Hovedprosjekt:

TITTEL

”KINGDOMS OF CHAOS”

Forfattere:

Jørgen Belsaas
Knut Elvesæter
Anders Elton
Øystein Fladby

Dato:

16. mai 2003

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	"The Kingdoms of chaos"	Nr. : 12 Dato : 16.05.03
Deltakere:	Jørgen Belsaas Knut Elvesæter Anders Elton Øystein Fladby	
Veileder:	Øivind Kolloen	
Oppdragsgiver:	Øivind Kolloen	
Kontaktperson:		
Stikkord (4 stk)	Fantasy, Stratergi, Online, Spill	

Antall sider:48	Antall bilag: 33	Tilgjengelighet (åpen/konfidensiell):

Kort beskrivelse av hovedprosjektet:

Ved hjelp av PHP har vi utviklet et massive online multiplayer spill med en MySQL database som grunnstein, tilrettelagt for at flere tusen spillere skal kunne spille det samtidig. I et multiplayer spill er kommunikasjon mellom spillere en viktig del, så vi har derfor laget bådet et forums og meldingssystem. Vi har utvilket en klient del som man kan spilles ved hjelp av en vanlig web leser og en server del som står på en linux maskin. Vi har designet en database som inneholder all spilldata som også er måten klient og server kommuniserer på.

Spillet er laget i en fantasy setting og inneholder derfor et mørkt design slik at spillerene skal komme i stemmning.

The Kingdoms of Chaos

"The Kingdoms of Chaos " er et hovedprosjekt ved HiG våren 2003.
Prosjekt gruppen består av studenter både fra data- og grafisk-ingeniør studiene.

Jørgen Belsaas 3DP
Ander Elton 3DP
Knut Elvesæter 3GM
Øystein Fladby 3DP

"The Kingdoms of Chaos" er et online strategi-spill. Spillet kan ha inntil 10,000 deltagere. Hver spiller får kontroll ever sin egen provins i et kongedømme. Sammen med sine medspillere skal de bygee opp sitt kongedømme til å bli størst og herske over verden.

Spillet er programmert i henholdsvis PHP rettet mot MYSQL og databaser, JavaScript, CSS og HTML.
På serversiden er det programmert en spillmotor som kjører spillet.

Vi vil takke www.thurmann.net for hosting og selvfølgelig betatesterne!

FORORD

I sitt tredje år av ingeniørutanningen ved Høgskolen i Gjøvik skal studenter data og grafisk linje gjennomføre et hovedprosjekt. Prosjektet skal gjenspeile en arbeidsoppgave man kan få senere i arbeidlivet, og dermed hjelpe studentene å utvilke gode arbeidsvaner og evnen til å ta avgjørelser. Ved utførelse av prosjektet får studentene mulighet til å bruke de kunnskaper de har tilegnet seg i løpet av utdanningen, men også mulighet til å tilegne seg nye.
Oppgaven kan være praktisk, teoretisk eller begge deler. Men den setter fokus på å løse en problemstilling, og hvordan denne løses. Problemstilling, arbeidsutførelse og valg skal rapporteres, og man kan si at prosessen kommer i fokus.

Skolen har nært sammarbeid med flere bedrifter innenfor de bransjene høgskolen utdanner studenter. Flere av bedriftene er flinke til å gi skolen relevante oppgaver som kan brukes til hovedprosjekt. Skolen kommer også med egne prosjekter.
Etter å ha fått presentert disse prosjektene, var det ingen av dem som fa nget vår interesse. Vi bestemte oss derfor for å gå for et egenkomponert prosjekt, og etter samtale med veileder og skolen fikk vi godkjent prosjektet.

Under utviklingen av spillet er det flere personer som har vært til stor hjelp for oss.
Vi synes disse fortjener en takk for hjelpen!

Øivind Kolloen, for veiledning. Han har gitt oss svar på problemer under utvklingen, innspill under koding og en annen vinkling på problemet.

Erlend Rinstad, for hosting på www.thur mann.net. Uten han måtte vi brukt unødige resurser på server og vedlikehold.

Betatestere, takk til alle som har vært med å spille og gitt oss rask tilbakemelding. Takket være mange ivrige sjeler fikk vi raskt lagt til endringer og feilsøkt koden.
Dere vet selv hvem dere er!

Gjøvik 19.05.03

Jørgen Belsaas	Knut Elvesæter
Anders Elton	Øystein Fladby

INNHOLDSFORTEGNELSE:

1. INNLEDNING

1.1. OPPGAVEN

Vi vil lage et webbasert strategispill med et tilhørende GUI. Slike spill finnes det allerede en del av på internett (se vedlegg B - spill) med forskjellige tidsepoker som utgangspunkt, men vi ønsker å lage et fantasy/ middelalderpreget spill. Dette innebærer utvikling av en database for å lagre relevant informasjon om spillet, forum for kommunikasjon og eventuelle diskusjoner, kildekode og GUI som vil danne grunnlag for et fullt brukbart system/spill. I utgangspunktet vil vi utvikle en kjerne for å håndtere de grunnleggende tingene i spillet, hvor vi siden kan utvikle og legge til funksjoner som moduler slik at spillet kan bli mer komplisert, både i spillestil og kildekode.

Spillet skal i første omgang tåle en belastning på opptil ti tusen spillere som spiller mot hverandre i en kunstig/oppdiktet verden hvor denne verden skal forestille å være i middelalderen. Spillerne skal ha mulighet til å kommunisere med hverandre via et forum og via direkte meldinger. GUI'en, som først og fremst skal gi brukerne muligheten til å styre spillet/landet sitt, må være enkel og lett forståelig slik at brukeren skal kunne få spillglede så fort som mulig. Alle spillere skal kunne angripe alle andre spillere, noe som vil si at verdenen det spilles i er forenklet slik at alle har grense til alle.

For å utvikle dette vil vi bruke SQL, PHP, HTML og JavaScript. Dersom det skulle vise seg at dette blir utilstrekkelig, vil vi eventuelt bruke andre språk for å optimalisere kritiske deler. Utviklingen av server delen vil foregå mot en Linux platform der vi kjører Apache som web server og MySQL som database motor. Spillet vil være på engelsk noe som kan begrense målgruppen noe.

1.2. MÅLGRUPPER

1.2.1. FOR PROSJEKTRAPPORTEN

Prosjektrapporten er skrevet for de som vil vite hva vi har laget, hvordan vi har arbeidet for å få dette til og de som måtte ønske å lage noe lignende senere i tillegg til sensor og veileder ved sensur. Leseren bør ha litt forståelse for programmering.

1.2.2. FOR SPILLET

Spillet er laget for personer som forstår skriftlig engelsk, har en interesse for rollespill eller statistikkspill og har tilgang til internett.

1.3. STUDENTENES BAKGRUNN

Vår bakgrunn er blandet fra tredje året på grafisk ingeniør og dataingeniør utdanningene ved Høgskolen i Gjøvik. Noen av oss har programmert mye allerede før de startet med studiene, mens andre har lært alt mens de har gått på skolen. I forhold til spillets type og struktur, er noen av oss godt kjent med lignende spill på internett, mens andre ikke har vært borti slikt før. Dette gir oss en bred bakgrunn der vi selv kan være med og bedømme resultater ut i fra forskjellige synspunkt. Ingen av oss har laget noe lignende før.

1.4. VÅRE ARBEIDSFORMER

I starten hadde vi mange gruppemøter for å komme frem til et godt forprosjekt og en god databasestruktur som vi kunne gå ut i fra. Deretter hadde vi gruppemøter hver uke for å diskutere problemstillinger, sette opp kravspesifikasjon for modulene, fordele oppgaver og passe på at tidsskjemaet ble fulgt så godt som mulig.

Vi har stort sett sittet hver for oss og programmert, men har hatt kontakt over internett og mot slutten har vi også sittet sammen og programmert en gang i uken.

Vi hadde møte med oppdragsgiver / veileder en gang i uken i starten for å diskutere løsninger, men mot slutten av prosjektet har det blitt få slike møter.

De gruppemedlemmer som ikke hadde mye programmeringserfaring fra før måtte selv skaffe seg bøker om emnene (se vedlegg B - bøker) og lære seg med hjelp fra de andre på gruppa. De som ikke hadde vært borti lignende type spill, fikk linker til en del sider der de kunne prøve slike spill (se vedlegg B - spill).

1.5. FORMÅL MED OPPGAVEN

Denne oppgaven lagde vi så å si selv i samarbeid med Øivind Kolloen som gikk med på å være oppdragsgiver. Den ble lagd fordi vi ønsket å programmere og lage et helt system og fordi vi ville ha en motiverende oppgave å jobbe med.

2. KRAVSPESIFIKASJON

2.1. INTRODUKSJON

Kravspesifikasjonen for systemet som helhet kom vi frem til gjennom forprosjektet. Kravene til de forskjellige modulene kom vi frem til på gruppemøtene vi har hatt hver uke der vi har diskutert krav, problemer og løsninger.

2.1.1. KORT OM KRAV TIL SYSTEMET

Det skal være en administrativ del til spillet slik at administratorer kan utføre en del administrative oppgaver. (se pkt 2.3.3.18 admin) Det skal være forskjellige raser, bygninger, formler, forskning, militære enheter og tyveri operasjoner (se pkt 2.3 kravspesifiksjoner). Vi skal ha et forum som benytter seg av maler (se pkt. 2.3.3.7). Vi vil begrense antall spillere til ti tusen, det vil si at alle script skal kunne kjøres uten stor tidsbruk selv ved dette antall spillere. Grensesnittet skal være mest mulig brukervennlig, og det skal utvikles en GUI-standard (se pkt 2.3.3.17 GUI). Det er ønskelig å unngå andre kostnader enn til plakaten som skal leveres inn og noen cd plater for sikkerhetskopi og innlevering.

2.1.2. KORT OM SYSTEMETS OMGIVELSER

Systemet vil bli liggende på internett via en server som står plassert i Norge. Linjen inn/ut fra denne er på 100Mbit og brukes også til noen andre systemer. Serveren skal ha PHP 4.x og MySQL 4.x og kjøre Linux operativsystem.

2.1.3. REFERANSER

For at alle skulle forstå hva slags spill vi skulle lage og hvordan tankegang som ligger bak, brukte vi følgende spill:

Dominion, Dystopia, Planetarion, Earth og Last-Horizon (se vedlegg B - spill for adresser).

For å finne ut av funksjonalitet vi trengte i PHP og MySQL, og dermed hvilke versjoner vi måtte bruke, leste vi dokumentasjon på følgende websider:

MySQL og PHP dokumentasjon (se vedlegg B for adresser)

2.2. BRUKERBESKRIVELSE

2.2.1. SYSTEMETS BRUKERE

Ettersom vi som utviklere selv vil være administratorer, trengs det ingen spesiell opplæring på denne siden. Spillerne skal ha tilgang på en guide og en gjennomgang for å komme i gang og å forstå denne typen spill. I tillegg vil det bli et forum der man kan få hjelp av de andre spillerne.

Det skal være enkelt å legge til nye elementer i spillet som for eksempel bygninger, forskning, tyveri, magi og militære. Vedlikehold av systemet håndteres av en administrativ modul (se pkt 2.3.3.18).

2.2.2. OPERASJON

Det er forventet at serverne har en oppetid på 95 %.

I en feilsituasjon der databaseserveren blir utilgjengelig vil ikke PHP scriptene fungere og brukerne vil få feilmeldinger på skjermen. En e-post link til administratorene skal være tilgjengelig fra forsiden som ikke skal benytte databasen. Dersom webserveren blir utilgjengelig, vil brukeren få en standard ”server ikke funnet” melding.

Ved tap av data, for eksempel ved harddisk feil, skal siste sikkerhetskopi gjenopprettes så fort serveren er klar for dette.

Serveren har for øyeblikket ikke *SSL* og er derfor ikke helt sikker i bruken av forms. Scriptene skal være sikret mot skadelig input fra brukere.

2.2.3. ASPEKTER OMKRING LIVSSYKLUS

Systemet bør være enkelt å flytte fra en server til en annen uten endringer i koden. Slik kan også nye moduler kan testes på andre servere før de legges ut på spillserveren ved å kopiere kildefilene til serveren som skal brukes til testing.

Alle rapporter skal skrives på norsk. En brukerguide skal skrives på engelsk og kommentering av kode skal også skrives på engelsk. Koden skal være godt kommentert, med et minimum på å ha kommentert hva hver funksjon gjør.

Systemet skal enkelt kunne utvides med flere bygninger, formler, tyverioperasjoner eller forskninger.

Vedlikehold skal begrense seg til å måtte resette spillet med jevne mellomrom. Det er ønskelig at utviklerne er innom sidene en gang i blant også etter at prosjektet er ferdig slik at de kan svare på eventuelle spørsmål som brukerne har lagt i forumet.

Testing av nye moduler bør foregå på en annen server etter at spillet er lagt ut for betatesting. Denne betatestingen vil være tilstrekkelig testing for å simulere spillets bruk også etter prosjektinnlevering.

2.2.4. YTELSE

En bruker skal ikke behøve å vente mer enn maksimum to sekunder på at en webside skal prosesseres på serveren ved stor belastning. Det eneste unntak er tidsrommet da serverscriptet kjøres hver time. Tiden for å laste websiden ned til klienten vil avhenge av internett tilkoblingen klienten bruker, men alle bilder og script skal utformes med tanke på at brukerne ikke skal behøve å vente på hver side.

2.2.5. BEGRENSNINGER

2.2.5.1.HARDWARE

Serveren for websidene er en gammel P3 800Mhz som i tillegg tjener ganske mange andre sider / systemer. Serveren for databasen tjener også mange andre databaser som bruker mye ressurser. Dette kan legge begrensninger på hvor mange SQL spørringer og hvor kompliserte de er i servermodulen for å få denne til å kjøre fort nok. Nettverkskort på 100Mbit bør likevel ikke sette en begrensning for oss dersom vi har opptil ti tusen spillere.

2.2.5.2.SOFTWARE

Sidene skal fungere i MS Explorer 5.x og Opera 6.x.

2.2.6. ANTAGELSER

Vi antar at serveren vil bli oppgradert fra mySQL 3.23 til 4.x så fort som mulig.

2.3. DETALJERT KRAVSPESIFIKASJON

2.3.1. FUNKSJONELL STRUKTUR OG TVERR-RELASJONER

Databasen skal utvikles og lages etter hvert som modulene tar form. Alle script skal ta hensyn til at det kan komme flere felter i tabellene ettersom modulene kommer på plass. En oversikt over databasen finnes i vedlegg EE. Modulene skal benytte hverandre for å gjøre databasekall til de forskjellige delene av systemet. Det vil si at Serverscriptet skal benytte funksjoner i for eksempel bygnings-, tyveri- og forskningsmodulene for å hente ut effekter og egenskaper. På samme måte skal bygnings-, militær-, kamp-, magi-, forsknings- og tyverimodulenes funksjoner benyttes av hverandre for å hente ut nødvendig data. Dette krever et godt samarbeid ved utvikling av grensesnittet mellom modulene.

2.3.2. OVERORDNEDE OPERASJONELLE SYSTEMKRAV

2.3.2.1.MODI OG KONTROLL

En person som kommer til sidene defineres som en gjest til han har logget seg på. En gjest kan logge seg på som enten administrator eller som en vanlig bruker, ergo har vi i utgangspunktet tre modi en person som er innom siden kan befinne seg i. Hvilken modus man er i skal kontrolleres på hver side.

2.3.2.2.YTELSE

Modulene bør ikke bli så komplisert at de har merkbar innvirkning på ytelsen for brukeren utover det som står skrevet i pkt. 2.2.4.

2.3.2.3.TILGJENGELIGHET

Modulene skal ha en oppetid på 99 % av serverens oppetid etter at de er ferdigstilt. Den tiden de ikke er oppe, vil være tid der administratorene er inne og ordner små feil i scriptene. Se også pkt. 2.2.2.

2.3.2.4.FEILRAPPORTERING

Dersom små logiske feil oppstår, og brukeren oppdager dette, skal han ha muligheten til å rapportere dette til administratorene ved å benytte forumet. Dersom større feil oppstår slik at PHP parseren ikke klarer å forstå koden, blir feilen enten skrevet til skjerm eller til en fil alt etter hva serveren er konfigurert til. Den nåværende serveren skriver til skjerm.

2.3.2.5.GJENERVERVELSE ETTER FEIL

Dersom feilen er stor, vil man kunne utnytte siste sikkerhetskopi av databasen etter å ha ordnet feilen. Ellers vil spillet kunne fortsette bare feilen er rettet opp.

2.3.2.6.SIKKERHET

Modulene skal forsikre seg om at kun påloggede brukere kan vise sidene. De skal benytte HTTP protokollens "post" metode i alle forms i HTML koden for å hente informasjon som skal lagres eller oppdateres i databasen. Hvor sikker denne metoden er avhenger av om serveren har kryptering som for eksempel SSL (se vedlegg A) eller ikke.

2.3.3. FUNKSJONELLE KRAV

Dersom ikke annet er oppgitt, gjelder punktene under pkt. 2.3.2 for alle modulene.

2.3.3.1.FUNKSJONELLE KRAV TIL DATABASEMODUL

2.3.3.1.1. INPUT

Database modulen tar imot brukernavn, passord, host (IP adressen til databasen) og databasenavnet til databasen den skal koble seg opp mot. I tilegg skal den kunne ta i mot SQL spørringer og sende dette videre til databasen.

2.3.3.1.2. PROSESSERING

Det skal være metoder for å koble seg til og fra en database og sørge for at resultatet av spørringene den utfører blir lagret riktig for senere bruk. I tilegg skal det være mulig å sjekke hvor mange resultat spørringen ga. Det skal også være feilkontrollering, slik at det er mulig å finne ut om databasen rapporterte en feil på en SQL spørring.

2.3.3.1.3. OUTPUT

Modulen må kunne sende resultatet av spørringer tilbake til bruker. I tilegg skal det være en mulighet for å hente eventuelle feilmeldinger som databasen rapporterer.

2.3.3.1.4. YTELSE

I og med at databasen er grunnsteinen for alt som skjer er det viktig at databasemodulen ikke inneholder unødvendige sjekker som senker prosessering. Feilsjekking kan selve databasen ta seg av.

2.3.3.1.5. FEILRAPPORTERING

Modulen skal rapportere feil om databasen er nede og feil på spørringer.

2.3.3.1.6. SIKKERHET

Modulen kobler seg til databasen ved hjelp av brukernavn og passord. Det trengs ikke å gjøre noen spesielle sikkerhetshensyn med tanke på

dette.

2.3.3.2.FUNKSJONELLE KRAV TIL SERVERMODUL

2.3.3.2.1. INPUT

Ingen input.

2.3.3.2.2. PROSESSERING

Servermodulen skal lese inn konfigurasjonen til spillet i fra databasen. Her ligger det data om spillet som hovedsakelig styres ifra admin modulen. Modulen skal ta seg av progresjonen i spillet og skal derfor kjøres hvert tick (definert som 1 – èn - time). Modulen skal laste inn alle klassene i spillet og kjøre doTick funksjonen deres. I tilegg skal serveren inneholde ytelsestester slik at det er mulig å se hvilke klasser som krever mye prosessering.

2.3.3.2.3. OUTPUT

Modulen skal skrive ytelsestestene til en fil.

2.3.3.2.4. YTELSE

Serverskriptet skal kjøres kontinuerlig og kommer til å kreve mye av databasen mens det kjøres. Under dette tidsrommet vil spillere kunne oppleve heng eller forsinkelser i spillet. Det er derfor viktig å redusere antall databasekall til et minimum slik at web scriptene når fram med sine forespørsler.

2.3.3.2.5. FEILRAPPORTERING

Modulen skal rapportere feil til fil eller database.

2.3.3.2.6. GJENERVERVELSE ETTER FEIL

Om feil oppdages er det stor sannsynlighet for at det er en logisk feil. Feilen bør rettes så fort som mulig da slike bugs kan skape store ubalanser i spillet.

2.3.3.2.7. SIKKERHET

Servermodulen skal kjøre på serveren i en evig løkke. Det er viktig at riktig bruker kjører slik at modulen blir kjørt med riktig rettigheter.

2.3.3.3.FUNKSJONELLE KRAV TIL FORSKNINGSMODUL

2.3.3.3.1. INPUT

Forskningsmodulen skal ta i mot ønsket valg fra brukeren når det gjelder hvilken forskning han vil forske frem.

2.3.3.3.2. PROSESSERING

Modulen skal sjekke om brukeren faktisk kan forske frem det han prøver på. Deretter skal det sjekkes om brukeren har nok ressurser for å betale for forskningsoppdraget.
Det skal også tilbys et interface slik at det blir lett å implementere

effektene av forskningen inn i de andre klassene.
På serversiden skal det tilbys en mulighet for å kjøre effektene av all forskningen på de provinsene som har forsket dem frem.

2.3.3.3.3. OUTPUT

Brukeren skal få beskjed om hvilke forskninger han allerede har og prisen og tidsbruk på de nye han kan forske frem. Om han har en science under forskning skal ha få opp en beskrivelse av hva denne gjør og hvor lang tid det er før denne blir fullført.

2.3.3.4.FUNKSJONELLE KRAV TIL TYVERIMODUL

2.3.3.4.1. INPUT

Tyverimodulen skal kunne ta i mot brukerens valg av kongedømme, provins og tyveri operasjon.
Det skal være minst fire spionasje operasjoner og minimum to ødlegge operasjoner.

2.3.3.4.2. PROSESSERING

Modulen må kunne sjekke at brukeren har nok ”influence”, tyver og forskning til å kunne utføre tyveri operasjonene. Den må deretter regne ut sjansen for at operasjonen er vellykket. Det skal være forskjellig vanskelighetsgrad på operasjonene slik at skade operasjoner er vanskeligere å utføre enn spionasje operasjonene. Suksessen skal også avhenge av hvor mange tyver/banditter man har i forhold til sitt eget land og i forhold til motstanderens tyver.
Om man lykkes skal operasjonen gå som planlagt, men om man misslykkes skal man miste noen tyver og det skal være en viss sjanse for at tyvene som mistes sladrer på hvem som sendte dem.
Modulen må også passe på at provinser som er nyskapte og er i ”protection” ikke kan berøres med tyveri av andre provinser.

2.3.3.4.3. OUTPUT

Brukeren skal få beskjed om hvordan den utførte operasjonen gikk. Spionasjeoperasjonene skal gi ganske nøyaktig informasjon om hva motstanderen har. Han skal også få vite hvor mye skade ødelegge operasjonene gjør. Om operasjonen feiler skal han få beskjed om dette, men han skal ikke få beskjed om tyvene sladrer på han.

2.3.3.5.FUNKSJONELLE KRAV TIL MAGIMODUL

2.3.3.5.1. INPUT

Magimodulen skal kunne ta i mot brukerens valg av kongedømme, provins, formel, antall magikere og evt. antall tikk formelen skal være aktiv.
Brukeren skal også kunne stoppe aktive formler som han selv har kastet.
Han skal kunne velge mellom minst tre formler som kan kastes på ham selv og minst fem som kan kastes på andre provinser.

2.3.3.5.2. PROSESSERING

Modulen må kunne sjekke at brukeren har nok ressurser og forskning til å kunne kaste formelen og at input er riktig formatert. Den må deretter regne ut en viss mulighet for at kastingen av formelen lykkes. Dette bør gjøres ved noe bruk av tilfeldige tall og noe bruk av kasterens data sammenlignet med målets data. Dersom kastingen lykkes og det er en langvarig formel, skal formelen legges inn i Spells tabellen i databasen. Modulen må skille mellom formler som utføres direkte og formler som blir liggende og være aktive i en periode. Den bør også skille mellom vennlige og fiendtlige formler. Servermodulen (se vedlegg T - server) tar seg av egenskapene til de langvarige formlene og å slette dem når de er gått ut.

2.3.3.5.3. OUTPUT

Brukeren skal få informasjon om hvor mye ressurser han må bruke for å kaste formelen, en beskrivelse av formelen og informasjon om hvordan det gikk i form av en nyhetspost til både kasterens provins og målets provins og en liten beskjed direkte til skjermen. I tillegg skal formler som er blitt kastet både av og på brukeren vises med antall magikere, hvem den er kastet på og hvor lenge den er aktiv.

2.3.3.6.FUNKSJONELLE KRAV TIL BYGNINGSMODUL

2.3.3.6.1. INPUT

Bygningsmodulen skal gi brukeren mulighet til å bygge nye bygninger og å ødelegge bygninger han allerede har bygd. Det skal være minst ti forskjellige bygningstyper med forskjellige egenskaper.

2.3.3.6.2. PROSESSERING

Modulen må kunne sjekke at brukeren har nok ressurser og forskning til å kunne bygge denne bygningen og at input fra bruker er riktig formatert. Den må deretter legge bygningen inn i databasen. Dersom en bygning skal slettes, må modulen sjekke at brukeren har dette antall av denne bygningen og deretter slette dette antall bygninger fra databasen. Servermodulen (se vedlegg T - server) tar seg av selve utviklingen og de egenskapene bygningene har.

2.3.3.6.3. OUTPUT

Brukeren skal få informasjon om hvor mye ressurser han må bruke for å bygge bygningen og en beskrivelse av bygningen. Han skal få en feilmelding dersom han ikke har nok ressurser til å bygge alt han har satt opp. Han skal få informasjon om egenskapene til en bygning, hvor mange han har av en bygning og hvor mange bygninger han har under bygning.

2.3.3.7.FUNKSJONELLE KRAV TIL FORUMMODUL

2.3.3.7.1. INPUT

Forumet skal gi administratorer, brukere og gjester mulighet til å skrive innlegg og svar i et forum alle kan lese, verdensforum. En gjest skal kunne skrive inn navn mens en pålogget bruker eller en administrator automatisk skal få sitt navn på innlegget. Det skal ikke være mulig for brukere og gjester å skrive html kode, men administratorer skal kunne gjøre dette. Administratorer skal kunne slette innlegg fra dette forumet.

Administratorer og brukere skal i tillegg ha tilgang til å lese og skrive innlegg til et kongedømmeforum. Brukeren skal kun ha tilgang til sitt eget kongedømme og verdensforumet, mens administrator skal kunne velge blant alle kongedømmene eller verdensforumet. Administratorer og de enkelte kongene i kongedømmene skal ha mulighet til å slette innlegg fra de respektive kongedømmene.

2.3.3.7.2. PROSESSERING

Modulen må sjekke om personen som kommer til siden er pålogget som bruker eller administrator. Er personen ikke pålogget som noen av delene blir han / hun en gjest.

Forumet skal benytte seg av maler, det vil si at all html kode skal befinne seg i egne filer, mens all PHP kode skal befinne seg i andre filer. Dette skal gjøres ved å lage en klasse som tar seg av koblingen mellom PHP og HTML koden.

Forumet skal passe på at bare administratorer får lov til å skrive html kode.

Dersom en gjest skriver et innlegg, skal navnet lagres sammen med all annen informasjon i Forum tabellen i databasen. Dersom en bruker skriver et innlegg, skal ikke navn, men provins id lagres i tabellen. For en administrator skal en reservert id brukes.

2.3.3.7.3. OUTPUT

Forumet skal ha en hovedside med tråder. Ved å trykke på disse trådene, skal man få opp alle innlegg som er blitt skrevet til denne tråden.

Alle tråder og innlegg fra gjester skal markeres slik at man vet de er gjester og ha med navnet de har oppgitt. Alle fra brukere skal vise brukernavnet, provinsen og kongedømmet til brukeren. Alle fra administratorer skal markeres slik at andre ikke kan late som at de er administratorer.

Dersom kravene til input ikke er oppfylt, skal en feilmelding vises på skjermen.

2.3.3.7.4. FEILRAPPORTERING

Dersom feil oppstår slik at modulen ikke klarer å forstå koden, blir feilen enten skrevet til skjerm eller til en fil alt etter hva serveren er konfigurert til. Den nåværende serveren skriver til skjerm. Det skal finnes en link for å sende e-post til administratorene dersom brukerne finner feil ved forumet.

2.3.3.8.FUNKSJONELLE KRAV TIL MELDINGSMODUL

2.3.3.8.1. INPUT

Brukeren skal kunne velge kongedømme og provins han vil sende en direkte melding til. Deretter skal han kunne skrive en tekst og sende den. Meldinger skal også kunne slettes.

2.3.3.8.2. PROSESSERING

Brukerens melding skal sjekkes for html kode og legges inn databasen. Dersom en mottaker leser en melding, skal den oppdateres som lest i databasen. Dersom en mottaker eller sender sletter en melding, skal dette oppdateres i databasen, og hvis begge har slettet meldingen skal den slettes fra databasen også.

2.3.3.8.3. OUTPUT

Brukeren skal se hvor mange uleste meldinger han har og kunne liste opp alle meldinger som han har sendt eller fått og ikke har slettet. Ved å klikke på meldingen, skal han kunne lese den.

2.3.3.9.FUNKSJONELLE KRAV TIL MILITÆRMODUL

2.3.3.9.1. INPUT

Militærmodulen skal kunne ta imot antall militære brukeren ønsker å trene

2.3.3.9.2. PROSESSERING

Ressurser brukeren har må kunne sjekkes slik at det er mulig å avgjøre om det er nok ressurser til å trene de forskjellige militære. Modulen må også kunne sende militære til trening hvis det er nok ressurser.

2.3.3.9.3. OUTPUT

Brukeren må få se hvor mange militære han til enhver tid har, samlet antall hver type, og hvor mange som er tilgjengelige(ikke ute i krig). Brukeren må få se en bekreftelse på at militære ble sendt til trening – hvor mange, og hvilken militærenhet. Hvis det ikke er nok ressurser eller nok folk å trene fra, må brukeren få beskjed om dette. Modulen må også skrive ut hvor lenge det er til soldatene er ferdig trent.

2.3.3.10. FUNKSJONELLE KRAV TIL KAMPMODUL

2.3.3.10.1. INPUT

I denne modulen skal brukeren ha mulighet til å gi opplysninger om hvilket kongedømme og hvilken provins innen det kongedømmet han har lyst til å angripe. Det skal også være mulighet for å skrive inn antall militære som skal sendes i krig.

2.3.3.10.2. PROSESSERING

Modulen må kunne forsikre seg om at kongedømme og provins eksisterer i tabellen. Og det skal ikke være mulig å angripe seg selv. Det skal ikke være mulig å angripe nylig opprettede provinser. Deretter må må modulen regne ut angrep og forsvar, og legg til eventuelle bonuser. Angripende part må kunne drepe militære i forsvarende provins, og forsvarende provins må kunne drepe soldater i angripende provins. Angripende provins skal også kunne ødelegge bygninger og stjele land, hvis den vinner.

2.3.3.10.3. OUTPUT

Provins som ble angrepet får beskjed om at det har vært ett angrep, og om hva utfallet ble. Begge involverte kongedømmer får en liten notis om krigen, og den som angrep får bekreftelse på antall sendte soldater, og en beskjed om utfall og eventuelle tap.

2.3.3.11. FUNKSJONELLE KRAV TIL REGISTRERINGSMODUL

2.3.3.11.1. INPUT

I denne modulen skal kunne ta imot disse brukerdata: navn, fødselsdato, land, e-post adresse. Skal også kunne ta imot disse spilldata: brukernavn, rase, kjønn(på hersker), ønsket navn på provins, ønsket navn på hersker

2.3.3.11.2. PROSESSERING

Modulen skal forsikre seg om at alle felter er utfylt, skal sjekke om e-post adressen er på riktig format (xx@yy.zz), sjekke at ingen felt blir fylt ut med for mye tekst. Sørge for at det ikke er mulig å få lagt javascript, html eller php kode inn i databasen fra denne modulen. Modulen må forsikre seg om at grensen på 10 000 (ti tusen) spillere ikke blir overskredet, og at provinsen blir opprettet og riktig konfigurert slik at brukeren kan starte å spille

2.3.3.11.3. OUTPUT

Et skjema som bruker kan registrere seg via. Bekreftelse på at registrering er fullført, og en e-post til brukeren med informasjon om den nye provinsen som er opprettet og brukerens brukernavn og passord

2.3.3.12. FUNKSJONELLE KRAV TIL INN/UTLOGGINGSMODUL

2.3.3.12.1. INPUT

Modulen skal ta imot brukernavn, passord og cId cookien.

2.3.3.12.2. PROSESSERING

Scriptet skal opprette standardobjekter som også nyttes av de senere scriptene. Den skal opprette et databaseobjekt og koble seg opp til databasen. Den skal også opprette et brukerobjekt og sjekke om

brukeren er markert som logget inn. Om bruker ikke er markert som logget inn i bruker objektet skal scriptet avsluttes. Om brukeren er markert som logget inn skal et provins objekt opprettes og standard data skal hentes ut. Om provinsen ikke er markert som å være i live skal scriptet avsluttes.

2.3.3.12.3. OUTPUT

Dersom brukeren ikke klarer å logge seg inn, eller blir logget ut på grunn av tidsavbrudd, skal han få melding på skjermen om dette. Skulle innloggingen gå i orden, vil han få vist startsiden i spillet. Dersom brukeren har blitt drept eller slettet på grunn av misbruk eller inaktivitet, vil han også få melding om dette på skjermen

2.3.3.12.4. SIKKERHET

Modulen skal sørge for at scriptet avsluttes for alle som ikke har tilgang til videre kjøring av det. Det vil si brukere som ikke er logget inn og provinser som ikke er i live.

2.3.3.13. FUNKSJONELLE KRAV TIL NYHETSMODUL

2.3.3.13.1. INPUT

Nyhetsmodulen må kunne ta imot et tidspunkt for når nyheten ble lage/skjedde. Den må også kunne ta imot hvor den skal sendes, og den må selvsagt kunne ta imot selve nyheten. Må også kunne ta imot en ID som sier noe om hvem sine nyheter som skal hentes ut

2.3.3.13.2. PROSESSERING

Modulen må forsikre seg om at mottaker av nyhet faktisk eksisterer. Sørge for at tekst blir lagt riktig inn i database, også tekst som inneholder apostrof. Når det blir bedt om å få se nyheter som er sendt til en mottaker må også da databasen forsikre seg om at denne oppgitte mottaker faktisk eksisterer i databasen. Modulen skal også kunne slette alle nyheter eldre enn en hvis dato

2.3.3.13.3. OUTPUT

En liste med alle nyheter som er sendt til bruker

2.3.3.14. FUNKSJONELLE KRAV TIL UTFORSKNINGSMODUL

2.3.3.14.1. INPUT

Denne modulen skal kunne ta imot input om hvor mange soldater som skal sendes ut for å utforske land.

2.3.3.14.2. PROSESSERING

Sjekke om bruker har nok ressurser til å sende det gitte antall militære ut for å forske ut verden. Sjekke om brukeren har nok soldater av type rekrutt. Regne ut hvor masse land brukeren vil få når alle soldatene er ferdige med å utforske.

2.3.3.14.3. OUTPUT

Skrive ut bekreftelse på hvor mange soldater som ble sendt til å forske ut land. Skrive ut hvor lenge det er igjen til land er ferdig utforsket.

2.3.3.15. FUNKSJONELLE KRAV TIL POLITIKKMODUL

2.3.3.15.1. INPUT

Modulen skal ta i mot brukerens valg av konge/dronning. Hvis brukeren er den som er blitt valgt konge/dronning skal han/hun ha ekstra valg. De skal da kunne endre navnet på kongedømmet og legge til en banner.

2.3.3.15.2. PROSESSERING

Scriptet skal oppdatere databasen med hvem som har stemt på hvem, hva navnet på kongedømmet er og hva adressen til banner-bildet er. Den skal regne ut antall stemmer hver provins har fått og ut fra dette sjekke om brukeren er blitt valgt konge/dronning og oppdatere dette i databasen. Dersom en ny konge/dronning blir valgt, skal dette sendes som en nyhet til alle i kongedømmet.

2.3.3.15.3. OUTPUT

Brukeren skal kunne se alle provinsene i kongedømmet sitt, hvor mange stemmer den enkelte har fått og hvem de har stemt på. Dersom brukeren er konge/dronning skal valgene for å endre navn og banner bli vist, samt et bilde av et tronrom.

2.3.3.16. FUNKSJONELLE KRAV TIL STATISTIKKMODUL

2.3.3.16.1. INPUT

Brukeren skal kunne velge å se enten statistikk på ett og ett kongedømme eller en topp femti liste over provinser. Han skal kunne angi hvilket kongedømme han vil se statistikk for når det vises ett og ett kongedømme.

2.3.3.16.2. PROSESSERING

Modulen skal kun hente ut og vise informasjon fra databasen.

2.3.3.16.3. OUTPUT

Brukeren skal på topp femti siden kunne se de femti største provinsene med litt statistikk fra dem. På statistikksiden for hvert kongedømme, skal det også vises enkel statistikk fra hver provins. Fra de forskjellige statistikksidene skal det være linker mot angreps, magi og tyverisiden.

2.3.3.17. FUNKSJONELLE KRAV TIL GUI

2.3.3.17.1. LAYOUT

Alle sidene skal være oversiktlig og brukeren skal lett få oversikt over de ulike elementene på siden. Tomrom, luft rundt tabeller, skjemaer eller annen informasjon skal fylles opp av bakgrunnsfargen eller med bilder som er representativt for den enkelte siden.

Bildene skal bygge opp en atmosfære eller skape en stemming slik at brukeren får opplevelsen av at dette foregår i en middelalderepoke.

2.3.3.17.2. INPUT

All input fra brukeren foregår via skjemaer eller linker. Ellers foregår navigasjonen på siden via en meny på venstre side. Fra denne har brukerne tilgang til nesten alle sidene.

2.3.3.17.3. DESIGN

Designet skal være enkelt og oversiktlig. Det skal ikke være for mye informasjon/elementer som trekker oppmerksomheten vekk fra det viktige, altså spillerens valg/input.

Det skal brukes CSS for å få likt utseende på like elementer

2.3.3.17.4. BRUKERVENNLIGHET

For nye spillere skal det lages både en guide og en tutorial (se pkt. 2.3.3.19).

Skulle brukeren trenge mer veiledning finnes både forum internt i kongedømmet og for hele spillet (se pkt. 2.3.3.77) og en meldingstjeneste (se pkt. 2.3.3.88) der brukeren kan komme i kontakt med andre spillere for å få hjelp.

Inne i selve spillet vil alle input-felter og knapper har en tittel slik at en liten hjelpetekst kommer opp om brukeren holder musepekeren over feltet. Det er også en beskrivende tekst for hver side. Ellers er valgene brukeren har begrenset slik at han vil komme i mål

2.3.3.18. FUNKSJONELLE KRAV TIL ADMINMODULEN

2.3.3.18.1. INPUT

Modulen tar i mot ønsket valg fra bruker.

2.3.3.18.2. PROSESSERING

Modulen skal kunne stenge, slette og åpne kontoer. Det skal også være mulighet for å se hvor lenge siden en bruker er pålogget eller ikke. Det skal være mulig å restarte eller stoppe spillet. Det skal gå an å legge til nyheter og gå inn i forumet som administrator. Man skal kunne legge inn nye Thievery, Magic, Building og Science klasser. Adminmodulen skal sjekke om klassefilene eksisterer før den legger til klassen i spillet.

2.3.3.18.3. OUTPUT

Skrive ut om ønsket operasjon var vellykket eller ikke, med eventuelle feilmeldinger.

2.3.3.18.4. YTELSE

admin modulen brukes sjelden og kan derfor inneholde tunge søk, hastighet er ikke kritisk.

2.3.3.18.5. TILGJENGELIGHET

Adminmodulen bør være oppe hele tiden mens spillet kjører, da dette er den eneste måten å slette spillere på.

2.3.3.18.6. SIKKERHET

Passe på at kun administrator har tilgang til menyen. Verdier som legges inn skal feilsjekkes først, slik at feil verdier ikke blir lagt inn i et aktivt spill.

2.3.3.19. FUNKSJONELLE KRAV TIL GUIDE OG TUTORIAL

2.3.3.19.1. GUIDE

Guiden skal være en kort og grei beskrivelse til spillets sider/elementer. Undersider med detaljert informasjon

om blant annet militærenheter, bygninger og magi skal være tilstede. Den skal ha linker til sider med detaljert informasjon om militærenheter, bygninger, kunnskap, magi og tyveri.

2.3.3.19.2. TUTORIAL

Tutorialen skal gi spillere innføring i spillets gang. Spesielt nye spillere skal ha nytte av tutorialen, med tanke på registrering og innlogging. Den skal inneholde skjermbilder av alle sidene og vise spilleren hva han skal gjøre på hver side.

2.4. BEGRENSNINGER

2.4.1. SOFTWARE DESIGN BEGRENSNINGER

2.4.1.1.SOFTWARE STANDARDER OG SPRÅK

Kildefilene skal utformes etter maler og standarder i vedlegg F.
Møterapporter skal utformes etter mal vist i vedlegg H.
Arbeidslogger skal utformes etter mal vist i vedlegg E.

2.4.1.2.SOFTWARE STANDARDER OG SPRÅK

En internettoppkobling og ftp og telnet/ssh grensesnitt er alt som trengs for å kommunisere tilstrekkelig med serveren.

2.4.1.3. DATABASE

Det skal brukes MySQL versjon 4.0 eller bedre, men serveren er i utgangspunktet versjon 3.23. Denne vil bli oppgradert. Grunnen til at vi vil bruke versjon 4.x er at man da kan utnytte nøstede spørringer og dermed la databasemotoren ta seg av mer prosessering når oppdateringer skal utføres.

2.4.1.4.OPERATIVSYSTEM

Serversystemet skal fungere på Linux i første omgang, men siden PHP og MySQL stort sett er plattformuavhengig vil det ikke trenge store forandringer å legge det over til en server med et annet operativsystem.

Klienten trenger ikke et spesielt operativsystem da hele systemet ligger på serversiden.

2.4.1.5.TOLERANSER, MARGINER OG MULIGHETER/TILFELLER

Med vår begrensning på ti tusen spillere, bør det ikke være noe som begrenser kapasiteten. Dersom alle brukerne utfører spillkommandoer som krever databasekall samtidig som serverscriptet kjører, kan det bli merkbare forsinkelser på hver sidevisning.

2.5. UTGIVELSER UNDERVEIS

Vi vil prøve å gi ut en betaversjon av spillet før alle modulene er ferdige. Denne betaversjonen vil så få påbygginger uten at spillets gang stoppes. På denne måten vil vi ha en spillbar, men uferdig utgivelse ganske tidlig, og deretter bare ha oppdateringer. Spillerne skal ikke behøve å gjøre noe spesielt ettersom disse oppdateringene kommer ut ettersom hele systemet ligger på serversiden.

3. DESIGN

Grunntanke:

Systemet tenkes oppbygd av moduler, hvor hver modul skal jobbe mot en bestemt del i spillet. Det vil si at man har en modul for bygninger, som da naturlig nok tar seg av det som har med bygninger å gjøre, og for eksempel vil alt som har med magi å gjøre, bli lagt i en annen modul. Jobben til hver enkelt modul vil være å presentere de registrerte spilldata som antall soldater, bygninger, ressurser og antall mål med land. Hver enkelt modul skal også ta imot informasjon fra bruker om hva han vil gjøre i spillet, for så å regne ut nye spilldata basert på denne informasjonen og informasjon som allerede er i databasen. Spilldataene lagres deretter i databasen. Hver modul skal også sørge for å ta sin del av oppgaven med å drive spillet fremover(flytte alle "brikkene" frem til neste runde). Disse modulene vil bli realisert som klasser i et objektorientert miljø.

Database:

I databasen jobbes det rundt en hovedtabell, provinstabellen. Det er i denne tabellen hovedinformasjon om bruker lagres. Rundt provinstabellen er det flere andre tabeller som inneholder data som militærtyper, bygningstyper og raser. Det er også tabeller som binder disse tabellene sammen med provinsen slik at en provins kan være en og bare en rase, mens en rase kan være i flere provinser. Fullstendig beskrivelse av hver tabell ligger i vedlegg L.

I militær modulen lagres alle egenskaper om de militære i databasen, mens i bygningsmodul, magimodul, tyverimodul og forskningsmodul lagres det bare et klasse filnavn i databasen. Kampmodul og utforskingsmodul bruker data fra militærmodulen, fordi det er militæret som blir brukt til disse oppgavene. Alle de andre modulene lagrer informasjonen de trenger i databasen.

Det er også tre avskilte tabeller som hører til adminmodulen, slik at man kan legge til adminbruker og sørge for autentisering av personer i admindelen.

Det er en egen modul for databasen slik at ingen andre moduler kommuniserer direkte med databasen, men bruker modulen for databasetilgang til å kommunisere med databasen.

Grunnpilaren som driver spillet fremover er et script på serversiden. Dette scriptet tar seg av oppdatering av databasen slik at spillet går fremover. Dette gjøres en gang hvert tikk. Planen er å la dette scriptet samarbeide med hver enkelt modul for å generere nye verdier for de spilldata produsert av modulen. Moduler kan også generere data som har innvirkning på oppdatering av andre data. Dette gjør det veldig viktig at moduler kan samarbeide med serverscriptet for å oppdatere disse. Som et eksempel kan man ta bygningsmodulen der bygninger skal gi bonuser i befolkningsvekst og i hvor mye ressurser provinsen klarer å samle inn. Først skal serverscriptet legge inn normal befolkningsvekst og ressursøkning. Etterpå skal bygningsmodulen gjøre sine endringer. Den vil da legge til eventuelle ekstra ressurser som bygninger kan gi, og så øke ressurser og befolkning med

visse prosenter som en del bygninger skal gi. For og lettere kunne implementere andre moduler i serverscriptet skal alle moduler ha en funksjon – og bare en – som serverscriptet skal kalle når det kjøres. Denne funksjonen skal hete det samme i alle moduler, slik at alt serverscriptet må gjøre er å opprette en instans av modulen, for så å kalle oppdateringsfunksjonen.

Det er også viktig med samarbeid moduler i mellom, ettersom data fra en modul skal kunne ha innvirkning på data som blir beregnet i en annen modul. For å få til dette, bestemte vi oss for at moduler som har en slik innvirkning, må ha funksjonalitet for å beregne og hente ut innvirkningsgraden. Dermed kan modulen som trenger å vite en innvirkningsgrad bare kalle en funksjon i modulen som gir innvirkningen, og innvirkningsgraden blir hentet ut.

Dersom noe skal gjøres i en modul, skal modulen ta seg av dette selv. Det vil si at har man behov for å endre data i en annen modul, skal man be denne modulen gjøre dette istedenfor å forandre disse data selv. All data som har med en modul og gjøre skal også hentes ut igjennom modulen. Man skal ikke gå inn i databasen å hente ut dette selv. Som et eksempel på en modul som kan ha innvirkning på en annen, kan bygningsmodulen igjen brukes som et eksempel, men denne gang sammen med militærmodulen. Det skal være bygninger som gir innvirkning på hvor lang tid det tar å trene militære, bedre angrep og forsvar og hvor mye ressurser det må brukes for å få trent de militære. Når militærmodulen skal trene militære vil den da først finne original pris for enhetene for å så beregne ny pris. Dette gjøres ved å kalle funksjonen som returnerer innvirkningsgraden og modifisere original pris med det bygningsmodulen leverte ifra seg.

Andre moduler enn militærmodulen, som magimodul og tyverimodul skal også ha mulighet til å for eksempel drepe militære, det vil si endre data i militærmodulen. Det er da viktig at disse to modulene ber militærmodulen om å endre disse dataene, istedenfor å gå inn i databasen og gjøre dette selv.

I tyverimodulen er det ønskelig å kunne spionere på andre i spillet. Hvis man da under spionering skal finne ut hvor mange militære det er i en provins, så ber man militærmodulen om å hente ut disse dataene.

Alternativer:

Alternativer til objektorientering kunne vært å lage rett frem funksjonsorienterte script, som ville vært litt raskere enn objekter. Er man nøye når man programmerer vil objektorientert kode gi oss veldig oversiktelig kode og gode muligheter for senere utvikling (ved hjelp av arv). En ting med objekter er at hvis et objekt er tregt, kan man omstrukturer, omprogrammere og optimalisere koden i klassen uten å forandre i andre deler av systemet som bruker objekter av den klassen. Dette ville fort blitt mye vanskeligere å få til i et funksjonsorientert miljø. Med tanke på den tiden man sparer i

funksjonsorientert programmering, eller skal man si tiden man ikke sparer, syns vi ikke funksjonsorientering gir oss nok Med tanke på at dette er et stort prosjekt, hvor oversikt er viktig, falt valget på objektorientering.

I databasen har vi i modulene magi, tyveri, bygninger og utforsking valgt å ikke legge dataene om disse inn i databasen, men isteden bestemte vi oss for å lagre en referanse til hvor disse dataene lå. Denne referansen er navnet på en klassefil tilhørende enheten. I bygnings klassen vil dette være en klassefil som beskriver et bygningsobjekt for gitt bygningsenhet. I magimodulen vil klassefilen beskrive et trylleformel objekt (se vedlegg L - databasemodul). Ved at informasjonen om for eksempel de forskjellige bygningene blir lagt i objekter, trenger hver enkelt bygning bare å tenke på sine bonuser. Med hjelp av arv kan man nemlig plassere alle funksjoner i base klassen, og la alle underklasser arve dem. Dette fører til at man i bygninger der man vil implementere en bonus bare trenger å overstyre baseklassens funksjon. Ut ifra dette ser man at det også blir enkelt å legge til bonuser som ikke før har eksistert i klassen. Ettersom man kan implementere funksjonen som returnere bonusen i baseklassen, og la den returnere en standardverdi som sier at det ikke er noe bonus. Deretter overstyrer man bare denne funksjonen i klasser som skal ha denne bonusen. Det blir også enklere å legge til nye bygninger. Dette på grunn av at man slipper å flette funksjonaliteten inn i eksisterende funksjonalitet, man tar isteden og legger ny funksjonalitet inn i den nye klassen. Det å gjøre det på denne måten fører også til små og oversiktlige kildefiler.

Alternativet til dette ville ha vært og lagret egenskapene til bygningene og trylleformlene i databasen. For oss ser det ut som en komplisert måte å gjøre det på, ettersom forskjellige bygninger skal ha forskjellige egenskaper. Det samme er det å si om trylleformlene, alle skal ha forskjellige egenskaper. Dette ville ha skapt tabeller med mange felter, der mesteparten av feltene ville vært tomme i hver post. Dette fordi ingen av bygningene skal ha alle egenskapene. Skulle man i slik løsning implementere en ny bygning med nye egenskaper måtte man først ha endret metadataene i databasen slik at det ble plass til ny egenskap, for å så legge til denne nye bygningen. Etterpå måtte man ha flettet funksjonaliteten for denne nye bygningen inn i allerede eksisterende funksjonalitet. Det å stadig forandre metadata i en database, fordi man ønsker å legge til en ny enhet er heller ikke ønskelig. Slike forandringer kan ha uheldige virkninger på strukturen i databasen og er noe man ikke gjør så ofte. Ettersom vi vil ha mulighet til å stadig legge til nye bygninger og nye bonuser, finner vi det best å legge klassenavn i databasen og ikke egenskaper.

Vi kunne valgt å ikke dele opp i moduler, men lage et stort system som tok seg av alt. Med en sånn løsning ville feil forplantet seg gjennom hele systemet, slik at feil en plass ville ført til feil i hele systemet. Dette kan føre til store problemer med å isolere feil og enkelt kunne fikse dem. Skal man da isteden gjenopprette systemet fra siste sikkerhetskopi, vil hele systemet bli satt tilbake til samme versjon som sikkerhetskopien er på og mange arbeidstimer kan gå tapt. En annen ting med å ikke dele opp i moduler er at man gjerne må ha hele systemet oppe å gå for å få testet det. Å vente til man har fått opp et helt system for å kunne teste det kan skjøre seg total. Hvis man ikke har et system som kan testes oppe å gå før i de siste timer, og man finner ut at det ikke fungerer, har man svært få timer å få fikset dette på.

Hvis man deler et system inn i moduler, som har sine egne oppgaver å ta seg av, er det større sjanse for at resten av systemet fortsatt kan fungere hvis en av modulene feiler. Ved feil i en modul som alle andre moduler er avhengige av, vil selvsagt denne feilen få hele systemet til å krasje, men dette er ikke til å unngå. I et system oppdelt i moduler er det også letter å isolere en feil. Opprettingen av denne feilen trenger selvsagt ikke å være enkel, så mens man fikser feil kan man hente ut siste versjon av modulen fra sikkerhetskopi uten at det vil sette hele systemet tilbake til forrige versjon. Dermed taper man bare arbeidstimer i forhold til modulen det ble feil i. Et system som er bygd opp av moduler kan også være enklere i forhold til testing. Man bygger opp et grunnsystem med hovedmodulene og tester dette. Etterpå kan man utvikle modulene hver for seg og teste dem, for å siden putte dem inn i systemet og teste dem der. Dette fører til at man slipper å finne ut at man har utviklet et system som ikke fungerer - dagen før deadline.

Isteden for å bruke en databaseklasse kunne man ha brukt de innebygde mySQL funksjonene i PHP. En slik fremgangsmåte vil føre til store oppdateringer hvis PHP forandrer standarden på mySQL kallene. Det ville også ført til akkurat like mye arbeid hvis man blir nødt til å skifte database. Da må man gå igjennom alle scriptene som bruker databasen og forandre all bruk av databasen. Dette er en oppgave vi ikke har lyst til å utføre. På grunn av dette blir valget å bruke en database klasse som et lag over databasen, som igjen gjør produktet letter å flytte over til andre systemet.

Skal man bruke en klasse til å ta seg av kommunikasjonen med databasen er det fortsatt flere valg. Det finnes et bibliotek – PEAR – som er en utvidelse til PHP. Dette biblioteket støtter flere databaser og gjøre det enkelt å flytte produktet til andre systemer. Men det er kanskje ikke den beste løsningen i situasjonen vår. Veldig stor usikkerhet om hvor man kan få server, og om hvor medgjørlig system administrator vil være til å installere et system som PEAR kan få folk over på tanken om å kanskje lage noe selv. Det å installere et PEAR, og sette seg inn i det er en oppgave som vil ta en del tid. Ettersom vi ikke trenger all verdens funksjonalitet ifra databasen, men bare ting som å sende en spørring, hente ut resultatet, hente ut en og en linje fra resultatet og kanskje vite hvor mange linjer resultatet ble på. Blir valget å lage en egen klasse. På grunn av kravet til funksjonalitet vil dette ta kortere tid enn å sette seg inn i PEAR. Med vår egen klasse har vi også muligheten til å legge til funksjonalitet i et senere stadium hvis behovet skulle melde seg.

At hver modul skal hente ut sine egne data er en følge av den objektorienterte tankegangen, og at når man først har delt opp i moduler så er det ingen vits i å starte å bryte ned skillet mellom dem.

3.1. BESKRIVELSE AV MILITÆRMODUL

For mer detaljer for modulen, se vedlegg P.

3.1.1. OPPBYGGING

Implementeringen av dette spillet skulle forgå objektorientert i php. Derfor valgte vi å sette opp en Militærklasse for å ta seg av funksjoner som har med det militære å

gjøre. Script og klasser som inngår i Militærmodulen er *Military* klassen og scriptene *trainNewMil.php* og *trainMilitary.php*. *Det er trainNewMil.php* som skriver ut hvor masse militære man har, og *trainMilitary.php* sørger for å sende militære til trening.

3.1.2. KORT BESKRIVELSE

Det første modulen gjør er å lese inn informasjon om militære, hvor mange det er i alt, og hvor mange som er ute i krig. Deretter blir det skrevet ut for hver militærtype hvor mange det er totalt, hvor mange som ikke er ute i krig, antall dager det vil ta å trene militærtypen, kostnaden i gull og eventuelt metall. Det må også blitt gitt beskjed for hver militærtype i trening, hvor lenge det er til hver enkelt av enhetene blir trent ferdig. Bruker får også opp mulighet til å skrive inn hvor mange enheter som han vil trene. Da vil total kostnad for den militærtypen komme opp. Hvis brukere bestemmer seg for å trene militære blir militære lagt inn i treningstabellen i databasen og en bekreftelse blir skrevet ut

3.2. BESKRIVELSE AV KAMPMODUL

For mer detaljer for modulen, se vedlegg Q.

3.2.1. OPPBYGGING

Denne modulen er en utvidelse av *Military* modulen og blir dermed en utvidelse av *Military* klassen. For denne modulen er det laget et hjelpescript som heter *goToWar.php*. Det er denne modulen som tar seg av utkjemping av slag i dette spillet.

3.2.2. KORT BESKRIVELSE

I kampmodulen blir først antall militære hjemme av hver angrepstype skrevet ut, slik at bruker lettere skal vite hvor mange han kan sende til krig. Det blir også skrevet ut hvor lenge det er til de forskjellige militærtyper ute i krig kommer tilbake. Hvis det under krigen er blitt stjålet land, vil dette landet bli vist sammen med hvor lenge det er til militære kommer tilbake. Deretter kommer muligheten for å skrive inn hvor mange krigere man vil ha, og sende dem ut i krig. Hvis så vil angreps og forsvarspoeng bli regnet ut og eventuelle bonuser for bygninger og forskning vil bli lagt til. Deretter blir den med mest poeng vinner av krigen. Tap av soldater og eventuelt land blir regnet ut og det blir sendt beskjed til provins som ble angrepet om hvordan det gikk, og begge involverte kongedømmer får en beskjed.

3.3. BESKRIVELSE AV UTFORSKINGSMODUL

For mer detaljer for modulen, se vedlegg R.

3.3.1. OPPBYGGING

Utforskingsmodulen er en utvidelse av *Military* modulen og derfor også en utvidelse av *Military* klassen. Det er også lagt til et ekstra script for denne modulen – *exploreLand.php*. Det er dette scriptet som legger utforskingsmodulen inn i en ramme.

3.3.2. KORT BESKRIVELSE

Det første utforskingsmodulen gjør er å skrive ut hvor mange recruits/goblins/soldiers som er totalt, og hvor mange som er hjemme. Kostnad og totalt antall tikk for å få alt land som blir satt under utforsking blir også skrevet ut. Deretter kan et vist antall med militære postes til modulen som da regner ut riktig mengde land som blir utforsket og fordeler dette utover tidsaspektet det skal ta og utforske alt landet. Dette blir så lagt inn i riktig tabell i databasen, og en bekreftelse blir skrevet ut til bruker. Det blir også skrevet ut en oversikt over hvor lenge det er til land er ferdig utforsket.

3.4. BESKRIVELSE AV REGISTRERINGS MODUL

For mer detaljer for modulen, se vedlegg S.

3.4.1. KORT BESKRIVELSE

Modulen skriver ut en form hvor det er mulighet for å skrive inn informasjon om seg selv, og registrere seg i spillet. Når det trykkes på registreringsknappen vil all informasjon blir sjekket for eventuelle feil/mangler. Hvis det er plass til flere brukere vil brukeren bli registrert, hvis ikke skrives en feilmelding om dette. Brukerens kontoinformasjon blir sendt i e-post, og skrevet til skjerm i form av en webside

3.5. BESKRIVELSE AV MAGIMODUL

For mer detaljer for modulen, se vedlegg K.

3.5.1. OPPBYGGING

Magimodulen er i hovedsak bygd opp av to klasser: *Magic* og *SpellBase*. *Magic* er klassen som tar seg av all prosessering, brukergrensesnitt og grensesnitt til de andre modulene. *SpellBase* er en baseklasse som alle formelklassene, som for eksempel *EnchantedLand* og *EarthQuake* arver fra.

3.5.2. KORT BESKRIVELSE

Siden brukeren kommer inn på, magic.php sjekker at brukeren er pålogget og finner provins id'en. Et objekt av *Magic* klasen opprettes og denne genererer selve innholdet på siden.

Brukeren får lov til å angi kongedømme enten ved å skrive inn kongedømme nummer selv, eller ved å velge kongedømme fra en liste. Brukeren får også velge provins og formel fra lister. Brukeren kan så skrive inn antall magikere han vil bruke, og avhengig av typen formel han vil kaste, kan han skrive inn dager den skal vedlikeholdes eller velge formel fra en liste for å fjerne den.

Brukeren får opp hvor mye mana han har, en beregning på hvor mye ressurser og magikere som trengs for å kaste formelen, en generell beskrivelse av formelen og formler som er kastet på eller av ham selv. De formlene han selv har kastet på seg har han mulighet for å stoppe.

Når brukeren kaster en formel, sjekkes det at han har nok ressurser og magikere og alle tall som skrives inn blir sjekket så de virkelig er tall. Antall magikere sjekkes for

å være mellom 1 og det antall brukeren har. Antall dager sjekkes for å være mellom 1 og 24. Brukeren får en direkte tilbakemelding på hvordan det gikk, og avhengig av formel og resultat kan både kasteren og målet få en nyhet om resultatet i tillegg.

Modulen inneholder også funksjoner for *server-, bygnings-, forsknings-, tyveri- og militærmodulene* slik at disse kan hente ut effekten av formler.

3.6. BESKRIVELSE AV BYGNINGSMODUL

For mer detaljer for modulen, se vedlegg J.

3.6.1. OPPBYGGING

Bygningsmodulen er bygd opp av *Buildings, BaseBuilding* og de forskjellige *XxxBuilding* klassene i tillegg til *buildings.php, buildingDescription.php* og *Buildings.js.inc.php*. *Buildings* klassen tar seg av all prosessering av data mens *BaseBuilding* er en klasse som arves av alle de spesifikke bygningsklassene. *BuildingBase* klassen inneholder et slags grensesnitt med variabler, funksjoner og standard verdier som alle bygninger må ha.

3.6.2. KORT BESKRIVELSE

Siden brukeren kommer inn på, *buildings.php*, sjekker at brukeren er pålogget og oppretter et objekt av typen *Buildings*. Bygninger slettes hvis det er blitt gitt beskjed om det eller bygges om det er beskjeden. Så vises selve siden.

Brukeren kommer inn på en side der han har oversikt over bygningene han kan bygge med sin nåværende forskning. Han får for hver bygningstype se hvor mange han har bygd og hvor mange som er under bygging i tillegg til at han kan skrive inn det antall bygninger han vil bygge. JavaScript oppdaterer felter for total kostnad for bygningstypen og for alle bygningene han har skrevet inn ønske om å bygge. Han får også se hvor mye ressurser han har og hvor mye som blir igjen etter byggingen. Det sjekkes at han ikke får bygd mer enn han har ressurser til både ved hjelp av JavaScript og PHP i det han prøver å bygge.

Siden gir også mulighet for å skrive inn antall bygninger som skal ødelegges der brukeren må velge bygningstype fra en liste. Det sjekkes at han ikke får ødelagt mer enn de bygningene han har.

Nederst på siden vises en oversikt over alle bygninger han har under bygging og hvor mange tikk som er igjen før fe er ferdige.

Ved å klikke på navnet på en bygningstype, sender et JavaScript brukeren videre til *buildingDescription.php*. Denne siden viser bilde av bygningen, beskrivelse av den og alle dens egenskaper samt hvilken innvirkning dette har for brukeren.

Modulen inneholder også funksjoner for *server-, magi-, forsknings-, tyveri- og militærmodulene* slik at disse kan hente ut effekten av bygningene.

3.7. BESKRIVELSE AV MELDINGSMODUL

For mer detaljer for modulen, se vedlegg I.

3.7.1. OPPBYGGING

Meldingsmodulen er bygd opp av *Message* klassen, *Template* klassen og HTML filer. *Message* klassen tar seg av all prosessering og hvilke HTML sider som skal brukes ut i fra hvilken side brukeren vil vise. *Message* klassen benytter disse HTML filene som maler og bruker *Template* klassen som bindeledd mellom PHP koden og HTML koden.

3.7.2. KORT BESKRIVELSE

Siden brukeren kommer inn på, message.php, sjekker at brukeren er pålogget og oppretter et objekt av typen *Message*.

Brukeren kommer inn på siden for innkommende meldinger først. Her har han mulighet til å se på meldinger han har fått eller slette dem. Hvis brukeren går inn og leser en melding, vil han ha mulighet for å trykke på en ”svar” knapp som tar ham direkte til siden for å sende meldinger, og da vil riktig kongedømme og provins være valgt på denne. Alle meldinger signeres med brukernavn, provins og kongedømme. Når en melding er lest, blir den merket som sådan. Siden for sendte meldinger er identisk meldingene merkes ikke som lest dersom senderen leser den.

I menyen på toppen finnes det link til innkommende meldinger som også sier hvor mange innkommende meldinger som er ulest, send melding og sendte meldinger.

Send melding gir brukeren mulighet til å velge kongedømme fra en liste og deretter hvilken provins han vil sende til. Når meldingen sendes, sjekkes det for at brukeren ikke skriver inn HTML kode.

3.8. BESKRIVELSE AV SERVERMODUL

For mer detaljer for modulen, se vedlegg T.

3.8.1. OPPBYGGING

Server modulen er bygget opp av filene server.php og run (som er et shell script). Server modulen inkluderer også alle klassefilene i spillet og lager seg objekter av disse. Run er et script som går på serversiden i en crontab, som kjøres hver time. Scriptet sørger for at man står i riktig katalog, og starter så eksekusjonen av server.php.

3.8.2. KORT BESKRIVELSE

Server modulen er byggesteinen i spillet, siden det er denne modulen som driver alt fremover. Modulen flytter med andre ord alle ”brikkene” i spillet fram til neste runde. Modulen skal kjøres hvert tikk. Et tikk, som nå er definert som én time, bestemmes av hastigheten som crontaben kjøres med.

Server modulen starter med å opprette et databaseobjekt, for så å lese inn konfigurasjonen av spillet fra databasen. Om spillet er satt til å kjøre, begynner modulen å oppdatere ressursene til alle provinsene. Den sørger også for å fjerne

spillere som har blitt redusert til 0 land. Deretter laster den inn alle klassefilene og kjører doTick funksjonen på disse. Klassefilene lastes inn i rekkefølgen: Buildings, Military, Science, Magic, Thievery og News. Her sørger modulen også for å ta tida på hvor lang tid kjøringen av hver enkelt klasse tar.

Modulen inneholder feilsjekking for å kontrollere at alle spillverdiene er lovlige. Om ulovlige verdier oppdages, skrives det en feil i fila error.log. Etter at all nødvendig oppdatering er foretatt, oppdateres konfigurasjonen i databasen og statistikk for spillet skrives til fila game.log.

3.9. BESKRIVELSE AV DATABASEMODUL

For mer detaljer for modulen, se vedlegg L.

3.9.1. OPPBYGGING

Database modulen er bygget opp av filen Database.class.inc.php.

3.9.2. KORT BESKRIVELSE

Databasemodulen tilbyr et objekt orientert interface for å koble seg opp mot en database og utføre spørringer mot denne. Når modulen startes mottar den brukernavn, passord, host og database den skal koble seg opp mot. Etter at modulen har blitt tilkoblet databasen holder den selv orden på en link mellom seg selv og databasen, slik at andre moduler (som bruker databasen) slipper å bry seg om dette.

3.10. BESKRIVELSE AV PÅLOGGINGSMODUL

For mer detaljer for modulen, se vedlegg N.

3.10.1. OPPBYGGING

Påloggingsmodulen består av følgende filer. Login.php, logoff.php, User.class.inc.php og isLoggedOn.php.

3.10.2. KORT BESKRIVELSE

Påloggingsmodulen sørger for at spillere får tilgang til riktig provins og at ikke registrerte spillere ikke får tilgang til de underliggende modulene.
Den sørger også for å opprette database, provins og bruker objekter som de andre modulene skal bruke.
Påloggingsmodulen spør først etter brukernavn og passord. Etter å ha mottatt dette sjekkes det mot databasen. Om det er en feil kombinasjon føres brukeren tilbake til login siden med en feilmelding.
Om brukernavn og passord stemmer lages en cookie og en databasereferanse til denne opprettes i databsaen. I tilegg opprettes det et rammeverk for spillet. Menyen settes til venstre og spillområde blir da resten av skjermen. Menyen til venstre lastes fra sin egen fil, mens spillområde settes til å laste showProvince.php – et script som viser provinsen. Dette kaller igjen metoden isLoggedOn.php for å sjekke om provinsen er logget på.

Alle script som kalles fra menyen kaller isLoggedOn.php for å sjekke om brukeren har tilgang til siden. Om brukeren ikke har tilgang stoppes scriptet og en feilmelding vises.
Brukeren har kun tilgang til sidene dersom han er markert som å være 'Alive'. Om brukeren ønsker har han også mulighet til å logge av. Da slettes cookie og databasereferansen settes til å bli ugyldig i databasen.

3.11. BESKRIVELSE AV FORSKINGSMODUL

For mer detaljer for modulen, se vedlegg R.

3.11.1. OPPBYGGING

Forskingsmodulen er bygget opp av filene science.php, Science.class.inc.php, ScienceBase.class.inc.php og de arvede klassefilene som hentes ut i fra databasen.

3.11.2. KORT BESKRIVELSE

Forskingsmodulen innholder all forskningen i spillet. Alle provinser har sitt eget nivå i forskningen og for hver ny forskning man får tak i, får man nye eller bedre fordeler i spillet. Noen forskninger gir for eksempel tilgang til ny magi, mens andre gjør deg bedre i angrep.
Forskning har sin pris: Det koster både metall, gull og tid.
Forskingsmodulen starter med å laste inn all forskning fra databasen og bygger seg et science tre. Deretter laster den inn all forskningen den gitte provinsen har. Så viser den hvilke vitenskaper man allerede har til brukeren. Om man forsker frem noe nytt vises det hvor lang tid det er før denne blir ferdig, samt en beskrivelse av hva den gjør.
Om man ikke forsker frem noe vises alle tilgjenglige forskninger man har, samt en liten beskrivelse av hvilke egenskaper de gir og hvor lang tid det tar å forske dem frem.

3.12. BESKRIVELSE AV TYVERIMODUL

For mer detaljer for modulen, se vedlegg M.

3.12.1. OPPBYGGING

Tyveri modulen er oppbygget av filene thievery.php, Thievery.class.inc.php og ThieveryBase.class.inc.php. I databasen ligger det lagret hvilke tileggsfiler (tyverioperasjoner) som skal inkluderes. Hver tyverioperasjon er en egen fil som er arvet fra ThieveryBase.

3.12.2. KORT BESKRIVELSE

Det første Tyver modulen gjør er å lese inn alle operasjonene som finnes, inkludere klassene og lage seg et array med objekter av disse operasjonene. Så skriver den ut en liten tekst som helt basic forklarer hvordan tyvene fungerer. Deretter skriver den ut hvor mye "influence" som er igjen, hvilket kongedømme man har valgt, hvilken provins man skal utføre operasjonen på og hvilken operasjon man skal utføre. For å finne ut hvilken operasjon man kan utføre må

modulen finne ut hvor mye forskning provinsen har ved hjelp av et forskings objekt.

Man har mulighet til å utføre operasjoner på alle andre enn seg selv og spillere som fremdeles er i ”protection” – en magisk beskyttelse for helt nye spillere.

3.13. BESKRIVELSE AV NYHETSMODUL

For mer detaljer for modulen, se vedlegg U.

3.13.1. KORT BESKRIVELSE

Modulen får beskjed om å sende en nyhet til en bruker eller et kongedømme. Hvis bruker går inn på inn på sitt ’hjemmeområde’ vil han få listet opp alle nyheter som er sendt til provinsen, og som ikke er sett før. Det er mulighet for å få listet opp alle nyheter som er kommet inn som ikke er slettet enda. Går brukeren inn på nyhetsområdet blir nyheter for kongedømmet listet opp

3.14. BESKRIVELSE AV POLITIKKMODUL

For mer detaljer for modulen, se vedlegg W.

3.14.1. OPPBYGGING

Politikkmodulen er bygget opp av fila vote.php

3.14.2. KORT BESKRIVELSE

På denne siden kan brukerne stemme på en konge. Den so blir valgt konge får flere valg på denne siden. Han kan endre navnet på kongedømmet sitt og legge inn en url til banner.
Kongen vil også få andre fordeler i form av resurser, men det blir ikke fordelt i dette scriptet

3.15. BESKRIVELSE AV STATISTIKKMODUL

For mer detaljer for modulen, se vedlegg V.

3.15.1. KORT BESKRIVELSE

Disse sidene skal gi brukeren en del forskjellig informasjon om andre kongedømmer og provinser. Man får se alle provinser i kongedømmet sitt, de femti størsteprovinsene og de ti største kongedømmene. Alle sidene er linket opp mot provinceAction.php der man kan angripe, sende ut tyver eller bruke magi på den valgte provinsen

3.16. BESKRIVELSE AV GUIDEMODUL

For mer detaljer for modulen, se vedlegg X.

3.16.1. OPPBYGGING

Består av filene *guide.html, guide01.html, guide02.html, guide03.html, guide04.html, guide05.html, guide06.html*

3.16.2. KORT BESKRIVELSE

Guiden skal gi brukeren basis informasjon om de ulike elementene i spillet. De elementene som krever litt mer detaljert informasjon har egne sider. Guiden er også linket mot en tutorial for helt nye spillere. Denne er en slags ”walkthrough” for spillet, med skjermbilder og hjelp til hva man skal gjøre.

3.17. BESKRIVELSE AV ADMINMODUL

For mer detaljer for modulen, se vedlegg Y.

3.17.1. OPPBYGGING

Server modulen er bygget opp av filene Admin.class.inc.php, admin.functions.inc.php,
users.php, game.php, science.php, thievery.php, news.php, resetMil.php, login.php og logout.php

Adminmodulen finens på http://www.thurmann.net/chaos/admin/

3.17.2. KORT BESKRIVELSE

Admin modulen er administrasjonsdelen av spillet. Her kan man starte/stoppe spillet, slette brukere, få tilgang til forumet som superbruker, legge til nye ”news” og legge til nye klassefiler. Adminmodulen inneholder også inn og utloggingsegenskaper for å ivareta sikkerheten.

Adminmodulen starter med å spørre om brukernavn og passord. Dersom man blir autentifisert, kommer man inn i et menysystem.

1. Game Users
2. Game Edit
3. Game Sciences
4. Game Thievery
5. Forum
6. News
7. Reset Military
8. Logout

1. Game users

   Om man trykker på dette valget får man spørsmål om man vil se alle kontoer eller kontoer som ikke er merket med ’Alive’ flagget.
   Om man velger å se alle kontoer vises alle spillere sortert, etter når de sist var logget inn, slik at inaktive brukere vises øverst. Her kan man se litt kontoinformasjon i tilegg til at man har muligheten for å sperre kontoen ved å trykke på ”close” valget.
   Velger man å se kontoer som ikke er merket med ’Alive’ flagget vises disse også sortert etter når de sist var logget inn. Her har man to valg. Man kan enten gjenåpne kontoen eller man kan slette den totalt i fra systemet.

2. Game Edit

Om man trykker på dette valget får man opp en status på spillet, hvor mange aktive spillere som finnes, hvilket tikk man er i og om spillet kjører eller ikke.
Man har også muligheten til å stoppe og aktivere spillmotoren herifra.
Se også beskrivelse for game.php

3. Game Sciences
   Om man trykker på dette valget får man opp en oversikt over hvilken science klasser som allerede finnes i databasen og hva som kreves for å utforske de forskjellige sciencene.

   Man kan også laste inn nye klasser i databasen. For at klassen skal kunne lastes inn må den ligge i riktig katalog på serveren (www/scripts/sciences). Scriptet gjør en test på om filen eksisterer. Om fila finnes, legges filnavnet til i databasen og Science klassen prøver å laste den. Om det filnavnet man skrev inn ikke inneholder en klasse som er arvet fra ScienceBase, eller klassen har feil navn (ref: science mod for regler) vil Science klassen komme med feilmelding og bli ubrukelig til feilen blir rettet opp.

4. Game Thievery
   Ved å trykke på dette valget får man opp en oversikt over hvilke thievery klasser som allerede fnnes i databasen.

   Man kan også laste inn nye klasser i databasen. For at klassen skal kunne lastes inn må den ligge i riktig katalog på serveren (www/scripts/thievery). Scriptet gjør en test på om filen eksisterer. Om fila finnes, legges filnavnet til i databasen og Thievery klassen prøver å laste den. Om det filnavnet man skrev inn ikke inneholder en klasse som er arvet fra ThieveryBase, eller klassen har feil navn (ref: Thievery mod for regler) vil Thievery klassen komme med feilmelding og bli ubrukelig til feilen blir rettet opp.

5. Forum
   Her får man tilgang til forumet som administrator. Ref: forum

6. News
   Ved å trykke her får man opp en oversikt over alle nyhetene som er postet i spillet. Man har også mulighet til å legge til nye nyheter. Nyhetene vises på ”News” siden i spillet, i tilegg til at den nyeste nyheten, i opptil 2 dager, blir vist på førstesida i det man logger inn.

   Nyhetene kan inneholde html kode og javascript.

7. Reset Military
   Om man trykker på dette valget vil man få et spørsmål om man virkelig vil resette militæret. Svarer man JA vil alle militærtypene i spillet forkastes og nye vil lages. *Dette må på ingen omstendigheter gjøres under et aktivt spill!*
   Denne linken skal kun brukes om man skal legge til/forandre egenskaper til militæret.
8. Logout

Sletter cookien som sier at du er pålogget i tilegg til å sette et flag i databasen om at cookien ikke lenger er gyldig.

3.18. BESKRIVELSE AV FORUMMODUL

For mer detaljer for modulen, se vedlegg **Feil! Fant ikke referansekilden.**.

3.18.1. OPPBYGGING

Forummodulen er bygd opp av følgende filer: *Forum.class.inc.php*, *forum.php*, *forumUser.php* og filer med HTML kode. Her er alle HTML filene maler som brukes av *Forum.class.inc.php* for å vise forumet til brukeren. Siden forumet skulle lages ved bruk av maler, er *Template.class.inc.php* (se vedlegg **Feil! Fant ikke referansekilden.**-**Feil! Fant ikke referansekilden.**) filen som inneholder klassene som binder sammen HTML og PHP kode fra forskjellige filer. *forum.php* og *forumUser.php* er filene man faktisk kommer inn på når man kommer inn i forumet.

3.18.2. KORT BESKRIVELSE

En pålogget bruker eller en administrator kommer inn på *forumUser.php* siden. Denne siden sjekker om klienten er *administrator* eller en pålogget *bruker*. En klient som trykker på forum linken på hovedsiden, kommer inn på *forum.php* siden. Denne siden sjekker ikke for noen pålogging, men oppretter et *Database* objekt og klienten klassifiseres som *gjest*.

En *gjest* får opp oversikten over innlegg i verdensforumet og har muligheten til å se på disse eller skrive innlegg selv. Han må skrive inn et navn, en innleggsbeskrivelse på maksimum 180 tegn og selve teksten til innlegget. Alle innlegg vil bli signert med navn og ”guest in forum”.

En *bruker* kan være enten en *konge* eller vanlig *bruker*. Begge disse vil komme inn på siden som viser forumet for sitt eget kongedømme. Denne siden viser banneret til kongedømmet hvis de har noe. Begge kan lese og legge inn nye innlegg på samme måte som *gjesten*, men navnet vil bli gitt automatisk ut fra brukernavnet. Alle innlegg vil bli signert med ”<brukernavn> <provins> in <kongedømme>”. Det er mulighet for å gå til verdensforumet også og skrive innlegg i dette slik at signaturen blir som i kongedømmeforumet. En *konge* har i tillegg muligheten til å slette innlegg i kongedømmeforumet.

En *administrator* har mulighet for å velge kongedømmeforum fra en liste over kongedømmer og ha de samme mulighetene som en *konge* i disse. *Administratoren* har i tillegg mulighet for å slette innlegg fra verdensforumet. Alle innlegg fra *administratorer* blir signert med ”ADMIN administrator in the game”.

Forumet benytter seg av maler, det vil si at HTML kode og PHP kode er helt avskilt fra hverandre. Det er laget en *Template* klasse som tar seg av koblingen mellom PHP og HTML.

4. IMPLEMENTERING / KODING / PRODUKSJON

4.1. PLANLEGGING, OPPFØLGING OG RAPPORTERING

4.1.1. HOVEDINNDELING AV PROSJEKT

Prosjektet skal deles inn i forskjellige moduler, det vil si, deler som skal kommunisere med hverandre. Disse omfatter blant annet militære, magi og bygninger (se vedlegg Ø – gant-skjema) Ferdigstilling av disse modulene følger fremdriftsplanen.

4.1.2. KRAV TIL STATUSMØTER OG BESLUTNINGSPUNKT

Gruppen skal ha møter en gang i uken for å diskuterer problemer, valg nye og løsninger. Det skal føres et møtereferat fra hvert av disse møtene. Disse skal følge en mal og legges i egen katalog på server. Gruppen blir enig om hvem som skriver referatet på møtet. Statusmøter skal holdes ved hver milepæl.

4.2. VERKTØY

Vi har stort sett programmert i Macromedia Dreamweaver, EditPlus og pico. Pico er blitt brukt ved små endringer i script som allerede lå på serveren ettersom man da slipper å laste ned og opp filene hele tiden. EditPlus og Dreamweaver er blitt brukt for selve hovedutviklingen da disse har bra fargemarkering av kode og de har ikke irriterende automatisk fullførelse av koden.

4.3. KODING

Kodestandarden vi alle har prøvd å følge ligger i vedlegg C, noe som stort sett har fungert veldig bra. Et eksempel finnes i vedlegg F. Vi har prøvd å fordele oppgavene så godt som mulig slik at alle har hatt noe å gjøre hele tiden, og alle har fått omtrent like stor arbeidsmengde. Alle skulle også skrive en liten tekst om sitt arbeid hver dag man hadde gjort noe. Dette ble fulgt opp nøye i starten, men den siste måneden ble det litt glemt. Malen for disse arbeidsloggene er i vedlegg E, og eksempler fra loggen vår finnes i vedlegg **Feil! Fant ikke referansekilden.**. Hele loggen finnes i katalogen *worklog* på vedlagt CD.

4.4. MØTER

Vi har hatt møter hver uke hvor vi har diskutert, planlagt nye moduler og gått gjennom feil vi måtte rette opp. Fra de fleste av disse møtene har vi skrevet møterapporter etter malen i vedlegg H. I vedlegg **Feil! Fant ikke referansekilden.** finnes det eksempler på møtereferater og i katalogen *Documents* på vedlagt CD finnes alle møtereferatene. På disse møtene begynte vi etter hvert også å sitte og programmere sammen en stund for å klare å samkjøre grensesnittet mellom modulene. I starten hadde vi også møte med veileder hver uke, noe som etter en stund falt bort. I etterkant ser vi at det muligens hadde vært lurt å fortsette disse ukentlige møtene selv om vi kanskje ikke hadde hatt så veldig mye å si hver gang.

Bortsett fra disse møtene har vi delt på modulene og stort sett programmert for oss selv med mobiltelefon og meldingssystemer som IRC og ICQ som kommunikasjonsmiddel.

Dette har fungert bra, men de ukentlige møtene var likevel veldig viktige for å få en god sammenheng mellom alle moduler.

4.5. VALG

Under arbeidet har vi kommet over en del problemer og beslutningspunkt hvor vi har valgt ut i fra vår viten på det aktuelle tidspunktet. Noen av disse har vist seg å være veldig gode valg, mens andre har vist seg å være mindre bra.

4.5.1. OBJEKTORIENTERT?

Vi kunne velge å lage rett frem funksjonsorienterte script som vil være litt raskere, eller å bruke en litt tregere OOP tankegang med klasser. OOP vil gi oss veldig oversiktlig kode og gode muligheter for senere utvikling ved hjelp av arv. Maskinene er i dag så raske at de få millisekundene vi sparer på funksjonsorientert kode ikke er verdt det. Valget falt raskt på OOP, da dette er et stort prosjekt der vi trenger oversikt.

4.5.2. PROGRAMMERINGSSPRÅK?

Vi hadde flere alternativer når det kom til valg av programmeringsspråk.

Perl: Perl er et script språk som oppstod rundt 1987, med andre ord har det vært med internett helt siden starten av. Perl har, i dag, klasser og databasebiblioteker som lar oss programmere objekt orientert og koble oss opp mot database. Desverre tilbyr ikke skolen et godt kurs i perl, som web programmering, så om vi skulle valgt dette språket måtte vi funnet ut ganske mye på egen hånd. Dessuten har perl en syntaks som er ganske så ulik det vi er vant med fra blant annet c++ og java.

c/c++:
CGI kan også programmeres i fullverdige programmeringsspråk, slik som c/c++. På nettet finnes det biblioteker som både gir tilgang til databasebiblioteker og POST/GET metoder. Siden c++ blir kompilert er dette uten tvil det raskeste språket vi kan velge. Men.. Siden databasen vår skal stå å gjøre mesteparten av jobben er ikke hastigheten på scriptene våre så farlig. Å kode en web applikasjon i c++ byr på store utfordringer og muligheter til å gjøre veldig store feil. Det er veldig mye kode som skal til for å gjøre veldig lite. I tilegg må vi finne biblioteker for å mota forms og biblioteker for databasetilgang, og hvordan dette igjen skal fungere med web serveren er vi ikke sikre på.
Vi valgte også å ikke bruke c++ på server siden selv om c++ er lynraskt. C++ har mange fordeler når vi tenker på klassestruktur og lignende, men serveren og web scriptene bør kodes i samme språk, da klassene både skal inkluderes i serveren og i web skriptene.

ASP:
Asp er et basic lignende som så dagens lys rundt 1996. ASP, har som basic, en veldig enkel syntaks og er et forholdsvis enkelt språk. ASP har også databasebiblioteker, som lar oss lage de sidene vi trenger. Et problem med ASP er at det ikke er så oversiktlig å lage større web applikasjoner med det. Asp støtter ikke objekt orientert programmering, som er noe vi hadde bestemt oss for å programmere

under. Vi har .NET som støtter ASP, men denne pakka er dyr og er ikke noe fattige studenter har råd til. ASP har også vist seg å være tregt i følge diverse tester. Grunnen er at asp benytter seg av com objekter som skal oppstå og forsvinne i hver operasjon. (http://php.weblogs.com/php_asp_7_reasons)

Java servlets:
Java servlets er java sitt svar på "server side" kode. Java servlets har tilgang på hele java familien sitt bibliotek og fungerer tilegg som et plattform uavhengig språk. Fordelen med servlets er at koden vil bli kompilert, så vi får en raskere kjøring enn det scriptspråkene tilbyr. Kunnskapsnivået i gruppa på servlets er labert, og om vi velger dette må vi bruke lang tid på å sette oss inn i det nye utviklingsmiljøet. Dessuten har vi ikke avklart hvilken server vi skal bruke og om denne støtter java.

PHP:
PHP er et forholdsvis nytt språk som ble utviklet rundt 1997. PHP har en blandingssyntaks mellom perl og c, støtter OOP og har databasebiblioteker. PHP har en fantastisk integrasjon med MySQL og tilbyr en rekke måter å kommunisere med en MySQL database på. PHP er også et veldig raskt språk og oppfyller dermed våre krav til hastighet. Gruppeleder har i tilegg programmert en del større prosjekter i PHP mot MySQL, samt at alle gruppemedlemmene er kjente med syntaksen. Dette gjør at vi kan komme i gang tidlig med programmering, i tilegg til at det blir lettere å strukturere koden. PHP er også plattform uavhengig og i og med at vi ikke har funnet oss noen server enda, er det jo greit å kunne være litt fleksibel.

4.5.3. DATABASE?

Interbase: Interbase er en gratis databaseløsning fra borland. Interbase har vi brukt i tidligere fag og er kjent med både sterke og svake sider fra denne. Interbase har triggere og muligheten for å lage egne funksjoner i databasen og det er jo helt klart et pluss. I tilegg kan vi legge til regler for hva som skal være i databasen og ikke, noe som gjør at vi opprettholder datbaseintegriteten.

MySQL: MySQL er databaseløsninga vi bruker i faget Klient og server. Den mangler triggere (slik som interbase har) men på den andre siden er ikke triggere noe vi kommer til å absolutt trenge i prosjektet. Triggere hadde vært kjekt, men er ikke så veldig vanskelig å komme rundt. MySQL er også kjent for å være rask og stabil og det er helt klart et pluss – da databasen kommer til å stå for det meste av jobbingen. Alle gruppemedlemmene er kjent med MySQL fra tidligere prosjekter og har mange positive erfaringer. Eneste som er dumt med å velge MySQL er at verktøyet vi bruker for å designe databasen ikke er helt kompatibelt med MySQL. Men det skal ikke by på store problemer å forandre SQL scriptene slik at de fungerer korrekt. Valget falt da til slutt på MySQL fordi den har fungert tilfredstillende i tidligere prosjekter og har all funksjonalitet vi trenger.

4.5.4. SPILLKONSEPT

Hele gruppa var fast bestemt på at vi skulle lage et fantasy spill, altså et spill i middelalderen med alver, troll og magi. Vi ville først basere spillet på bokserien "The Wheel of time", men her var det noen copyright ting som måtte ordnes opp i først. Så i starten laget vi et generellt spillkonsept først mens vi ventet på svar.

Ledgend entertainment hadde tidligere laget et FPS (first person shoot them up) spill basert på bokserien, så vi tok kontakt med prosjektlederen for det prosjektet (se vedlegg AA - Brev). Etter mailkorrespondering fant vi ut at det kunne ta ganske lang tid før vi fikk nødvendig tillatelse, og at vi kanskje ikke fikk det i det hele tatt! Derfor baserer vi ikke spillet direkte på bokserien, men heller på standard fantasy litteratur.

4.5.5. BETATESTING

Under et av våre tidligste møter, fant vi ut at vi skulle sette av god tid til betatesting av spillet. Det er tross alt et spill for mange tusen spillere vi skal lage og vi er nødt for å finne ut om serveren tåler påtrykk. Dessuten kan mange spillere tilby gode tilbakemeldinger på design og spill (u)balanse, i tilegg til at mange småfeil som vi ellers ikke vil finne i et ganske lukket miljø, blir avdekket. Bakdelen er at spillet bør være ferdig i det betatesting finner sted, så det blir mer press på oss om vi setter av tid til dette. Det er full enighet i gruppa om at produktet bør testes før det settes ut på markedet og at gjennomførelsen av dette kan være en nyttig erfaring å ha med seg. Målet er derfor å bli ferdig med spillet til slutten av april og ha testing oppe helt til innlevering.

Alternativt kunne vi unnlatt å teste spillet og brukt mer tid på å kode og luke ut feil for oss selv. Men fire stykker kan ikke teste et spill for flere tusen spillere, i tilegg til at spillbalansen ikke får noen ordentlig test på seg før man får en spillmasse på minst 50 spillere. Først da får man en indikasjon på hvordan serveren vil takle spillmassen.

4.5.6. SERVER

Da vi startet opp prosjektet var det veldig usikkert hvor vi skulle ha serveren. Veileder sa han kunne skaffe til veie en maskin og i tilegg hadde noen av gruppemedlemmene ekstrautstyr å bidra med. Etter å ha vært i kontakt med IT tjenesten på skolen for å få tillatelse til å ha maskinene stående på skolen, oppstod det et nytt problem. IT tjenesten ville ikke uten videre la oss kjøre en web server som folk kunne ha tilgang til utenfor skolen. Med andre ord kunne bare studenter på skolen få tilgang til serveren (spillet). IT tjenesten foreslo å sende en skriftlig søknad, men antydet at vi burde se etter andre løsninger om vi ikke kunne godta tilbudet. Anders tok da kontakt med en kamerat, Erlend Ringstad, som har web serveren thurmann.net. Han kunne godt ”hoste” prosjektet vårt som en vennetjeneste. Eneste bakdelen var at MySQL serveren inneholdt en elder versjon (MySQL 3.x) enn det vi hadde basert oss på (MySQL 4.x), og at serveren allerede sto og kjørte en del tunge sider fra før av. Vi ble lovet at MySQL skal oppdateres, men var nødt til å bruke den gamle versjonen så lenge. Vi skjønte etter hvert at oppgraderingen kom til å ta litt lengre tid og bestemte oss for å gjøre om på noe av koden for å få det til å passe med den eldre MySQL versjonen. Dette innebærte å traversere hele databasen ved hjelp av PHP opptil flere ganger for å oppnå ønsket effekt, noe som reduserer serverens oppdateringshastighet. MySQL ble oppdatert 20.04.03, et par dager før betatestingen startet. Vi bestemte oss for ikke å forandre koden i en så kritisk fase, men heller vente til etter betatesting før vi gjør noe med koden som fungerer. Alternativt kunne vi utsatt betatestingen på ubestemt tid for å re optimalisere koden, men det er liten vits når test-massen er såpass liten.

Estimert oppdateringstid for serveren er nå rundt 4 minutter en gang i timen, for ti tusen brukere. Under denne tiden vil fremdeles spillet være tilgjenglig, men brukerne kan oppleve LAG (heng i serveren) før de får lastet sidene. En oppgradering av MySQL kallene vil redusere total tiden til rundt 2 minutter på nåværende server. En oppgradering av maskinvare og flytting til en dedikert server, vil kunne radikalt redusere server tiden ytterligere.

5. TESTING OG KVALITETSSKIRING

Alle moduler ble underveis testet av utviklere for å få fjernet de groveste feilene. Da prosjektet var kommet så lang at det bare var magimodul som totalt manglet, og tyveri og forskning hadde kommet i sine første utgaver, ble spillet sluppet for betatesting.

For at vi skulle få tilbakemelding på feil som oppstod, brukte vi forumet aktivt. Det ble opprettet egne tråder i forumet for feil og forslag til forbedringer. Spillerne rapporterte feilen i den respektive tråden, og så fort vi fikk se innlegget, begynte vi feilretting. Mange av spillerne var veldig aktive og flinke til å rapportere feil og komme med forslag. Dette førte til at eventuelle feil fort ble oppdaget og fikset. Flere av spillerne var også i kontakt med via andre medier som IRC, ICQ eller personlig

Første betatest runde forgikk i 4 dager, men på grunn av litt uheldig manuell oppdatering av database ble militæret nullstilt. I løpet av disse fire dagene ble det også oppdaget noen alvorlige feil. Personer kunne logge seg på uten å skrive inn brukernavn og passord hvis de hadde vært logget på fra før og hadde glemt å logge seg av. Ved angrep ville det lønne seg å bare sende en og en soldat i krig, fordi man klarte å ta livet av mange av motspillerens soldater. Hvis man angrep en provins og vant krigen, skulle en provins miste en viss sum med land. Eventuelle bygninger på det landet som ble stjålet skulle også mistes, men isteden mistet provinsen nesten alle bygningene sine. Folk fikk også negative ressurser, noen bygninger forsvant fra alle oversikter og ble lagt til i provinsen hvert eneste tikk, noe som førte til at man kunne ha flere bygninger enn man hadde land til. Det ble også oppdaget tilfeller av at man hadde et negativt antall militære.

Vi gikk igjennom koden hvor feil oppstod og kodet om slik at de feilene ikke skulle skje igjen. På grunn av disse feilene ble det også bestemt at hele spillet måtte resettes slik at alle rader i databasen som innehold feil ble fjernet. Fiksing av feil og resetting av spill gikk nesten knirkefritt, men det ble oppdaget noen feil der script gikk ut ifra at første ID i databasetabell var 1. Dette førte til at registrering av militære ikke gikk som det skulle, men denne feilen ble raskt fjernet ved å organisere dataene i militærobjektet litt annerledes i det man leste dem ut fra databasen (se vedlegg P - militær).

I forbindelse med resetting ble det også implementert begrensning av antall angrep per tikk. Dette fordi man kunne angripe innaktive provinser til man ikke hadde mer militære igjen hjemme. Dette førte igjen til at enkelte vokste for fort, og ikke tapte militære. For å begrense antall angrep ble det innført moral (se vedlegg Q - kamp). Det ble klart at feilen med at alle bygninger ble ødelagt under et angrep fortsatt eksisterte. Minus ressurser og minusmilitære var også fortsatt til stede.

Forskning og tyveri modulen kom også i ny utgave og magi modulen ble presentert for betatesterne. Minus i militære var da delvis på grunn av feil i både militær og magi modulen – det var ingen kontroll på om enhetene som ble drept i magiske formler kastet på provins faktisk var i provinsen. Slik kunne man da ha negativt antall militære hjemme. Fikk slettefunksjonene i militærmodulen til å sjekke at det ikke ble slette mer enn det var i provinsen.

Negative ressurser og negative militære er blitt lokalisert til å være et samtidighetsproblem i forbindelse med databasen og det er mulighet for å unngå feilen ved å skrive om scriptene til å alltid låse tabeller og lese ut ressursene og sjekke om det er nok for å så skrive tilbake og låse opp tabellen.

6. DISKUSJON AV RESULTAT:

- Grafisk - design er ikke utfyllende nok. Det mangler en del bilder på de forskjellige sidene. Det er for mye luft på sidene, tomrommet bør fylles opp av grafikk som beskriver den enkelte siden. Prioriteten har ligger på å få spillet til å fungere ordentlig, og da har det visuelle kommet i andre rekke. Konkrete eksempler følger:
  - Ressurs oversikten i toppen burde ha fem like bilder. Nå har networth og gold forskjellig "stil" fra metal, food og peasants.
  - Bygnings siden. All informasjonen er midtstilt, blir for mye luft på sidene. Dette bør fylles opp med bilder
  - Military. Mangler bilder her og. Burde ha noe av militærenheter som trener. Bør har bilde av hver enhet, noe lignende det som finnes i bygningsklassen, der man kan klikke på en link og få opp et nytt vindu med bilde og detaljert informasjon.
  - Council. Denne siden burde sett annerledes ut. Det burde for eksempel vært av forskjellige ministere (forskning, militære, magi) og representative tall innenfor disse feltene. Basert på noen forholdstall, burde de kommet med forslag til hva som bør gjøres. Med en slik løsning kunne det også vært linker til de samme sidene.
  - Politics. Bilde som blir vist til de som ikke er konge.
  - Messages. Burde ressursbaren vært med? Eller noe annen relevant informasjon? Siden har for mye luft, burde hatt mer grafikk.
  - Attack. Bilder av militære som kriger. Både her og på trening burde bildene være relatert til rasen. Alle linke skulle vært linker her også.
  - Tabeller i buildings, explore, military, attack, skulle de hatt annet utseende? Nå er de veldig oversiktlige, men kanskje litt kjedelige

- I registreringsmodulen skulle det bli sendt en mail til bruker med kontoinformasjon. Slik løsningen er nå blir det både sendt mail og vist en webside med kontoinformasjon like etter at bruker er registrert. Man vil at kontoinformasjon skal sendes i e-post fordi det er en mulighet til å vite at brukeren faktisk er den han utgir seg for. Brukeren får heller ikke mulighet til å velge passord under registrering, dette for at man skal være litt sikrere på at det er bare han som ser passordet som skal brukes, og at email adressen er korrekt. Dessverre blir mail serveren vi bruker plukket opp av en del spam-filtre, noe som fører til at vi ikke får sendt mail til diverse e-post adresser. Spillet foregår for tiden under nokså kontrollerte omgivelser med 74 spillere i skrivende stund. Derfor har vi valgt å gi kontoinformasjon på webside like etter registrering slik at også de som har e-post adresse på en plass som velger ut vår server som spam-server skal få mulighet til å spille.
- Slag blir ikke utkjempet realistisk. Sånn som det er nå sender en person sine militære i angrep og slaget blir utkjempet med en gang og hæren blir borte i 7 ticks. Det kunne vært gjort mer realistisk ved at det tar en viss tid for å dra ut å kjempe, det faktiske slaget tar mer enn et tick, og hæren bruker en viss tid på å komme hjem. En sånn løsning vil åpne for at andre kan sende forsterkninger som hjelper til i forsvar. Vil også åpne for at andre kan angripe provinsen som startet krigen, slik at han blir nødt til å trekke styrkene sine tilbake. Dette fører til større grad av samarbeid provinser og

kongedømmer i mellom. Grunnen til at det ble valgt å utkjempe en krig med en gang, og la det ta en hvis tid før soldatene kommer tilbake er at der er det mindre å holde styr på og faktisk er en overkommelig oppgave.

- Verden er ikke representert på noe grafisk vis. Dette er fordi spillet faktisk er en forenkling av virkeligheten. Det kunne selvsagt vært ønskelig med en grafisk representasjon, men ettersom vi vil at alle skal ha mulighet til å angripe alle blir dette litt vanskelig. Skulle man lagd et skikklig kart med forskjellige sektorer som ville tilsvart kongedømmene. Ved en sånn representasjon kan man angripe alle andre provinser i kongedømmet, og provinsene i nabokongedømmer. Dette vil begrense antall provinser man kan angripe betraktelig, så lenge ikke kartet er et kakediagram hvor hvert kakestykke er et kongedømme. Dette ville igjen lagt grunnlaget for en veldig liten plass å utkjempe alle slagene på – midt på kaken. Man kunne selvsagt valgt å implementere støtte for allianser, slik at man kan reise gjennom kongedømmer på veien mot den man vil starte en krig med. En av tingene man skal passe på er å ikke ta seg vann over hodet. Et slikt allianse system kan bli veldig komplisert å holde styr på og var ikke aktuelt i dette spillet. Da står vi tilbake med kakediagrammet, hvor alle grenser mot alle i denne fiktive verden.
- Militærmodulen og angrepsmodulen burde vært kodet annerledes. Slik at det ble lettere å legge til/fjerne militærenheter. Også gjort det lettere å få mer realistiske slag.
- Det kan skje flere ting fra servesiden. Det kan for eksempel forekomme naturkatastrofer, pester, gode vekstforhold, ting som påvirker provinsene. Dette kunne for eksempel ramme ett og ett konedømme/provins, eller omfattende kan ramme hele verden.
- Forumet kunne vært litt mer avansert. For eksempel innlogging i verdensforumet når du ikke er inne i spillet. Mer avanserte funksjoner som sitering, svar-knapp og html-kode.
- Det kunne vært mer forskjell på rasene. De burde vært mer unike. Men det er vanskelig å finne en balanse slik at de stiller likt. Det blir veldig ujevnt hvis en av rasene er mye mektigere enn de andre. Da vil jo alle velge den rasen! Vi kunne også hatt med flere raser (dverger), men planen var å ha tre raser.
- Det har vært moro å se at spillet har tatt form. Spesielt hvordan de forskjellige modulene kommuniserer med hverandre. Det har vært få problemer når det gjelder å samkjøre alle modulene. Vi hadde jo som mål å få til et spillbart spill innen fristen. Da på bekosting av en del funksjoner som finnes nå. (hvilke?) Men med mye jobbing i løpet av betatestingen har vi fått fikset masse feil og lagt til ønskede funksjoner og laget et fullt spillbart spill! De fleste av modulene er også kodet slik at det er lett og legge til nye elementer i spillet uten å foreta store endringer.
- Det som har vært virkelig moro er betatestingen. Å få tilbakemelding fra andre og høre deres meninger. Spesielt da den positive kritikken, der vi får høre at det spillet er bra og folk synes det er kult. Det øker arbeidsmoralen hos oss, og vi får jo inspirasjon til å gjøre det enda bedre. Av de betatesterne som har vært med så ser vi jo at veldig mange av dem er veldig aktive. De logger seg inn flere ganger om dagen, enkelte nesten hver time! Når man har så ivrige spillere, blir jobben mye lettere for oss.
- Vi har også fått laget både guide og tutorial til spillet. Dette var egentlig ikke prioritert, men ettersom mengden av betatestere økte så ble det også et behov! I starten ble det jo stilt mange spørsmål i forumet om hvordan ting fungerte og hva ting gjorde.

Ved å legge ut guiden hadde spillerne et sted å hente den informasjonen de trengte, istedenfor å spørre oss i forumet! Tutorialen ble lagt til senere da flere spillere ikke hadde prøvd slike spill før. En ting som kan gjøres bedre med guiden, er at den kunne vært dynamisk og hentet ut informasjon fra databasen. Nå må den oppdateres manuelt ettersom nye elementer blir lagt til i spillet.

- Når det gjelder den visuelle opplevelsen, har vi merket at dersom man bruker skjermer av dårligere kvalitet, eller gamle skjermer, vil man ikke klare å skille alle fargene. Men på våres skjermer hjemme og skolens skjermer, ser fargene like ut.

7. KONKLUSJON

Vi har i dette prosjektet sett på målene vi satt oss i starten og prøvd å oppfylle disse etter beste evne. Gant skjemaet ble ikke fulgt slavisk da noen moduler tok lengre tid enn planlagt. Vi oppdaget også under utviklinga at noen moduler måtte deles opp eller at vi måtte legge til helt nye. (Ref. vedlegg AA)

Provins klassen mangler doTick funksjon og det provinsklassen burde tatt seg av på serversida er heller hard kodet inn i serverscriptet. Klassen burde helt klart hatt denne funksjonen da Provins klassen bør står for alt som har med provinsen å gjøre. Dette skjedde fordi Provins klassen var en av de første klassene vi laget og vi hadde ikke laget malen for hvordan tikk skulle løses på serversida.

Da vi starta prosjektet bestemte vi oss for å følge OOP. Vi har ikke kommet helt i mål på dette fordi raser ikke er kodet inn som objekter. Dette er en ganske stor bom fordi implementering av nye raser blir vansklig, spesielt om de skal ha forskjellige egenskaper. Grunnen til at vi bomma her er fordi vi tenkte mest på rasene som navn, og ikke at de skal føre med seg unike egenskaper. Vi burde hatt en baseklasse Raser og arvet alle underraser fra denne.

Prinsippet nevnt ovenfor fant vi først da vi uviklet bygningsmodulen. Dette ble grunnsten i videre tankegang i utviklingen og gjorde videre arbeid mye enklere. Dessverre var militærmodulen laget så denne løsningen ble ikke implementert her. Utforsk og kamp modulen bygger direkte på militærmodulen så disse kunne heller ikke følge samme prinsipp.

Hadde vi sittet mer sammen i starten kunne vi sansynligvis fått implementert gode løsninger tidligere (se ovenfor). Vi startet ikke med programmering i gruppe før det hadde gått et par måneder, men når vi ser tilbake burde vi gjort dette helt fra starten av. Dette er fordi kommunikasjonen ble mye bedre og vi fikk mye hjelp og innspill av hverandre under tida vi satt sammen.

Vi utsatte betatestinga en rekke ganger fordi vi ville tilby et produkt med mer funksjonalitet. Når vi først fikk opp betaen fikk vi mange gode tilbakemeldinger veldig fort. Hadde vi klart å få opp betaen før, kunne vi begynt feilretting og spillbalansering på et tidligere stadie. Vi hadde fått feilsøkt hver modul grundigere og kunne fått en indikasjon på hvor ”skoen” trykket. Istede fikk vi veldig mange moduler å feilsøke på samme tid. Grunnen til at betatestinga ble utsatt var at fordi vi manglet forum og uten forum hadde vi ikke fått tilbakemeldinger. Tilbakemeldinger per epost var aldri et alternativ.

Den løsninga vi har kommet frem til i løpet av den tida vi har hatt til rådighet lever opp til de målene vi har satt oss. Det mangler fremdeles noen bygninger, tyveri oprasjoner og magi elementer, men dette er utvidelser – spillet er fullt spillbart slik det er nå. Og ingen vital funksjonalitet mangler.

Vi har ikke gått ut aktivt for å promotere spillet, tiltross for dette har vi nå over hundre spillere. I starten av testingen hadde vi informert femten personer, og etter dette har jungeltelegrafen gjort sitt. Det at folk bruker synes spillet er bra, og bruker mye tid på å spille det, er jo bevis på at spillet har slått an.