22 – 24**

Jeg brugte disse dage på at skrive rapport, dokumentation, lave grafer mm.

# **5. Konklusion**

Min konklusion for min proces i disse uger er, at selvom jeg vidste hvad jeg ville lave, så burde jeg planlægge *hvad* jeg vil lave. Mit største problem lå helt klart i, at jeg i midten valgte at det skulle være multiplayer, i stedet for at planlægge det for starten. Dog jeg valgte at tilføje multiplayer pga. at jeg havde massere af tid til overs, så kunne jeg have undgået rigtig mange af mine problemer hvis jeg havde planlagt det fra starten.

Chart, histogram

Description automatically generated

Her er en graf over, hvordan mine oplevelser reflekterer, hvor meget jeg har skulle ændre i min applikations, struktur og arkitektur. Som man kan se, så var halvdelen af mine dårligste dag, også dage som jeg lavede kæmpe ændringer min arkitektur. Dage som var okay, havde også nogle dage hvor meget ændrede sig. Og ud af alle de13 dage hvor det gik godt, var det kun 3 af dem hvor jeg lavede ændringer på min arkitektur og struktur. Det er derfor rimelig tydeligt, at jo mindre du skal ændre i din arkitektur og struktur i dit program, jo bedre kommer det generelt til at gå.