Indholdsfortegnelse

Indledning

Denne opgave er den skriftlige del af vores bachelorprojekt. Den beskriver designprocessen og de valg vi foretog i forbindelse med skabelsen af den anden halvdel af bachelorprojektet – vores multimedieproduktion.

Vi valgte at lave et computerspil som multimedieproduktion, fordi vi ikke tidligere har haft mulighed for det på vores uddannelse, og det er det vi begge gerne vil arbejde med fremover.

For at opnå mest muligt bekendtskab med spilmediet, har vi arbejdet særligt grundigt med at skabe et produkt på et solidt teoretisk grundlag, således at vi har kunnet udforske de forskellige muligheder, mediet giver os.

Operationalisering

- Vi bruger UML som notation til at vise systemstrukturen.
- Vi citerer med APA fordi det er kompatibelt med online kilder.
- Billederne af maskiner samt titelskærm i spillet er lavet af en ekstern grafiker, efter vores anvisninger.
- Opgaven fylder 91426 tegn, Søren er ansvarlig for side 1-16 og Nicki er ansvarlig for side 17-33 undtagen konklusion og indledning, som er fælles.

Strukturen i opgaven er som følger: Redegørelsen vil følge et mønster, hvor vi først klargør de grundlæggende teorier om opgavens genstandsfelt, dernæst den arbejdsmetode, vi fulgte i

artefaktproduktionen, og endelig teorierne bag de temaer, vi inkorporerede i artefaktet. Vi vil således først redegøre for Zimmermans teori om narrativitet, interaktivitet, leg og spil, og sammenligne hans teori om narrativitet med Neitzels begreber Fabula og Syuzhet, samt Qvortrups kunstbegreb og disses relation til computerspilsmediet. Dernæst vil vi redegøre for Juuls videreførelse af Zimmermans spilbegreb.

Herefter vil vi redegøre for den første del af den arbejdsproces, der anbefalet for computerspiludvikling af Rollings & Morris. Så følger en kort redegørelse for principperne bag objektorienteret programmering (OOP), samt de programmeringsmønstre (”Patterns”) vi fulgte.

Ydermere vil vi redegøre for de to teorier, vi brugte som lagde grundlaget for spillets konkrete udformning, Porters værdikæde samt Poppers kritiske rationalisme. Afsnittet om kritisk rationalisme vil have form af en introduktion til postmodernisme, som dernæst problematiseres og kritiseres, hvorefter vi foreslår en alternativ tilgangsvinkel til forfatter-rollen. Vi har valgt denne fremgangsmåde, da det fremherskende paradigme ikke giver noget grundlag for, hvordan vi som skaberne af et værk skal forholde os til det – perspektivet på et værk er publikum-centrisk. Afsnittet om kritisk rationalisme går derfor fra et redegørende over i et diskuterende afsnit, og udgør et mere forsvarligt filosofisk fundament for vores proces med at skabe artefaktet, end vi ellers ville have haft.

Herefter vil vi dokumentere vores designproces, hvor vi fulgte Rollings & Morris’ retningslinjer for ideudvikling, og vise vores beslutninger vedrørende hver enkelt af deres designfaser. Så vil vi beskrive artefaktets endelige form, med screenshots og en gennemgang af brugerens muligheder. Dette vil blive fulgt af et længere afsnit hvor vi anvender den relevante teori, og begrunder vores endelige designbeslutninger. Efterfølgende forklarer vi artefaktets dybere struktur med inddragelse af UML og kodeeksempler, og begrundelse for relevante strukturmæssige beslutninger. Endelig vil vi i et konkluderende kapitel reflektere over vores proces, og give forslag til mulige fremtidige udvidelser af artefaktet.

Den komplette kildekode til spillet, Brugervejledning til spillet, samt designskitser og -artefakter fra processen kan findes i bilaget (på CD-ROM).

Teorier

En Konstituerende Definition af Spil

Eric Zimmerman definerer et spil som ”a formal kind of play”[1], hvilket gør det nødvendigt først at forklare hvad de mener med ”play”[2].

> *“Play is the free space of movement within a more rigid structure. Play exists both because of and also despite the more rigid structures of a system.”*[3]

[1] Afsnittet bygger på artiklen “Narrative, interactivity, play, and games”, Zimmerman (2004).
[2] Svært at oversætte til dansk, da differentieringen mellem spil og leg er anderledes (jf. ”to play a game” overfor ”at spille et spil”). Man kunne foreslå ”legende aktivitet,” men for klarheds skyld holder vi os dog til de engelske udtryk.
[3] Ibid. P. 159.

Zimmermans definition af Play relaterer til en af Lars Qvortrups talrige definitioner af kunst[4], nemlig en legende eller kreativ aktivitet indenfor rammerne af et system eller medie, der dels prøver grænser af for at finde ud af hvor meget frirum (på engelsk: "Free play") systemet rummer, dels kan skabe nye systemer og mønstre inde i systemet, som for eksempel Quake-spillere der bruger "rocket jump"[5] til at nå ellers utilgængelige steder i banerne i det spil. Man kan sige at Play bruger et system til noget andet, end hensigten det var blevet skabt til, men uden at ødelægge systemet; eller måske mere præcist: **Når man bruger et system til underholdning, uanset dets oprindelige formål.**

Hvis systemets selve formål *er* underholdning, behøver man naturligvis ikke bryde hensigten for at lave "playful activity": Spilsystemet er med vilje designet med rigeligt "free play." Men før vi binder Play mere op på spil må vi klargøre "interaktivitet" og "narrativ".

Zimmerman definerer 4 modus for interaktivitet, men vi vil dog nøjes med at gengive nr. 3 og 4 til brug i denne opgave. Modus 3, kaldet "explicit interactivity, " er hvad man normalt forstår ved interaktivitet, i betydningen deltagelse i en proces. I computerspil kommer det til udtryk ved, at spilleren er en central del i funktionen af spillet, og der ikke sker noget hvis der ikke er en "vilje" der foretager en handling for at holde spillet i gang. Det kan så at sige kun reagere på input fra spilleren. Modus 4 er "meta-interactivity" eller også "cultural participation," og skal forstås som de referencer på tværs af værker, som præger den postmoderne kultur. I forhold til computerspil kan det ses som både bevidste *homages* til klassiske film og andre computerspil, og ubevidst (i hvert fald når man taler om hvilket udtryk spildesignerne har villet mediere med spillet), i den åbenlyse inspiration computerspil imellem; det vil sige efterligninger af succesfulde spilformer, der udmønter sig i uddifferentieringen i spil-genrer, der kan føres tilbage til et "moder-spil" for den genre: *Dune* for real time strategy-genren, *Doom* for First Person Shooter-genren, og så videre. Ikke at disse spil-eksempler nødvendigvis var de første, men at de slog igennem nok til at inspirere dusinvis af efterligninger, og således deltage i meta-interaktiviteten indenfor computerspil-kultur.

En narrativ er ifølge Zimmerman:

> *"A narrative has an initial state, a change in that state, and insight brought about by that change. You might call this process the "events" of a narrative. [It] is not merely a series of events, [however], but a personification of events though a medium such as language. This [personification or] representation is constituted by patterning and repetition. This is true for every level of a narrative, whether it is the material form of the narrative itself or its conceptual thematics."*[6]

Vi kan se at denne definition har nogle ting til fælles med Play, nemlig "patterning and repetition" som et udtryk for kreativitet og systemdannelse. Det vil her være nyttigt kort at tage Neitzels tre begreber, og bagefter binde dem op på Zimmermans narrativ-begreb. *Derefter* se alt dette i forhold til spil.

[4] Qvortrup, Den Nye Eftermodernitet (1998), p. 47.
[5] Et ekstra højt hop man kan udføre i flere shooter-spil af den mere urealistiske, *arcade*-agtige slags, hvor man bruger en rocket launcher til at skyde på jorden under sig, så man bliver skudt højt op i luften, på bekostning af en god portion hitpoints.
[6] Ibid. P. 156, ord i hårde parenteser er vores.

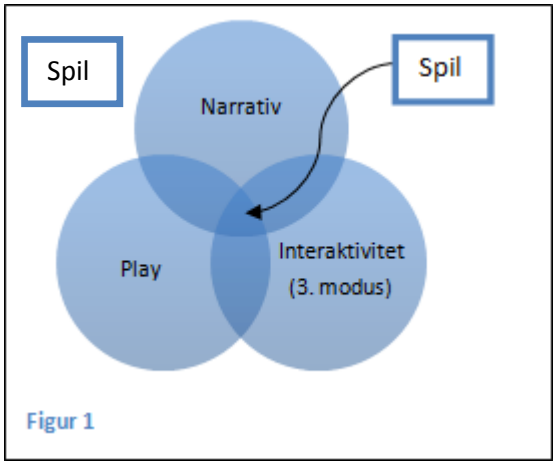
Britta Neitzel anvender begreberne Fabula, Syuzhet og Scripton. Fabula kan beskrives som "det der skete" i historien, altså alle begivenheder i en given fortælling, og deres årsagsforhold til hinanden. En tabel af begivenheder er dog ikke nødvendigvis den mest gribende måde at fortælle en historie på, så man "pakker det ind" eller "navigerer" fabulaen via syuzhet'en, som er den lineære fortælling man viser et publikum, for at de så selv kan sammensætte fabulaen. Man kan vælge at fortælle en historie gennem forskellige medier, og den konkrete form man vælger, det færdige produkt med fortællingen indeni, som evt. kan masseproduceres, er Scripton.[7]

I forhold til Zimmerman vil Scripton så være personificeringen eller repræsentationen af narrativen i et medium, og således også den ene form for "patterning and repetition", nemlig dette mediums sprog og konventioner. Syuzhet i er altså mediets sprog, det der kommunikerer til og engagerer modtageren; indenfor spil må det siges at være Play og interaktivitet, der er det engagerende element. Spilleren projicerer sig ind i mediet på en helt anden og dybere måde end for eksempel en film. Man kunne forestille sig to personer, der spillede et multiplayerspil imod hinanden, og den ene skød den anden. Han ville så muligvis udbryde "Ha! Jeg skød dig!", men nok ikke "jeg skød din virtuelle avatar!" Eller for den sags skyld "min avatar skød din avatar!" Man kan godt blive grebet og engageret af en film, men det sker højest på empati-niveau; man føler med personerne i filmen, men den handler stadig om dem, ikke om beskueren selv. I computerspil er det mere end empati; det er direkte personificering.

Således er vi endelig kommet til spil-begrebet. Det var som sagt "a formal kind of play" ifølge Zimmerman, og yderligere:

> *"A game is a system in which players engage in an artificial conflict, defined by rules that result in a quantifiable outcome."*

Og vi skal huske at spil også er indeholdt i hvert af de andre tre begreber, Narrativ, Play og Interaktivitet. Ved at "Play'e" spillet, altså handle interaktivt og legende indenfor spil-systemets regler, beskriver man et lineært forløb, altså en syuzhet, igennem spilverdenen, som så kan fortolkes som en narrativ.



Figur 1

Der er flaskehalse hvor frirummet er så lille, endda endimensionelt (for eksempel imellem spillets baner, bestemte missioner der skal udføres eller cutscenes), at disse steder vil optræde på samme måde i alle de

[7] Neitzel, Narrativity in Computer Games (2005), p. 231

syuzheter som spil-mekanikken kan skabe i samspil med spilleren. Her rammer vi også den fundamentale forskel mellem spil og traditionelle, lineære medier: Et spil kan have flere syuzheter, på en måde, siden hver gennemspilning vil følge en ny rute. Man kan yderligere sige at det også kan have flere fabulaer, siden forskellige gennemspilninger også kan gennemgå forskellige begivenheder. Det forholder sig modsat i film eller bøger, hvor der ingen egentlig "verden" er udenfor scriptonen; man kan måske forestille sig Tolkiens Frodo Baggins sige og gøre noget andet end det, der er nedfældet i Lord of the Rings, men det er ikke tilfældet, og det ville ikke være "kanonisk". Hvis de malende beskrivelser i en bog eller kulisserne i en film virker overbevisende, og man forestiller sig en helstøbt verden, er det alt sammen noget der foregår i hovedet på modtageren: Den illuderede, storslåede fantasiverden i f.eks. Lord of the Rings (både bogen og filmene, for den sags skyld), er en trompe l'oeil der kun eksisterer ud fra lige præcis kameraets/tekstens synsvinkel.

I spil-mediet *er* der til gengæld tale om en "helstøbt verden", måske ikke hundrede procent realistisk eller gennemført i de spil der er blevet produceret hidtil, men skabelsen af komplette, virtuelle verdener er det ideal, mediet bevæger sig i retning af. Spilverdenen er mere end en filmkulisse, den ligner mere en forlystelsespark[8], idet at der *er* en bagside og objekterne i spillet bare *er* polygonfigurer der er hule indeni, men illusionen er robust nok til at spillere kan slippes løs på egen hånd i verdenen, og ikke er nødt til at blive ført rundt af filmkameraets perspektiv.

Jf. Qvortrup er kunstnerens opgave i vores tidsalder at skabe fiktive verdener vi kan tænke os ind i, og i spændingen mellem virkeligheden og denne tænkte verden bane vores egne stier.[9] Computerspil manifesterer denne kunst-opgave bedre end medier som bøger og film ved at være interaktive og have frirum.

Vi må dog i vores behandling af spilmediet hele tiden holde os for øje, at det vigtigste ved spillet er underholdningsværdien, og ikke nødvendigvis en høj grad af realisme. Realismen kan hjælpe på underholdningsværdien, men kun til et vist punkt. Rigtig krig er jo f. eks. ikke sjovt at deltage i; det er hårdt og træls og fuldt af traumatiserende død og ødelæggelse. Men et krigsspil er tilpas distanceret og medieret igennem spilmediet, til at krigen kun fungerer som en kulisse der bidrager til spændingen. Det gælder således om at finde et balancepunkt mellem tilstrækkelig realisme til at man kan fordybe sig i spiluniverset, men samtidig tilstrækkeligt med abstraktion til at det er tilgængeligt for almindelige mennesker, der ikke er trænede soldater eller racerkørere el. lign., men bare gerne vil underholdes. Dette balancepunkt er forskelligt fra medie til medie (et populært observeret fænomen, er hvor stor forandring der sker, når en bog bliver filmatiseret). Dette gør sig også gældende for spilmediet. Engagementet i spil skabes som regel ved at implementere kriterier for hvordan man "vinder", eller har succes, der ikke nødvendigvis har meget med virkeligheden at gøre. I den forbindelse kan også nævnes, at "konflikten", som Zimmerman nævner som en bestanddel af spil, ikke nødvendigvis er det mest optimale ordvalg. Af samme årsag supplerer vi også senere i dette teorikapitel deres spildefinition med Jesper Juuls præcisering af samme, som han diskuterer og udlægger i sin bog Half-Real. Udover potentielt arbitrære succeskriterier, er der tekniske konventioner, såsom at kunne pause, gemme og loade et spil, der ikke på nogen måde er realistisk, men er nødvendige fra et praktisk og underholdningsmæssigt synspunkt, for at give spilleren frihed til at afbryde

[8] En sammenligning Esben Aarseth brugte ved en forelæsning i Århus d. 4/4/08
[9] Qvortrup, Den Nye Eftermodernitet (1998), p. 50.

interaktionen når som helst. Disse uvirkelige elementer er med til at forme det udtryk, man kan opnå i spilmediet, og sætter grænser for hvad der kan medieres.

Der findes spil overalt i spektret mellem regler og simulation, fra syvkabale der ikke kan siges at foregå i et "univers" på nogen meningsfuld måde, men udelukkende er manipulation af symboler efter et system af regler, til flysimulatorer som man ikke kan "vinde", men udelukkende er en genskabelse af et univers, uden spil-regler. Denne påstand om et spektrum af spil mellem abstraktion og repræsentation er dog blevet kritiseret; spilforsker David Parlett argumenterer imod en sådan skelnen ved at mene at spilleres oplevelser af spil er så subjektive, at et givent spil ikke kan placeres på spektret på en objektiv måde (Juul, 2005, s. 130) – en spiller vil måske leve sig ind i spilverdenen, mens en anden vil se bort fra al "pynten" og tilgå spillet udelukkende som et system af regler. Når man har med underholdning at gøre, er det selvfølgelig svært at definere objektivt hvad der er sjovt, og hvad der ikke er. Dog har Juul påpeget at interaktionen med spil foregår i en "*Lusory Attitude*" (begrebet er skabt af Bernard Suits i 1978[10]; afledt af "*ludus*", latin for "spil"). Denne interaktionstilstand defineres ved at spilleren respekterer de opsatte regler, og holder sin interaktion indenfor de opsatte rammer. I et digitalt medie er det selvfølgelig umuligt at gøre ting i et spil, der ikke tillades af reglerne, da den "verden" spillet lader til at foregå i, kun eksisterer i kraft af disse regler. Det vil sige at en handling der bryder med spillerens *Lusory Attitude* lettere kan forekomme i gammeldags brætspil, hvor brikker og handlinger eksisterer i den fysiske verden, og derfor kun er "spil" når spillerne skaber en konvention derom; Spil i det digitale medie er altid spil, selvom de ofte repræsenterer en meget realistisk verden (som for eksempel first person shooter-spil).

Regler

Reglerne i et spil er faste og restriktive, men spillet (eller legen) udfolder sig i den tid og det rum som reglerne udspænder, og legen er vilkårlig og åben og på mange måder modsætningen til reglerne. Denne sammensætning af regler og leg er et af de interessante aspekter ved et spil.

Juul (2005, s. 62-67) definerer flere former for regler, som de gør sig gældende i spil. Han refererer Goffmans (1972, s. 19) "irrelevans-regler". Når man spiller et spil, ignorerer man mange aspekter af virkeligheden og den givne kontekst, og fokuserer på enkelte dele af konteksten. Juul omformulerer Goffmans regel til, at enhver spilregel har en *relevansregel*, som specificerer hvilke aspekter af spillet og spilkonteksten er relevant for den givne regel. Disse relevansregler er hvor spilreglerne og spilfiktionen mødes, således at en spiller kan lære at reducere informationen, der bruges til at spille. En regel skal være veldefineret for at klart vise sin relevansregel, og for at vi nemt kan afgøre, hvornår reglens betingelse er opfyldt.

I et computerspil er alle regler klart og eksplicit defineret af spiludviklerne, som udvikler dem i samarbejde med testbrugere, som i første omgang er udviklerne selv, og i visse tilfælde "almindelige" spillere i åbne eller lukkede betaer. I disse spil er der en skelnen mellem hvad der er en regel der *kan* håndhæves og hvad der er en regel som spillere accepterer, da spillere mister interessen for spillet, hvis reglen ikke er tydelig – "hvorfor skete det der?"

Visse regler er ikke klart eksplicitte, såsom fair play. I sport er der typisk associeret ideer om fair play, som er guidelines som spillere typisk følger. Juul snakker om tre former for fair play

[10] Jesper Juul(2005) p. 38

1) Undgå fysisk skade. Selvom reglerne ikke forbyder det.
2) Vedligeholde fairness i tilfælde af *force majeure*. F.eks. hvis en computerspiller er nødt til at gå på toilettet, så pauses spillet.
3) Holde spillet interessant. Hvis en spiller er totalt overlegen, vil vedkommende give den/de andre spillere større chancer, ved at tage større risikoer for at holde spillet interessant.

Disse fair play regler er ikke faste regler, men mere guidelines, som en spiller kan vælge at følge eller ej og det indeholder tvetydighed.

Derudover er der de regler, som baserer sig på vores naturlige love, såsom tyngdekraft, som bliver brugt i mange spil i dag. Disse regler er eksempler på spils løse forbindelse med vores egen verden, og bliver approprieret efter forgodtbefindende, hvis udviklerne føler der er behov for det.

Juuls spiltyper

Jesper Juul fremviser i sin *Half-Real* (2005) et yderligere supplement til Zimmermans spilteori, baseret på spillenes fiktive verdener. Han foreslår at alle fiktive verdner er ufærdige simulationer af en verden, hvorfor spilleren selv må digte videre på verdenen ud fra de hints der er i spillet og fra spillerens egne erfaringer i sin egen verden. Ud fra dette aksiom opstiller Juul 5 kategorier af spil:

- Abstract Games
- Iconic Games
- Incoherent World Games
- Coherent World Games
- Staged Games

Abstract Games er defineret ved at de enkelte dele i spillet ikke repræsenterer noget andet, men eksisterer udelukkende i kraft af spillet og spillets regler. Der er således ikke nogen skjult mening om verden udenfor i spillet, det eksisterer i sig selv. Eksempler på disse kan være abstrakte brætspil som Dam og Kryds og Bolle, eller computerspil som Tetris.

Iconic Games er lidt mere avancerede, idet nogle af spillets elementer har en ikonisk mening, og derfor bærer mere mening end spillet i sig selv. Juul giver eksemplet med et spil kort, hvor der er både konger og dronninger, som bærer meningen af virkelighedens konger og dronninger. Nærmere end dette er der ikke nogen binding mellem den ikoniske mening og spillets regler.

Incoherent World Games har et større univers end et ikonisk spil, men forhindrer spilleren i at fylde universets "huller" med sine egne erfaringer. Spillet kan modsige sig selv eller udføre handlinger, der ikke kan forklares i spillets eget univers. Skak kan ses som en konflikt mellem to kongedømmer, men de enkelte brikkers ryk er ikke baseret på det univers; de er arbitrært bestemt i spillets regler.

Coherent World Games er relativt simple at forstå. De indeholder et udsnit af en fiktionsverden, hvor verdenen er større end det spille præsenterer som interaktivt. Vi er således ikke på nogen måde forhindret i at forestille os spillets verden i en meget højere detaljegrad.

Staged Games er et spil i et spil, hvor man kan spille et abstrakt eller mere detaljeret spil inde i en meget større spilverden. Eksempler på dette kan ses i mange spil - Black & White havde en udgave af Towers of

Hanoi, Prey havde en spilleautomat med et lille Space Invaders lignende spil og World of Warcraft bruger en udgave af Simon Says til en af deres gentagende quests.[11]

Denne kategorisering virker ifølge Juul på et rent tekstuelt niveau, men kan også fungere i spillerens sind, hvor vedkommende kan tolke videre på sit mondæne spil fodbold, og forestille sig at han selv er en fodboldstjerne og derved mikse det abstrakte spil, fodbold, med en fiktionsverden som han selv forestiller sig. Han citerer endvidere det klassiske eksempel på overfortolkning, nemlig Janet Murrays tolkning af Tetris som en perfekt genskabelse af de overarbejdede amerikanere i 1990erne, da det viser det konstante bombardement af opgaver som kræver deres opmærksomhed, og at de skal organisere opgaverne så de kan have plads til flere opgaver. Hun laver hermed en tolkning som de russiske udviklere slet ikke havde intenderet, og således er det vist at et spil kan læses som en allegori på noget andet, men at nogle læsninger er mere overbevisende end andre.

Ideudviklingsprocessen - "Creative Road Map"

Vi vil i udviklingen af vores spilkoncept støtte os til bogen Game Architecture and Design, som udlægger en struktureret fremgangsmåde for konceptudvikling[12]. Vi vil i mindre grad følge deres vejledning for systemudvikling, fordi bogen her er målrettet imod større spilproduktioner, og vores målsætning er blot at skabe et lille browserspil.

Rollings og Morris identificerer 3 kreative evner man skal gøre brug af i enhver ideudviklingsproces: Kreativitet (*creativity),* håndværk (*craft*) og teknik (*technique*). Kreativiteten er selve dét at få ideen til hvad man vil lave. Håndværket er det forberedende arbejde (f. eks. en kulskitse eller forstudie til et oliemaleri), og teknik er selve arbejdet med at producere det færdige værk. Når værket er et spil foreslår Rollings og Morris en lidt anderledes metode, hvor de tre evner modsvares af tre faser i ideudviklingen: Koncept, struktur og design. Koncept er så den originale ide eller vision for spillet (eller produktet). Struktur er implementeringen af visionen i et medie, hvor man foretager en kritisk behandling af konceptet, hvor eventuelle huller (uoverensstemmelser mellem vision og medie f. eks.) findes og repareres, og hvor den rene ide udarbejdes til et endeligt, rigt detaljeret koncept, der er kongruent med mediets styrker og begrænsninger. Den sidste fase af det konceptuelle arbejde, designfasen, er udarbejdningen af det endelige produkts fremtræden, finindstillingen af spillets mekanismer og konkrete detaljer.

Koncept-fasen underinddeles yderligere i 4 dele: Inspiration, syntese, resonans og konvergens. Gennemgangen af disse faser skal resultere i et *First Concept*-dokument, der bruges som grundlag for det videre arbejde.

Den første fase, Inspiration, er i sagens natur svært at sætte en finger på, og fasen får da også den kommentar med på vejen, at spilmediet har været mindre frugtbart for originale ideer end f. eks. filmmediet – spildesigneren opfordres til at opsøge nye oplevelser for at komme ud af den vanlige trummerum og få en samling originale ideer.

Når man har fået en samling ideer, går man til syntese-fasen, hvor formålet er at samle ideerne til et interessant hele. Her advares spildesigneren imod at benytte klicheer, da dette er en faldgrube der er let at falde i, i denne fase. Man skal afholde sig fra at tænke analytisk i arbejdet med at kombinere ideerne, da

[11] Juul (2005), pp.131-132.
[12] A. Rollings & D. Morris (2004) p. 3-34

man så ofte kommer til at lede efter tidligere eksempler på samme sammenstilling af ideer, og dermed ender med en kliché. Derimod skal man tænke kreativt, og forestille sig "naturlige" og interessante konsekvenser af sammenstillinger af ideerne, så det resulterer i kimen til et univers med en intern logik, som man siden kan bruge som basis for spillets regler.

Når dette er gjort går man til resonans-fasen. Her skal man få ideen til at blive en engagerende fiktion for spilleren, ved at appellere til temaer, spilleren kan relatere til. Her kan man eksempelvis bruge klassiske temaer fra litteraturen, pendanter til historiske begivenheder eller perioder, og traditioner fra diverse kulturer. Rollings og Morris påstår dermed at det er muligt for skaberen af et værk at *vælge,* hvilke temaer et værk kan indeholde, og at brugeren vil udlede de samme temaer.

Efter spilkonceptets resonans er udredet, skal man sørge for "konvergens", hvor man for første gang skal forholde sig kritisk til sit koncept – dette har været frarådet i de tidligere faser; argumentet lyder, at det kritiske perspektiv ikke tillader at spæde ideer kan overleve. På dette stadie begynder man dog at bevæge sig ind i håndværksfasen ("*craft*"), hvor man kan vurdere enkeltdelene i forhold til hinanden på et teknisk grundlag, og prøve at forudsige fejl – som de siger: "An accurate critical sense can save thousands of dollars further down the line.

Denne kritiske tænkning kommer også til at præge de følgende faser, og glider over i Structure-tænkningen. Her skal man implementere gode gameplay-principper – som beskrevet af Rollings & Morris, der citerer spilguruen Sid Meier: "A game is a series of interesting choices"[13] – i gameplay-fasen skal designeren fokusere på at gøre spillerens interaktion med spillet interessant. Dette er hovedsagligt et spørgsmål om at finde og løse "the Dominant Strategy Problem" de steder det optræder i spildesignet. Dette problem er karakteriseret ved et spil hvor det er tydeligt, at et bestemt valg er det optimale i en situation, og ikke kræver nogen speciel omtanke fra spillerens side. Hvis en bestemt strategi altid har den højeste succesrate, og ydermere er let for spilleren at identificere, bliver alle de andre valgmuligheder, spillet måtte tilbyde, aldrig udforsket af spillerne, da der ingen motivation er herfor (i hvert fald ikke en succesbaseret motivation). Derfor er det op til designeren både at camouflere eventuelle stærke strategier, eller gøre dem tilpas besværlige (et eksempel kan være de stærkeste angreb i et kampspil som Tekken-serien, der kræver timers øvelse at blive i stand til at udføre), eller balancere spillerens valgmuligheder, så vedkommende opmuntres til at afprøve og udforske så mange kroge af spiluniverset som muligt, og dermed få en mere komplet spiloplevelse, og slippe for den hurtige kedsomhed, monoton gentagelse af den samme strategi igen og igen kan give.

Programmeringsteori

Objekt Orienteret programmering

Vi vil her kort forklare hvad objekt orienteret programmering (OOP) går ud på, og vise fordelene ved at benytte dette paradigme.

Det helt basale i OOP, er at man bruger to koncepter kaldet *objekter* og *klasser*. Et objekt er karakteriseret ved tre egenskaber:

- Tilstand

[13] Rollings, Morris (2004), p. 61

- Opførsel
- Identitet

Et objekts *tilstand* (="state") er alle de værdier der bliver gemt i objektet. Et objekts tilstand kan ændres ved at køre funktioner på objektet, der ændrer denne tilstand. Denne tilstand kan have en indflydelse på hvordan objektet opfører sig ved givne funktioner. Tænk f.eks. på et bil-objekt, dets benzintank kan være tom eller ikke-tom, og alt afhængig om det er det ene eller det andet, så kan bil-objektet køre eller ikke.

Et objekts *opførsel* er alle de interne funktioner (også kaldet *metoder*), som objektet understøtter. Et givent objekt kan understøtte visse metode, men altid kun et udsnit af alle mulige. En bil kan f.eks. køre, men ikke flyve – dvs. at et bil-objekt må have en funktion der hedder `kør()`, men ikke en der hedder `flyv()`.

Et objekts tilstand og opførsel er dog ikke nok til at definere et objekt, da der på samme tid kan eksistere to objekter med identisk tilstand og opførsel, som er forskellige. Det er her et konceptet *identitet* kommer ind i billedet. To biler kan være identiske på alle måder, men det er stadig ikke den samme bil.

Definitionen på et objekt er altså, en enhed med en tilstand, en opførsel og en identitet.[14]

En *klasse* kan ses som en form man kan lave objekter ud fra. En klasse er en måde at gruppere lignende objekter i et overskueligt system. Objekter der bliver skabt ud fra en given klasse understøtter de samme interne funktioner og har samme mulige tilstande, og derfor skal en klassedefinition definere to ting:

1. De metoder/funktioner, der er tilladte i objekter fra denne klasse
2. De mulige tilstande objekter fra klassen kan være i.

Hvis vi kigger på et kort spil, kan man således lave en `Kort` klasse, der indeholder alle mulige tilstande for kort i et givent spil (kulør og nummer) og de relevante metoder der skal bruges på kortet, f.eks. for at få fat i nummer eller kulør.

To vigtige fænomener ved OOP er *indkapsling* og *nedarvning*.

Indkapsling er at et givent objekt ikke afslører noget af sin interne algoritmer for omverdenen. Dette betyder, at hvis vi kalder en af objektets metoder, som f.eks. `getName()` så ved vi ikke hvordan dette navn er gemt i objektet - det kan f.eks. være genereret ud fra andre oplysninger, og dette har vi ikke nødvendigvis viden om. I en metode ved vi dog hvad der returneres, så kan vi stille os tilfredse med det og blot benytte den. Objektets interne virkemidler er skjult for omverdenen, og vi kan kun stille spørgsmål, i form af metoder, og få svar.

Nedarvning er en måde at forenkle klasser på, ved at lade en "forælder" inkludere den overordnede opførsel og tilstand for en familie af klasser, hvor et "barn" af denne, så vil arve opførsel– og tilstandsdefinitionen fra den overordnede klasse. Det smarte ved dette, er at hvis der er mange metoder og gemte informationer, der går igen i klasser, kan man sætte dem i familie, og forenkle hver klasse, og spare kode. Endvidere betyder en nedarvning, at man kan benytte den fælles forælder type til at arrangere mange af børnetyperne.

[14] En interessant betragtning, er at ud fra denne definition er alle ting i den virkelige verden objekter, hvilket på en måde, også er korrekt.

Et eksempel: hvis vi skal implementere et system der kan håndtere en cykel og en bil, kan vi med fordel lave en klasse der hedder **`Køretøj`** (som kan indeholde basale ting som **`hastighed`**, **`hjul`**, **`farve`** og **`kør()`**) og lade **`Bil`** og **`Cykel`** klasserne nedarve disse. **`Bil`** klassen kan så selv indeholde mere specifikke informationer og metoder, så som **`motor`**, **`hestekræfter`** og **`startMotor()`**, hvor Cykel klassen kan indeholde **`ringeklokke()`**, **`antalSadler`**, osv. Et givent bil-objekt, der er skabt ud fra Bil klassen, vil så indeholde den samlede opførsel og tilstand fra Bil klassen og Køretøj klassen. Hvis man havde en liste af køretøjsobjekter, kunne man således også bruge **`kør()`** på dem uden at vide hvilken egentlig type den havde, da kør metoden ville kunne overrides i barnetypen, således at **`Bil`** og **`Cykel`** typerne bestemmer internt hvad der egentlig sker, f.eks. forbrænding af benzin overfor træthed.[15]

OOP er specielt fordelagtigt, hvis man vil lave en model af den virkelige verden, da vores hjerner observerer objekter omkring os konstant, og vi tænker derfor i objekter med egenskaber, tilstande og identiteter.

Programmeringsmønstre

Model-View-Controller(MVC) er et programmeringsmønster som er baseret på et Observer mønster, som betyder at et objekt kan have "observers", der lytter på om der sker visse ændringer.

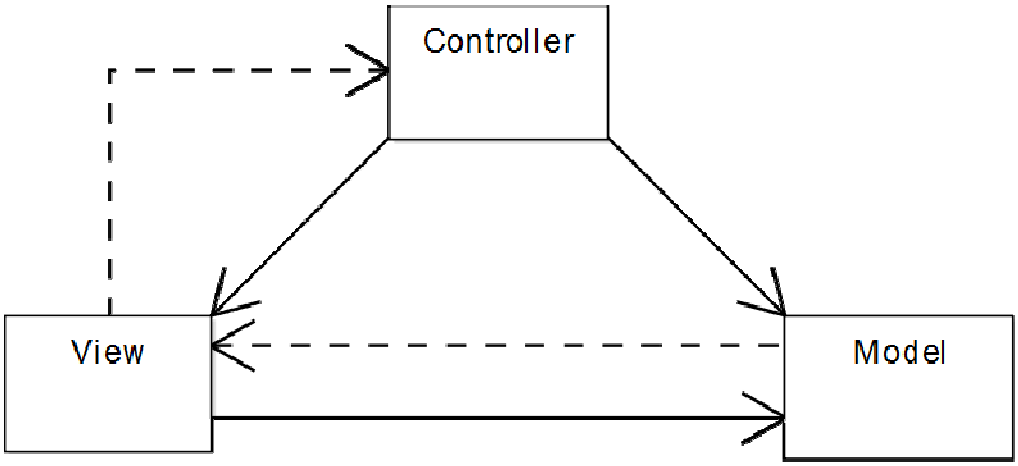


Fig. 1 - Model-view-controller diagram

Ovenstående vises en oversigt over MVC mønsteret, hvor de solide linjer viser direkte associationer, mens de stiplede er indirekte associationer. View er en, typisk, visuel fremstilling af modellen, og man kunne således have flere views koblet til en enkelt model. Model er den underliggende domæne specifikke information, som arbejder med den rå data. Controlleren er et specielt mønster i sig selv, som er det objekt bag UI laget, som håndterer interaktionen mellem view og model.[16] Ved at splitte de tre dele fra hinanden, reducerer man kompleksiteten i programmet.[17]

Singleton

En singletonklasse er en klasse der altid kun har et enkelt objekt. Det unikke objekt fungerer som et globalt objekt, som alle andre objekter kan referere til. Mønsteret sørger for at der ikke bliver oprettet flere af den

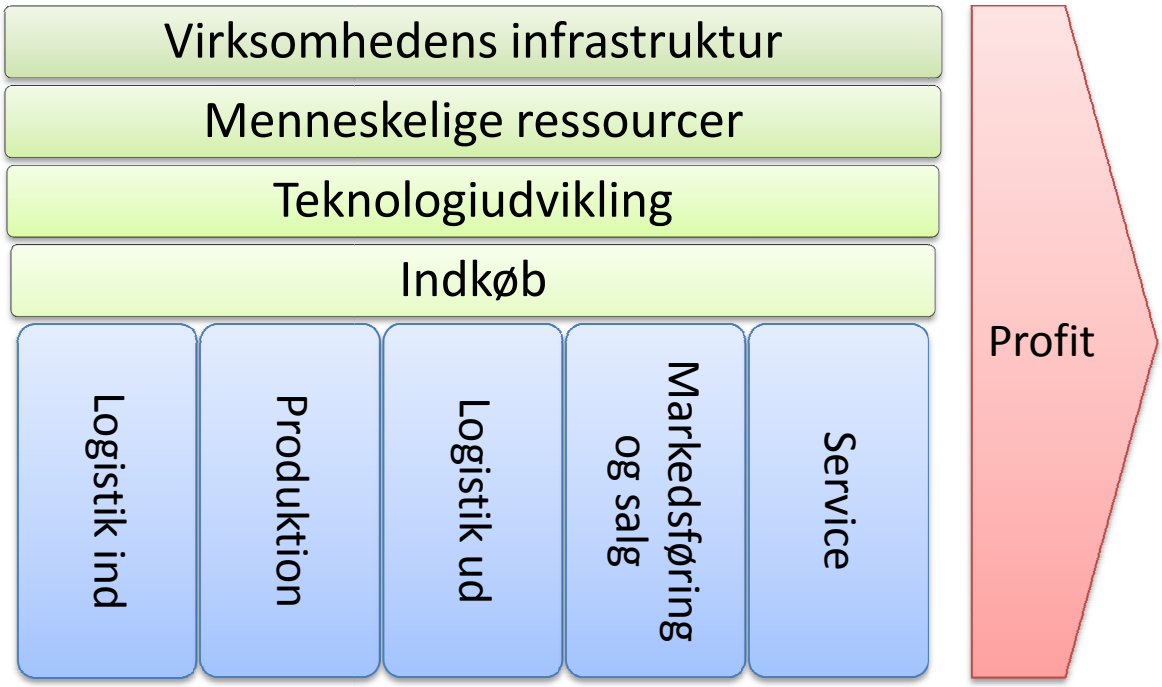
[15] Brookshear (2005), pp.305-311.
[16] Larman (2006), pp. 302-303.
[17] Horstmann (2004), pp. 182-183.

samme klasse, og der i stedet henvises til den allerede skabte singletonklasse. Dette gøres ved at der laves en static metode, som returnerer denne enkelte metode. Nedenstående ses et eksempel på dette.

```
ActionScript 3
static public function get Instance():Controller
{
        if ( instance == null )
                instance = new Controller();
        return instance;
}
```

Porters Værdikæde

Porters værdikæde bruges i virksomheder til at modellere alle de interne aktiviteter, der medfører en merværdi i det færdige produkt, som virksomheden producerer. Hver del af værdikæden tilføjer værdi til produktet, som måles direkte i, hvor meget en kunde er villig til at betale for det. Forskellen mellem omkostninger og produktets værdi er således profitmarginen. Hvis omkostningerne er højere end værdien for produktet, er produktet ikke levedygtigt, og firmaet må opgive dette produkt. Er værdien derimod højere end omkostningerne, er produktet en god forretning, og firmaet kan tjene på at producere dette produkt. Porter inddeler værdikæden i to overordnede dele; primæraktiviteter og understøttende aktiviteter (støtteaktiviteter); som hver har flere underdelinger. De primære aktiviteter har en direkte indvirkning på et produkts værdi, hvor de understøttende aktiviteter, som navnet indikerer, understøtter produktet i den forstand, at de kun har en indirekte effekt på værdien af produktet.



Figur 1 – Porters værdikæde (Blå er primæraktiviteter, grøn er understøttende aktiviteter)

Primæraktiviteter

De første tre primære aktiviteter – *Logistik ind*, *Fremstillingsprocesser* og *Logistik ud* – er enkle elementer at forklare. Det er transport af råvarer (eller andre dele, som skal bruges til det færdige produkt), selve fremstillingsprocessen, og transport af det færdige produkt. Disse tre udgør typisk den største merværdi for produktet. Den næste primæraktivitet, *Markedsføring og Salg*, er den del hvor der kan tilføjes, hvad man kan kalde ”tom” værdi, som baserer sig udelukkende på markedsføring. Et produkt kan give mindre ydelse end den værdi produktet bliver solgt for, sammenlignet med tilsvarende produkter. Denne imaginære værditilførsel er typisk baseret på markedsføring, som hæver køberens opfattelse af produktet, og derfor tilfører en reel værdi, da værdi er defineret direkte, som det en kunde vil betale for produktet. Markedsføring kan ud over at sprede produktet til flere kunder, hvilket ikke nødvendigvis giver nogen merværdi, tilmed fremhæve produktet på omkringliggende faktorer, såsom nationalitet, firmanavn og brand. Disse tre ikke har nogen direkte indvirkning på produktet, men kan højne værdien alligevel på grund af kundens opfattelse af produktet. Den sidste primæraktivitet, *Service*, er i stigende grad en vigtig del i mange virksomheder. Den indbefatter reparationer, vedligeholdelse, assistance og andre former for kundebetjening. Nogle vil måske anskue serviceaktiviteten som en ”efterdel” til selve produktet, men i forhold til værdikæden *er* den en del af produktet, da den også øger værdien på det endelige produkt og nogle gange kan koste ekstra i form af forsikringer og dets lige.

I modellen beskrives herefter ”understøttende aktiviteter”, som dækker alle de aktiviteter i en virksomhed, som kun har en indirekte indflydelse på selve produktet. Virksomhedens interne *infrastruktur* er en underliggende aktivitet i virksomheden, som ikke har nogen direkte effekt på virksomhedens produkt. I virksomheder der strækker sig over flere lande, kan denne infrastruktur hurtigt blive meget omfattende. Der findes myriader af virksomhedsopbygninger, men når en virksomhed overstiger en vis størrelse, er der næsten altid en variant af en hierarkisk opbygning, specielt i toppen af virksomheden – der skal altid være én chef. *Menneskelige ressourcer* omfatter alle ansatte i virksomheden, hvilket heller ikke har en direkte indvirkning på et produkt. Den tredje understøttende aktivitet er *Teknologiudvikling*. Denne har en indirekte effekt på produktet i den forstand at den kan optimere virksomhedens produktionsprocesser, og kan forælde et produkt ved at udvikle nye og bedre produkter. *Indkøb* sørger for både indkøb af råvarer og andre produkter til slutproduktet. Det er indkøbsafdelingens ansvar, at de værdielementer som virksomheden tilegner sig for at kunne producere et tilfredsstillende slutprodukt, erhverves til den laveste mulige pris.

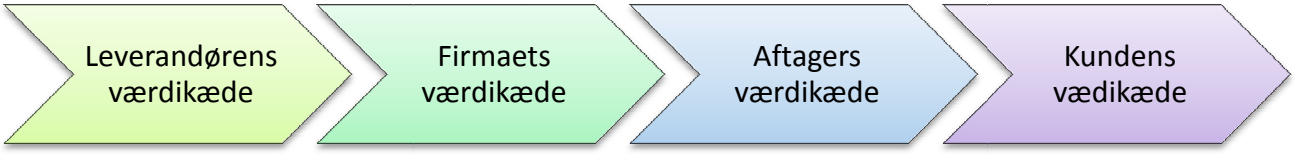
Generelt

En virksomhed er dog ikke kun en sum af dens aktiviteter, men er derimod et netværk af aktiviteter som er internt afhængige. Når disse aktiviteter arbejder fornuftigt sammen, opstår der en synergieffekt, som øger værdien af produktet, og samtidig kan sænke omkostningerne. En virksomhed er nødt til at se på værdikæden som helhed, for at udnytte den ordentligt og opnå konkurrencefordele. Ved at optimere de enkelte aktiviteter sænkes omkostningerne, som derved øger profitten i sidste ende.

Værdisystemet

De enkelte værdikæders netværk indgår sammen i et større system kaldet, værdisystemet (også kaldet forsyningskæden). Værdisystemet afspejler de forskellige led fra råvaren til forbrugeren. For at disse led kan fungere sammen er det vigtigt, at virksomhederne tilpasser deres værdikæde til resten af

værdisystemet. De forskellige værdikæders konkurrenceevne bliver stærkt påvirket af værdisystemet og optimeres i takt med at værdisystemet forbinder de forskellige værdisystemer.[18]



Figur 2 - Værdisystemet (forsyningskæden).

Postmodernisme og Kritisk Rationalisme

Postmodernisme

Da vi begyndte at tænke over mulige koncepter vi kunne lave et spil over, gik det hurtigt op for os at det var problematisk at betragte spillet som et værk, der medierede et budskab eller noget mening fra kunstneren til beskueren. Ikke nok med at adskillige paradigmer igennem det 20. Århundrede har benægtet, at sådan en meningstransferens over hovedet finder sted, men alle de metoder til at læse mening ud af værker, som vi kendte til, er alle skabt til at fungere fra publikums perspektiv; altså modtagersiden af mediet. Kunstnerens beskuelse af sit eget værk var et ukendt fænomen for os, fordi så meget af teorien bygger på netop ønsket om at fravriste ejerskabet over værket fra skaberen, og overdrage det til publikum.

Dette kan man iagttage hos tænkere som Roland Barthes, der i 1968 udgav essayet "Death of the Author", og proklamerede at forfatteren som centrum for tekstens læsning var et forældet perspektiv, af flere årsager. For det første mente han ikke at meningen eksisterede i teksten, men i modtagerens sind, og at det derfor var mere rigtigt at sige, at modtageren skabte teksten i sin læsning[19]. For det andet mente han at forfatteren som enligt geni og skaber af originalitet var en urigtig beskrivelse, og at forfatteren udelukkende sammensatte fragmenter af allerede eksisterende ideer, og ægte originalitet var en umulighed. Endelig mente han, at tvetydighed i tekst ikke kunne determineres til at have den ene eller den anden potentielle betydning, men var en effekt af tekstens særlige karakter som medie:

> *We shall never know, for the good reason that writing is the destruction of every voice, of every point of origin. Writing is that neutral, composite, oblique space where our subject slips away; the negative where all identity is lost, starting with the very identity of the body writing.*[20]

Hans argumentation bygger i høj grad på semiologisk teori, nærmere bestemt hypotesen om at signifiant (tegn) og signifié (mening) er arbitrært forbundne[21]. Det vilkårlige forhold mellem tegn og mening, og det

[18] Porter (1980), pp. 33-36
[19] Death of the Author (1968), Roland Barthes, p. 2
[20] Ibid., p 1
[21] Saussure (1916), p [?] [Cours de linguistique…]

vilkårlige forhold mellem forfatterens intentioner og de intentioner det er muligt at læse ud af en tekst, blev for Barthes ét og samme forhold. Denne tanke fik i anden halvdel af det 20. århundrede stor opbakning, og blev sammen med andre skelsættende udgivelser kimen til den postmoderne kulturstrømning i 80erne. Den nye forfatterrolle fik som centralt fokus at være ironisk overfor sin egen status som skaber, og gjorde sig umage for at citere direkte fra tidligere værker. Det blev et populært synspunkt indenfor mange humanistiske og samfundsvidenskabelige institutter i Vesten, at denne metode til litterær analyse kunne anvendes indenfor stort set alle discipliner; dette synspunkt blev adopteret af forskellige politiske strømninger internt i den akademiske verden, og for eksempel feministiske teoretikere kritiserede det anvendte sprog i såvel fundamentale videnskabelige værker som lærebøger på folkeskoleniveau som "maskulint" eller "vestligt" favoriserende[22]. Postmodernisme kan i det hele taget betragtes som ønsket om at afsløre konventioner eller vaner, som folk (forfattere eller almindelige mennesker) ubevidst følger, og som ofte påstås at have oprindelse hos et samfunds overklasse.

Kritik af postmodernisme

Denne jagt på bourgeois-konventioner kan ses som en marxistisk afledt overbygning på Barthes' tese, og ønsket om at stille spørgsmål til ikke bare værdimæssige konventioner, men til ting der hidtil blev antaget for at være objektive kendsgerninger, kan ses som en reaktion på oplysningstankens positivisme (der også var fremherskende i modernismen, som etymologien bag ordet "postmodernisme" også angiver, at den er imod). Den er også blevet beskrevet som en effekt af de traumatiske oplevelser, modernismens overbevisning af objektiv sandhed forvoldt, eksemplificeret ved de store ideologiers sammenstød i Anden Verdenskrig og den efterfølgende kolde krig.

Dominansen af dette subjekt-centriske paradigme, når det kommer til analyse af et værk, er det umuligt for os at påtage os den klassiske skaberrolle, hvor vores budskab bliver medieret til publikum; i det dominerende paradigme i dag er et værks skaber gjort komplet irrelevant, da al mening påstås at opstå hos beskueren. Hvis læsningen er subjektiv, og ingen mening bliver medieret gennem værket, nedbryder det kunstnerens mulighed for at have en hensigt med sit værk, der behøver end ikke at have været en bevidst handling bag, og der er ikke andet muligt indhold i kunstnerrollen, andet end et den helt generelle (hypotetiske) hensigt, at skabe et objekt der kan facilitere forskellige, endda modsatrettede læsninger. Hvis et værk ikke skabes med hensigten om at mediere, er det irrelevant at der står et bevidst væsen bag, og kunst-kritikere ville være lige så godt tjent med at analysere et meningsløst objekt, som for eksempel en sky, og læse mening ud af det – en god, og ironisk, illustration ville være, hvis man besøgte "the postmodernism generator"[23], og forsøgte at læse mening ud af den tilfældigt genererede tekst.

En alternativ, men mindre politisk motiveret, modreaktion på modernismens autoritære krav på objektiv viden, kan ses hos Karl Popper. Han foreslog sin kritiske rationalisme som en måde, hvorpå man kan opnå "så godt som" objektiv viden, ved at fremsætte hypoteser og forsøge at falsificere dem, og tilskrive en højere og højere grad af troværdighed til hypoteser, der modstår alle sådanne forsøg. Poppers kritiske rationalisme kan ses på flg. måde: Skabelsen af en maskine i den fysiske verden er en reificering af individets eget verdensbillede. Hvis maskinen fungerer som individets verdensbillede forudsiger, er det en bekræftelse af dettes validitet. Erstat 'maskine' med 'eksperiment', og denne påstand er fundamentet for

[22] Som for eksempel den feministiske filosof Luce Irigaray der mener at formlen $E=mc^2$ er diskriminderende, fordi "*it privileges the speed of light over other speeds that are vitally necessary to us*" – Dawkins (1998), p. 141

[23] http://www.elsewhere.org/pomo/

den videnskabelige metode. Denne fremgangsmåde anerkender den fysiske verden og dens lovmæssigheder i højere grad end den radikale relativisme, der er evident hos postmodernismens forkæmpere[24].

På det kulturelle plan har Christopher Alexander, arkitekt og grundlægger af Pattern-tænkningen indenfor programmering kritiseret postmodernismen berøringsangst overfor autoritativ retorik og "Grand Narratives":

"The modernist age, of 'one way, one truth, one city', is dead and gone. The Postmodernist age of 'anything goes' is on the way out. Reason can take us a long way, but it has limits. Let us embrace post-Postmodernism - and pray for a better name. [25] "

Postmodernismens fokus på små narrativer lever videre, men er blevet objektet for vores søgen efter "oprigtighed" og autenticitet, hvad der er blevet kaldt "post-postmodernisme", pseudo-modernisme eller *New Sincerity*. Forfatterens og læserens roller er blevet mindre forvirrende defineret, i forbindelse med den teknologiske udvikling; læseren kan i de nye medier godt siges at være skaberen, men nu er det i bogstaveligste forstand, i form af deltager-drevne TV-koncepter eller Web 2.0-baserede internetmedier. Denne strømning er blevet kritiseret for at have været skyld i den succes underlødig *exploitation*-præget underholdning som f. eks. Realityshows har haft på TV. At forklare tv-produkters lave lødighed med denne nye strømning er dog ikke helt fair, da den lave kvalitet snarere er et resultat af at TV er et passivt medie, hvor kvaliteten kun behøver at være god nok til at sikre at seeren ikke aktivt skifter kanal eller slukker. Internettet er til gengæld et aktivt medie, hvor kvaliteten for et givent produkt omvendt er nødt til generelt at være høj nok til at brugeren aktivt vil opsøge det[26]. Her kan man se for eksempel Web 2.0 fænomener, der virker i kraft af anbefalinger af produkter fra kilder man har tillid til, såsom ens bekendtskabskreds, og hvor de post-postmoderne elementer igen er til at genkende, for eksempel i youtube-videoer, hvor små bidder af "autenticitet" fra hele verden er samlet og tilgængelige.

Spilmediets plads i denne diskussion er interessant, fordi interaktiviteten og regelbundetheden i spil går imod postmodernisme og samarbejder godt med et kritisk rationalistisk verdensbillede, siden den verden, spillet foregår i, *er* defineret af objektive lovmæssigheder, og spilleren *er* i stand til at erkende disse. Skabelsen af en virtuel "eksperimentopstilling", eller bare en udførelsen af en handling med en bestemt hensigt, i et virtuelt miljø (et computerspil) giver naturligvis ingen anden 'objektiv' viden end den som skaberen af det virtuelle miljø har implementeret, men handlingen repræsenterer den samme proces: spilleren opstiller en hypotese om, hvad "naturlovene" i det virtuelle miljø er, og søger at bekræfte dem ved at arrangere en bestemt opstilling af maskinerne. Ydermere kan spildesigneren ganske vist ikke kontrollere spillerens fortolkning af spilverdenen, da denne kan forestille sig ikke-eksisterende elementer, og dette kan influere hans billede af spilverdenen som helhed. dog vil en *komplet* udforskning af spilverdenen (et overfladisk bekendtskab ville være et uforsvarligt grundlag for en læsning af spillet), gøre det klart, om spillerens forestilling var korrekt eller ej[27].

[24] Dawkins (1998)., p. 142
[25] Turner (2008), afs. 1.7
[26] Kirby (2006)
[27] Juul (2005), p. 136

Et godt computerspil belønner denne søgende adfærd, og denne belønning giver spilleren en tilfredsstillelse i form af en følelse af kontrol over det miljø han agerer i. På denne måde vil spillerens færden i spiluniverset, når han er bekendt med det givne univers' regler, være præget af en rationel interaktion, hvor handlinger har logiske konsekvenser, og selv dårlige konsekvenser giver spilleren en følelse af, at den verden han interagerer med er rationelt ordnet, og der er håb om at man kan komme til at forstå den.

Redegørelse for Ideudviklingsprocessen

Dette kapitel følger Rollings og Morris' anbefalede faser. Først inspiration, som de to forfattere beskriver som en temmelig løst defineret proces, som først i de efterfølgende faser bliver underkastet et mere præcist og kritisk perspektiv; inspirationsafsnittet er en række beskrivelser af ideer, vi overvejede i den spæde begyndelse af projektet, og, i de tilfælde vi kan forklare det, hvad der gav os ideerne.

Inspiration

Vi snakkede allerførst om at genskabe klassiske spil som *SimCity*, *X-Com UFO Defense* og *Fragile Allegiance* (Fig. 2). Disse er alle spil hvor man skal administrere et system, tilføje dele og allokere ressourcer, og ideen om at bruge dette som basis for spillets gameplay overlevede også igennem hele vores proces.

Fig. 2 – Fragile Allegiance

Derudover havde vi en vag ide om, at dette gameplay ville fungere godt sammen med et abstrakt *look-and-feel*, og vi overvejede i den forbindelse at lade os inspirere af det eksperimentelle musikinstrument *ReacTable*[28](Fig. 3),

[28] Produceret Pompeu Fabra University i Barcelona. En af de dengang få eksisterende prototyper blev anvendt af Björk på hendes koncertturné i 2008.

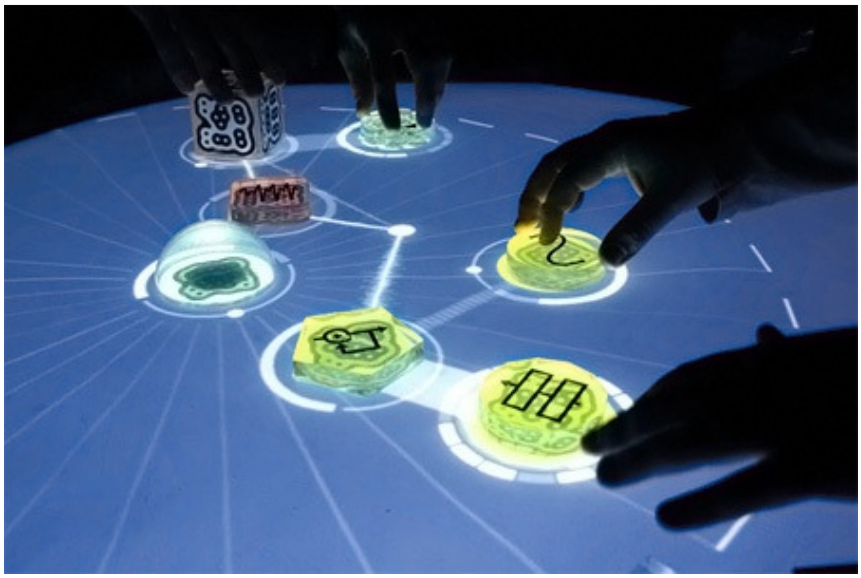


Fig. 3 – Reactable i brug

som er baseret på fiduciary markers[29] samt et grafisk interface der projiceres rundt om, og "augmenterer" de fysiske elementer man kan interagere med. Dets look-and-feel er abstrakt og afdæmpet, med diskrete mørkeblå og hvide farver, der giver mindelser om en ingeniørs byggeplaner trykt på blåt papir. Desuden kan brugeren gribe ind og ændre en fiduciary marker, hvilket resulterer i feedback gennem hele det opstillede system.

Endelig havde vi en ide om at lave et interface der tillod en form for visuel programmering. Her kan nævnes f. eks. browserspillet Lightbot[30] (Fig. 4).

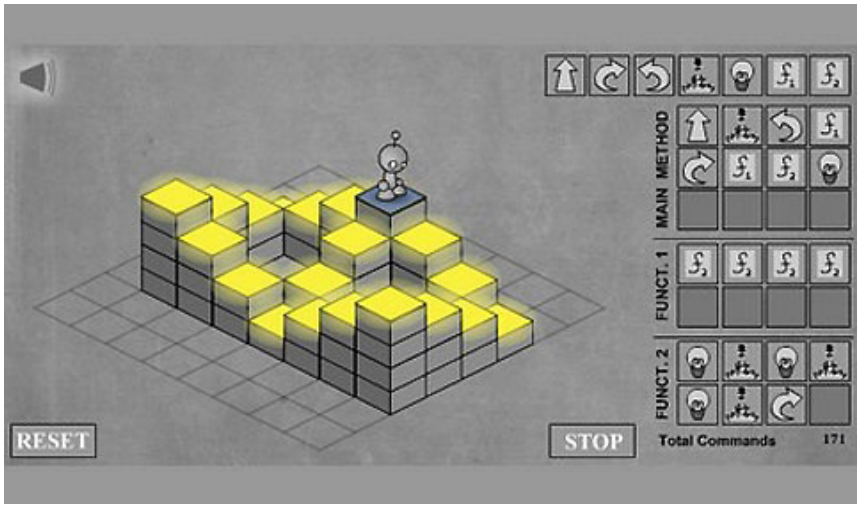


Fig. 4 – Browserspillet LightBot

[29] Et simpelt og unikt grafisk symbol, der kan bruges som ankerpunkt for computersystemer der skal forholde sig til visuelle input.
[30] http://www.weebls-stuff.com/games/Light+Bot/

hvor spilleren kan sammensætte en algoritme, og med den indirekte skal styre en lille robot igennem en labyrint. Denne tanke om at gøre programmeringsprocessen mere tilgængelig syntes vi var interessant, da programmering er et esoterisk håndværk, som ikke-specialister ofte er fuldstændig ubekendte med, og at gøre det til en mere populært udbredt kompetence ville kunne have ganske store konsekvenser for informationssamfundets fremtidige udvikling, på samme måde som bloggen har haft indflydelse på publishing, og multimedie-PC'en har haft det på f. eks. musikproduktion.

På samme måde var vi interesserede i at visualisere Porters værdikæde, da vi syntes den ligesom reactable var god til at giver overblik over komplekse processer, og ville være en god tilføjelse i en dynamisk computersimulation af f. eks. en virksomhed.

Syntese

Syntesen af ideer fulgte i vores proces naturligt, i form af en vurdering af konsekvenserne ved at kombinere de forskellige ideer. For eksempel ville en kombination af et management-spil a la SimCity, og look-and-feel fra Reactable være fordelagtigt, da det dynamiske interface med tydelige indflydelsesforbindelser ville gøre det let for spilleren at bevare overblikket. Når vi tænkte på et sådant interface sammen med et spil som Fragile Allegiance, kunne man forestille sig at spilleren kunne opstille enkelte basedele[31] og få dem til at arbejde som en samlet helhed, på samme måde som brikkerne i reactable skal justeres undervejs hvis resultatet skal være vellydende.

Reactable-ideen kunne også ganske gnidningsløst sammensættes med ideen om et visuelt programmeringssystem, da det ville være nærliggende at associere de enkelte brikker i systemet med klaser i et objektorienteret system, hvor opstillingen af brikker så ville svare til et "live" sekvensdiagramdiagram eller flowchart, hvor man kunne følge cursorens vej igennem programmet, og ændre det ved at interagere med brikkerne.

På samme måde kunne et Reactable-interface kobles med en animeret model af Porters værdikæde, fordi denne på samme måde som et sekvensdiagram, er en illustration af ruter igennem et system. Med et sådant interface ville man kunne overvåge processen i realtid, og justere flowet af f. eks. produkter igennem en fabrik. Porters værdikæde kan som bekendt sættes sammen i sæt, for at kortlægge råvarernes vej og stigende værdi fra de bliver høstet eller gravet op af jorden, til det færdige produkt bliver solgt til forbrugeren. Denne ide kunne let omsættes til et større management-spil, hvor man skulle igennem en serie af baner og konstruere de enkelte værdikæder, der så kunne indgå i en større sammenhæng. Indkapslingen af de underordnede mekanismer i en enkeltdel i værdikæden ville også tillade en ændring af perspektiv fra mikro- til makroniveau, uden tab af overblik, da man ikke ville være tvunget til "micromanagement" af enkeltdelene. Således ville vi kunne overvinde en af begrænsningerne i mange tidligere management-spil, nemlig manglen på uddelegering – denne mangel førte bl.a. i SimCity til at ens by praktisk set havde begrænset vækstpotentiale, fordi hyppigheden af "breakdowns" steg lineært med bystørrelsen, hvilket tidligt i et spil var gode til at sørge for spænding og afveksling i spiloplevelsen, men som i en tilstrækkeligt stor by betød at man ville ende med at bruge al sin spilletid på at adressere disse irritationsmomenter.

[31] Fragile Allegiance går ud på at bygge rumbaser på asteroider, som forposter for et fremtidigt mineselskab.

Resonans

Vores hypotese om at kritisk rationalisme er det paradigme, der bedst passer til spilmediets natur, ville vi gerne se manifesteret i spillets udtryk, og ikke bare dets struktur. Hvis spillet kunne have et udtryk, der støtter en sådan verdensanskuelse, ville det efter vores mening supplere spilmediets egen rationelle karakter godt (og derfor ville være meningsfuld for folk der glade for computerspil). Vi fandt derfor på, at spillet skulle udtrykke den rationelle optimisme, der var udbredt under den industrielle revolution og op til anden verdenskrig, og kom til udtryk i bla. Art Deco-stilen indenfor kunsthåndværk og arkitektur, hvor blandt andet himmelstræbende, magtfulde Prometheus-motiver, og det generelle syn på den naturlige verden var, at snart sagt alle hjørner af den kunne beherskes gennem menneskelig rationalisme og snilde.

For at modvirke det populære billede af entreprenør-arketypen som en kold og hård person, tænkte vi at det ville være fordelagtigt at lave vores management-spil om noget hyggeligt og behageligt: Øl. Yderligere, for at understøtte den optimistiske og kritisk-rationelle ånd, vurderede vi at det ville være bedst at gøre enkeltdelene i spillet så virkelighedstro som muligt, så de ville kommunikere tanken om at være forankret i den fysiske verden, og ikke bare være abstrakte symboler, spilleren ikke ville kunne relatere til noget, og hvis mønstre spilleren derfor ville have svært ved at aflure. Løsningen var at vælge en romantisk, malerisk stil til grafikken, som også genkalder en historisk periode hvor objektivitet ikke var under angreb.

Resonansen i spillet måtte nødvendigvis rettes imod en bestemt målgruppe, da vi ikke kan gøre os håb om at opfylde alle gruppers ønsker. Vi har derfor koncentreret os om at tilpasse spillets ånd til folk som os selv, 20-30-årige der er glade for management-spil. Vores aldersgruppe er vokset op under postmodernismens storhedstid, og er på mange områder så småt begyndt at reagere imod denne strømning, og i stedet søge efter autenticitet og oprigtighed i kulturen. Da vi kun havde tid og ressourcer nok til at skabe et lille spil, valgte vi browserspil-formen, baseret på Flash. Vi valgte derfor at holde spillet på engelsk, fordi publikummet for online browserspil er tilfældigt og internationalt, og det ville være ufornuftigt at begrænse adgangen til et på verdensplan temmelig obskurt sprog som dansk. Browserspil-formen passer desuden også til let underholdning (”casual gaming”), ofte med en munter og optimistisk stemning, så vi følte at denne form passede godt til vores koncept. Endelig passede ideen om at producere øl også fint på den valgte aldersgruppe.

Konvergens

Da vi begyndte at vurdere vores halvvejs formede ide kritisk, blev det tydeligt at det abstrakte formsprog i Reactable ikke passede særlig godt med ønsket om en malerisk grafisk stil. Vi var derfor nødt til at droppe den minimalistiske æstetik. Vi beholdt dog Reactables ide om manipulation af ”brikker”, som i vores koncept kunne oversættes direkte til maskiner på et fabriksgulv, og siden det ville passe dårligt med æstetikken at animere den enkelte maskines tilstand med et minimalistisk ”overlay” direkte omkring maskinen (da det ville stå i skarp kontrast til spillets romantiske, dekorative æstetik), valgte vi at vise hver enkelt maskines information i et kontekstvindue udenfor spilverdenen, i et kontrolpanel. Hver maskine ville stadig kunne manipuleres i detaljeret grad, ved at indstille dens hastighed, eller opgradere den til et højere niveau.

Redegørelse for spillets endelige form

Beer Factory er et strategispil, som visuelt repræsenterer ølbrygningsprocessen og illustrerer konceptet om Porters værdikæde ved at være en dynamisk visualisering og mediering af dens principper.

Spillet starter med en titelside, som via illustrationen indikerer vores intentioner med spillet. Man starter selve spillet ved at klikke på ”start game”-knappen.

Når spillet startes, ser man et tomt fabriksgulv med en lastbil bakket ind i hver side, samt et kontrolpanel i højre side af vinduet, (illustreret i **Fejl! Henvisningskilde ikke fundet.**).

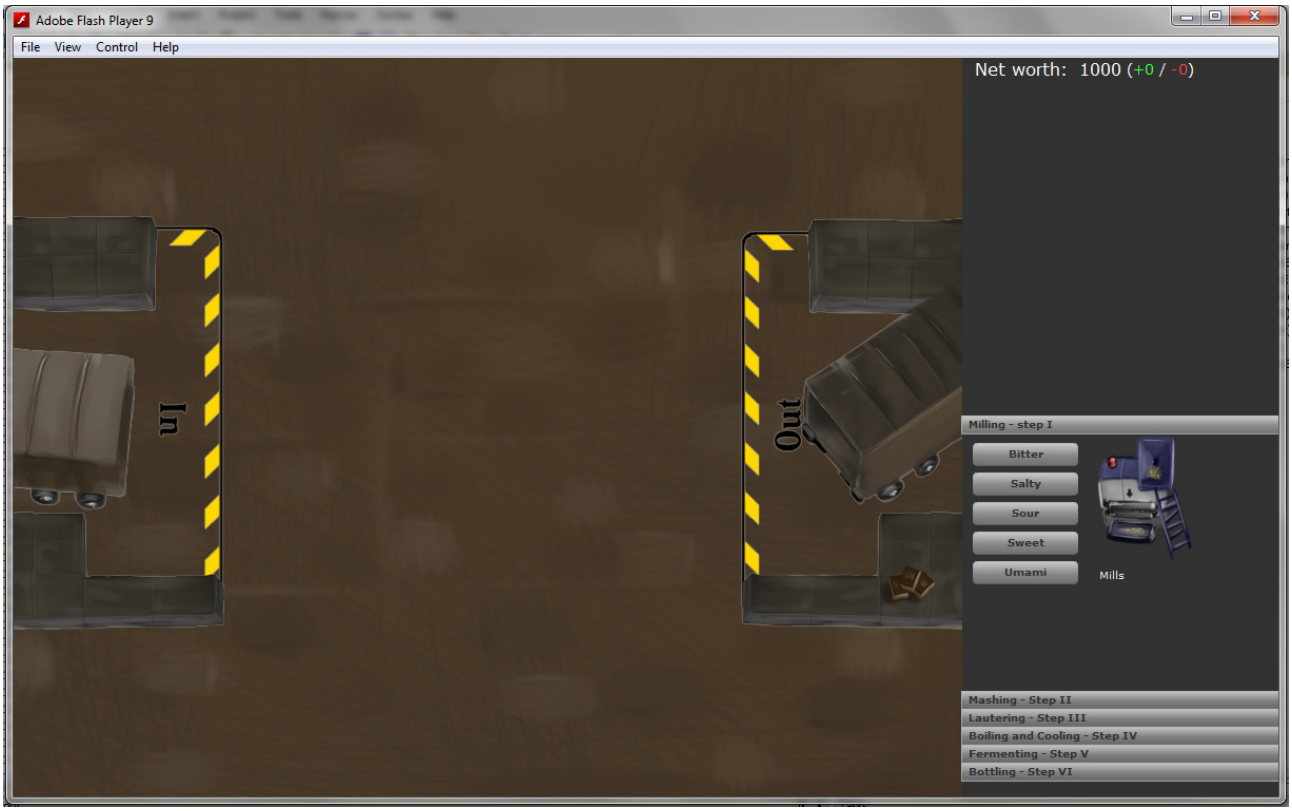


Fig. 5 - det tomme fabriksgulv

Øverst i kontrolpanelet står ens ”Net Worth”, der viser den samlede værdi af ens aktiver, det vil sige profitten fra den samlede ølproduktion, startkapital, samt en plus- og minusangivelse af den seneste omkostnings- og indkomstrate. Herunder er der et kontekst-sensitivt område, der ændrer indhold alt efter hvad man klikker på i ”gameworld”, altså inde på fabriksgulvet. Man styrer spillet udelukkende med musen, og hvis man markerer ”in”- eller ”out”-områderne, ser man kun navnet på objektet og en ”connect to”-knap; men hvis man klikker på en (på forhånd købt) maskine, vil man se navnet på maskinen, derunder en slider der kan indstille maskinens hastighed, på en skala fra 10-100% kapacitet. Herunder står ”output/sec” der angiver antallet af øl der bliver sendt videre fra maskinen. Herunder er produktionsomkostninger pr. sekund. Disse to parametre ændrer sig begge dynamisk alt efter hvad man stiller kapacitets-indstillingen på, og deres indbyrdes forhold er parallelt (men kunne ændres i en fremtidig version til at opfordre spilleren til en bestemt opførsel – mange parallelle maskiner, der kører på vågeblus, eller få maskiner der kører ved deres maksimale hastighed). Endelig har hver maskine i lighed med ”in” og ”out” en ”connect to”-knap, og

desuden en ”sell” knap, der samtidig angiver hvilken pris man kan få for maskinen (50% af købsværdi + 50% af summen af opgraderinger).

Herunder er kataloget over maskiner man kan købe, ordnet i en accordion-menu, der kollapser de sider i kataloget der er ude af fokus. Her er et billede på hver side, der viser den relevante maskine, og 5 knapper med en basissmag på hver. Når man klikker på en af smagsknapperne, får man en maskine af den relevante type frem på fabriksgulvet, hvor farven på maskinen angiver den valgte smag. Maskinen er nu i placerings-mode, og har en forbindelsesstreg til ”out”-området, og en ”build”-knap i kontekstområdet. Man kan nu flytte maskinen hen på en fordelagtig plads, og derefter klikke på ”build”-knappen i kontekst-vinduet, hvorefter man ikke kan rykke den længere. Dernæst skal man klikke på ”connect to” og forbinde maskinen med enten ”in”-området eller den foregående maskine i en kæde, ved at klikke på samme. Et gråt transportbånd forbinder nu de to, og den nyligt placerede maskine fyldes med øl-enheder, og sender dem videre i kæden så hurtigt som hastighedsindstillingen angiver. Når øl-enhederne når ”out”-området sælges de for en pris, der er høj eller lav alt efter hvor tæt den smagskombination maskinerne udgør, rammer i forhold til en ”ideel” opskrift, der genereres tilfældigt hver gang spillet starter. Det er nu spillerens opgave at få et maksimalt udbytte ud af den mængde råvarer, han har til rådighed, og undgå at løbe tør for penge i processen. Man gør dette ved at eksperimentere sig frem ved at købe og sælge maskiner med forskellige smage, og se om de gør produktet mere eller mindre værd. Når man mener man har en vinder, skal man skynde sig at opgradere maskinerne til en højere kvalitet, så man kan nå at få trukket så meget værdi som muligt ud af de råvarer der er tilbage. Man kan bygge flere parallelle kæder af maskiner, for at eksperimentere med smagskombinationer hurtigere, eller for at diversificere sin produktion som en strategi til at få en *nogenlunde* høj profit.

I Beer Factory repræsenterer vi de fem anerkendte smagstyper, *Bitter, Salty, Sour, Sweet* og *Umami*.[32] I vores program repræsenteres smagene af farver, da vi mente det var den bedste måde at oversætte smag til visuelt medie.

Fig. 6 - de fem smagstyper af Lauters

Vi endte med at bruge følgende maskinetyper i vores udgave af brygningsprocessen: *Mill, Masher, Lauter, Boiler, Fermenter* og *Bottler,* som illustreret i nedenstående figur. Disse seks maskiner er naturligvis ikke de eneste elementer, der indgår i den virkelige brygningsproces, men er vores simplificering af virkeligheden, for at gøre spillet spilbart og undgå unødig kompleksitet.

[32] http://en.wikipedia.org/wiki/Taste#Basic_tastes

Fig. 7 - de 6 maskinetyper

Vi har ikke satset på realistiske størrelsesforhold, som man kan se på de temmelig ens størrelser maskiner og de meget overdimensionerede lastbiler i input og output bays. Umiddelbart er dette et levn fra vores erfaringer i RTS spil, hvor størrelsesforhold er noget man meget hurtigt overser og accepterer. Suspension of disbelief spiller hurtigt ind i sådanne tilfælde, hvorfor vi ikke følte at det var nødvendigt at bruge mere realistiske forhold. På samme måde har vi fravalgt de realistiske ventetider, der er i ølbrygning. Gæring tager betydeligt længere tid end maling af råvarerne, men vi valgte at normalisere denne ventetid, for at gøre spillet mere tilgængeligt.

Hver maskine tilføjer en basisværdi i form af en raffinering af råvarerne, og derudover imprægneres øllen med en smag fra hver maskine. Hver maskine har endvidere en vedligeholdelsesudgift og råvaren der sendes fra input til Mill-maskinen koster også et fast beløb. Alt i alt er spillet en visualisering af Porters værdikæde, hvor hver del i kæden tilføjer en værdi til produktet, som dog ikke er absolut, men relativ til en markedspris.

Begrundelse af designvalg

Granulering - hvor høj/lav er detaljegraden i simulationen, hvorfor?

Realismen i et givent spil er i høj grad determineret af, hvilke elementer der kan medieres effektivt igennem mediet, dels igennem de afgrænsende regler, der bestemmer spilinteraktionen, dels hvilke dramatiske virkemidler, der virker effektiv i en interaktiv medieform, og endelig hvor store ressourcer man har til rådighed.

Et bryggeri i virkeligheden ville kunne medieres i et spilmedie ned til en meget lille skala af begivenheder. Der er enzym- og gærprocesser, der ville kunne simuleres og påvirkes med f. eks. Temperatur, men som vi nævnte i syntese-fasen af ideudviklingen, har vi observeret en tendens i management-spil som SimCity og Transport Tycoon, til at spillet bliver mere og mere hektisk efterhånden som det skrider frem, og at stigende succes opnås sammen med stigende arbejdsindsats, hvor spilleren ender med ikke at nyde gameplayet så meget som han kunne, fordi al tiden bruges på panisk at styrte rundt og micromanage en masse små breakdowns i det system man har opbygget. For at undgå denne onde cirkel, så spilleren kunne være frigjort til at nyde de sene faser af spillet, var vi nødt til at afholde os fra at simulere processerne i ølbrygning på et mikroniveau. I det endelige design er bryggeprocessen opdelt i komponenterne Mill, Masher, Lauter, Boiler, Fermenter og Bottler. Disse stemmer overens med virkelige bryggemetoder (med forbehold for specialøl med alternative metoder), men maskinernes indre processer er kun repræsenteret ved deres indbyrdes forskellige hastighed, driftsomkostninger og det associerede billede af maskinen.

Tilsammen udgør kæden af maskiner også en repræsentation af Porters Værdikæde. Her er detaljegraden dog også forsimplet noget, da overbygningen af porters model: management, R&D osv., er reduceret til én altfavnende driftsomkostning per maskine per sekund, som man så må vurdere i forhold til den tilføjede værdi per sekund (de to parametre, skalerer forskelligt, så man kan udnytte economies of scale, både ved at skrue en maskine op på fuld kapacitet, men også hvis en maskine er langsom, ved at indsætte flere parallelle maskiner af en enkelt type som et led i kæden, for at maksimere den overordnede kapacitet).

Afgrænsning - hvilke virkelige elementer er ikke taget med?

Udover skalamæssige begrænsninger, har vi også afgrænset bredden af elementer, der er med i spillets univers. Hele spillet finder således sted inde i en fabrikshal, og vi har stort set abstraheret resten af verden væk – leverandøren af råvarer optræder ikke som en vigtig del af spillet, råvarerne kommer bare pludselig frem i starten af samlebåndet – bagenden af to lastbiler er tegnet ved input og output for at antyde at man som brygger er en del af en større sammenhæng, men spilleren har ingen interaktive muligheder for at påvirke hastigheden, prisen eller kvaliteten af de råvarer, der bliver leveret, eller den pris man kan sælge den producerede øl for. Vi besluttede at gøre det på denne måde for at opretholde spændingen i spillet, ved at benytte "ticking clock"-effekten, som er en klassisk benyttet mekanisme i film til at indikere for seeren at vi kun har kort tid til noget sker. I spil betyder det, at spilleren vil føle sig tidspresset til at lave sine beslutninger.

Markedskræfter

Siden vi ønskede at implementere Porters værdikæde, og i øvrigt lade vores designvalg være determineret så vidt muligt af den virkelige verden, kunne vi ikke nøjes med at have hver enkelt maskine i fabrikken "tilføje værdi" til øllen direkte. Dette ville også gå imod Rollings & Morris' betragtninger om gameplay, nærmere bestemt Dominant Strategy Problem. Hvis der var en optimal opsætning af maskiner, der ville føre til den højeste værditilføjelse, ville spillerne hurtigt ende med bare gentage denne kombination af maskiner, og ikke eksperimentere yderligere. Spillet ville i så fald være "afkodet", og ikke længere nær så underholdende. Vi valgte derfor at simulere varierende markedskræfter, og brugte tid på at udforske forskellige muligheder. Vi endte med at vælge ideen om "den perfekte opskrift", da dette passede godt ind i ideen om den heroiske entreprenør, og den optimistiske stemning vi arbejdede for at skabe i spillet. Da spilleren skulle eksperimentere sig frem imod denne opskrift, var det nødvendigt at holde den usynlig i spil-interfacet, og vi valgte derfor en meget simpel repræsentation af markedet, da et meget dynamisk marked uden synlige clues ville være næsten umuligt (og sandsynligvis frustrerende) for spilleren at gætte mønsteret i. Vi sørgede i den forbindelse også for, i overensstemmelse med Juuls definition af "Coherent World Games", at undgå at implementere arbitrær eller konfliktende fiktion om denne usynlige markedssimulation, der kunne stå i vejen for spillerens egen fantasi.

Tilføjelser af arbitrære elementer til spillet, som ikke modsvarer noget fra virkeligheden

Vi gør visse valgmuligheder i spillet "lige gode", hvor de i virkeligheden ikke ville være det. For at bevare gameplay-værdien i spillet er vi nødt til at afveje de forskellige valgmuligheder, spilleren har, så der ikke er en dominerende "bedste" løsning, som spilleren så vil vælge hver gang. Løsningen på Dominant Strategy Problem har således for vores vedkommende været at afvige fra realismen. Dette er lidt uheldigt, set i forhold til vores ambition om at læne os så vidt muligt op ad virkelige forhold, men i forbindelse med vurderinger af forskellige muligheder for at simulere markedskræfter (som nødvendigvis måtte

determinere, hvilken strategi der ville være den bedste) måtte vi erkende at valget stod imellem en usynlig markedssimulation (der derfor skulle være stærkt forsimplet, for ikke at frustrere spilleren), eller en troværdig markedssimulation, der ville være nødt til at være synlig, og derfor ville ødelægge muligheden for den eksperimenterende dynamik, vi gerne ville opmuntre spilleren til at give sig i kast med; vi valgte den forsimplede fortolkning af markedet, udefra devisen om "rule number one is that your product should be fun"[33].

Modsvar til kapitalismekritiske spil

Vores spil har nogle fællespunkter med et andet flashbaseret browserspil, "The Burger Game". Dette er et såkaldt "activist game", i stil med Gonzalo Frascas politiske spil "September 12th". Man indtager rollen som manager for en multinational restaurantkæde injurierende meget lig McDonalds, og skal administrere de forskellige dele af produktionsprocessen, fra landbrug til restaurant til administration. Hele spillet har en aktivistisk tone, og hver enkelt led i kæden har nogle ondskabsfulde valgmuligheder, såsom at fælde regnskoven for at få plads til kvægmarker, eller at bestikke hygiejnekontrollanten, så restauranten kan slække på hygiejnen, og dermed spare penge, og en masse lignende muligheder. Den væsentlige forskel mellem dette spil og vores ligger i, at vores spils tema samarbejder med den underliggende struktur i spilmediet, og at succeskriterierne, der er defineret i spillets narrativ, svarer til succeskriterierne i spillets regler. Sagt på en anden måde, bliver man i The Burger Game straffet af spillets narrativ hver gang man tager et "fordelagtigt" valg indenfor spillets regler. Man bliver således konstant konfronteret med konsekvenserne for sine egne kyniske beslutninger (siden det er nødvendigt at vælge de mest usympatiske, for at have succes i spillet). Underholdningsværdien er dermed ret lav, medmindre man beholder et ironisk perspektiv mens man spiller, men man kan som spiller ikke lade være med at ønske at score så mange point som muligt, og beherske spillet. Hver gang man oplever en sejr i The Burger Game, er det derfor et følelsesmæssigt nederlag. Fordelen ved vores eget spil er simpelthen at vende denne dynamik på hovedet, og i stedet anvende et tema, der naturligt komplementerer spilmediets egen struktur.

Bemærkninger om spillets udtryk

Illustrationen på spillets startskærm viser både det eksplicitte emne, øl, og subteksten om at hylde entreprenøren – her skal de tegnede art deco-lignende statuer (og ditto skrifttype) vække mindelser om modernismens optimistiske verdenssyn, og tanken om entreprenørens beherskelse af verden.

Teknologiudvælgelse

Undervejs i forløbet, var vi nødt til at overveje hvilken udviklingsplatform vi ville benytte til at udvikle vores spil. Der var flere bestemmende faktorer i denne beslutning, så som brugbarhed, erfaring, medfølgende komponenter og tidsfaktoren. I sidste ende skulle udviklingsplatformen være således, at vi kunne færdiggøre et fornuftigt resultat i løbet af den relativt begrænsede tid vi havde til rådighed, men stadig kunne opføre sig som vi ville have det - dvs. for meget rigiditet i systemet, ville bryde vores vision med spillet og derfor måtte vi have et system som vi kunne bruge ordentligt. Vi er begge optrænet med objektorienteret programmering, hvorfor vores teknologifelt blev begrænset noget. Konkret overvejede vi følgende teknologier:

[33] Rollings & Morris (2004), p. 60

Python/Jython er et høj-abstraktionssprog, som bygger ovenpå C/Java, hvis umiddelbare mål er at dets kode skal være pænt, nydeligt og elegant. Dette betyder, at sproget er relativt simpelt, og da vi ikke kunne umiddelbart finde lette måde at håndtere grafik på den måde vi gerne ville i det, var vi nødt til at forkaste dette. Derudover er vores begges erfaring med sproget afgrænset til et enkelt semesters introducerende undervisning (2,5 år tilbage), hvorfor det ikke var optimalt for vores ønskede brug.

C++ er den de facto standard til spiludvikling på større spil, og på baggrund af det overvejede vi at benytte dette sprog til Beer Factory. Sproget er dog alt for avanceret til dette lille spil vi ville bygge, og der ville være for meget spildtid med at lære sprogets finesser til at vi kunne retfærdiggøre at bruge det i denne opgave.

Java er et højniveau sprog ligesom Python, og er operativsystemagnostisk, hvilket er en stor fordel. Endvidere har vi begge store erfaringer med, da vi har brugt sproget til i et tidligere fag, *programmering og systemudvikling*, og primært har anvendt dette sprog til at lære OO-paradigmet. Desværre er dets swing moduler ikke specielt brugbare til spil, hvorfor vi til sidst forkastede det til fordel for det næste.

Flash er bygget til at afvikles i en web-browser og er efterhånden blevet en standard for web-baserede spil. Flash indeholder sproget ActionScript, hvor vi fokuserede på den nyeste udgave AS 3.0. AS 3.0 ligner på mange punkter Java, og bruger også det objektorienterede paradigme. Flash arbejder derudover meget nemt med grafik.

Vi valgte det officielle Adobe Flash fra, da vi ikke har råd til at købe det alligevel, og de fordele der er med Timeline ikke var noget vi havde behov for. Endvidere fandt vi overbygningsmodulet *Flex* som tilbød diverse UI funktioner, som vi tænkte ville være fornuftige at benytte os af. Vores endelige udviklingsplatform endte således med at blive en kombination af Flex 3.3 og ActionScript 3.0.

Endeligt valg: Flex 3.3

Adobe Flex minimerer animations metaforen som Adobe Flash er designet med, og er derved nemmere at adaptere for traditionelle applikationsprogrammører som os. Miljøet benytter MXML, et XML baseret sprog, til at opsætte grafiske brugerinterfaces, hvilket gør brugen betydeligt nemmere når man har erfaring fra web-udvikling i html/xml. Til interaktion bruges ActionScript 3, som også benyttes i Flash, og er meget lignende Java kode. Flex udviklingskittet inkluderer flere standard UI elementer, knapper, layout containere, tekstbokse osv. på samme måde som HTML, og vi kan benytte disse til vores UI til Beer Factory spillet.

Systemudviklingen

Vores udvikling var primært iterativ og fungerede som en vekselvirkning mellem design og programmering. da vi havde slået os fast på en overordnet ide, byggede vi en simpel prototype, som skulle vise hvad vores spil gik ud på. Denne endte dog med primært at være en "tech demo", da vores første programmerings-omgang primært gik ud på at få Flex til at makke ret og gøre det vi bad den om. Denne udgave er tilgængelig på http://nickithansen.dk/projects/BeerPrototype.swf.

Efter denne prototype havde vi en opfølgende vejledning, hvor vi fokuserede mere på de spilelementer som manglede, som f.eks. point og mere frihed i spillet. Vores efterfølgende designsessions foregik sideløbende med udviklingen, hvilket kunne lade sig gøre fordi vores program var lille nok til at man hurtigt

kunne skifte retning og da den underliggende modellering var relativt fastlagt, hvorfor de basale ting, som næppe ville ændre sig, blev lavet først. Vi valgte ikke nogen fast udviklingsmetode, såsom *Agile* eller *Extreme Programming*, da vores projekt er så lille at det bureaukratiske arbejde ville være spild af arbejdstid som kunne være brugt direkte på projektet i stedet. Vi gik derfor mere pragmatisk til værks og lavede små opgaver hele tiden: "Vis point", "Beregn pris på øl" osv. Disse opgaver betød at de underliggende detaljer skulle på plads for at systemet kunne fungere. "Beregn pris på øl" ville skulle deles op i mindre opgaver, såsom at finde ud af, hvordan ølpriser skulle gemmes, om de skulle tilføjes pr maskine eller genereres til sidst, og ud fra hvad . Tidligere var denne implementeret simplere end den endelige version, da hver maskine bare tilføjede en værdi til hver Beer, hvor den nu har hele smagssystemet implementeret. Her fungerede vores opgave-baserede iterative proces ved, at vi først lavede en ad hoc hurtig løsning, som banede vejen for den mere komplekse, men mere givtige løsning.

Produktion

I dette afsnit vil vi kigge nærmere på den praktiske sammensætning af vores program og klargøre hvordan vi har udviklet programmet til den tilstand det har nu. Vi gennemgår først spillets benyttelse af objekt-orienteringsparadigmet, hvorefter vi ser nærmere på spillets UI, og følgende viser vi hvordan UIet er koblet med den bagvedliggende model. Overordnet har vi så vidt muligt fulgt model-view-controller arkitekturen, og har derved holdt de grafiske og underliggende model elementer adskilt.

Objekt-orientering

Spillet er en modellering af virkeligheden og passer således godt til objekt-orienteringsparadigmet. Vi har disse konkrete klasser i programmet:

Beer, Controller, InputMachine, Machine, Model, OutputMachine

Visual { GameSprite, GraphicsResource, Resource }

De øverste er direkte de der bruges i modelleringen af spilverdenen, mens de nederste bruges til fremvisningen af visningen. Hver øl genereres som sit eget objekt, hvilket vi havde lidt diskussioner om, da det i meget store mængder kan give performance problemer, hvorfor vores midlertidige løsning længe var at ville lave "BeerCrate" i stedet, og blot repræsentere antallet af øl som en multiplier. I sidste ende mente vi at vi kunne kontrollere det ved blot at holde antallet af øl på et fornuftigt niveau, og endvidere er der ikke så stort performance hit som vi frygtede.

View

Main menu

Beer Factory spillet starter med en illustration med en enkelt knap med teksten "Start Game". Dette opnås ved at have "states" i main.mxml.

main.mxml

```
<mx:State name="MainMenu">
        <mx:AddChild relativeTo="{gamePanel}">
                <mx:Canvas height="100%" width="100%" id="titleScreen"
                        backgroundImage="{title_test}"></mx:Canvas>
```

```
        </mx:AddChild>
        <mx:AddChild relativeTo="{titleScreen}" >
        <mx:Button x="515" y="600" height="50" width="150" label="Start Game"
                  id="btnStart" click="startGameClicked(event)"
color="#323232"/>
        </mx:AddChild>
</mx:State>
```

Den ovenstående kode viser, at der tilføjes en Canvas, som er en speciel type i Flex, som kan indeholde andre UI objekter, og kan placere disse præcis på x,y koordinater. Endvidere indeholder Canvas en bitmap baggrund, som vi tilføjer til via nedenstående embedding.

```
main.mxml
[Embed(source = './media/title_test.png')]
[Bindable]
private var title_test:Class;
```

Denne form for implementering af grafik ressourcer er speciel for flex, og betyder at det brugte bitmap bliver gemt direkte i swf filen; den endelige filtype.

Den tilføjede knap sætter, når den bliver klikket på, en event i gang, som bliver håndteret af `startGameClicked` funktionen, som ændrer currentState til "game" og starter systemets controller, hvilket betyder at UI'et ændrer sig til spillets primære UI og spillet bliver sat i gang.

Gamestate

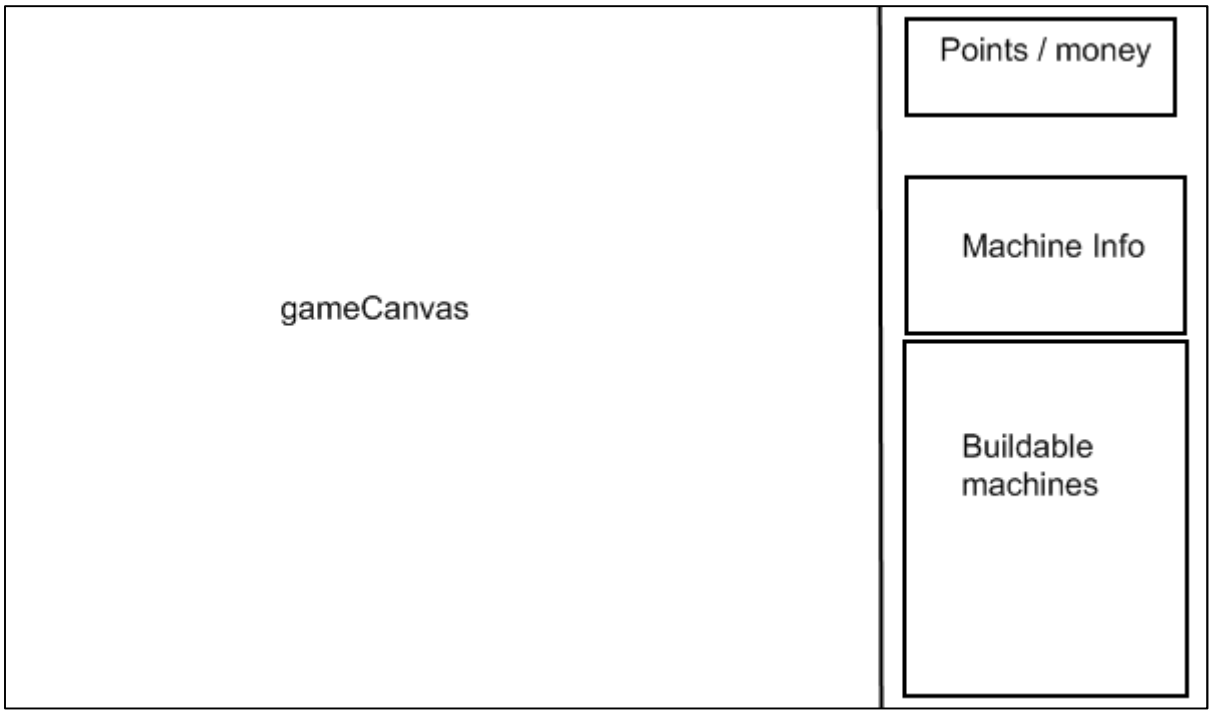


Fig. 8 - Oversigt over UI områder

Konceptet var at vi skulle have en spilplade eller *game world,* hvor vi kunne placere vores maskiner i et tetris-lignende miljø, så man således havde et "mini-spil" inde i spillet, idet at man kun har begrænset plads. Vi lavede denne ved igen at benytte Canvas, således at vi kan placere andre child objekter på denne med absolutte x,y koordinater og flytte dem dynamisk. Vi kalder denne for *gameCanvas*.

Til venstre lavede vi en sidemenu, i stil med strategispil som Command & Conquer. Denne lavede vi ved at bruge indbyggede UI containers fra flex, som opsættes på samme måde som bokse i HTML. Der er således bokse i bokse, der bestemmer hvordan de forskellige UI komponenter skal placeres. Vi benytter HBox (horizontal) og VBox (vertical), som bestemmer hvor dets "children" bliver placeret efter hinanden, og endvidere bruger vi Accordion menuen til at vise de maskiner, der kan bygges. Disse menusystemer er alle baserede på standard Flex elementer, og var en af grundene til vi netop valgte Flex som udviklingsmiljø. Hele UI opsætningen kan ses i mxml filen i kildekoden, der er tilgængelig på den medfølgende cd.

Koblet til vores backend er der endvidere grafiske elementer, som vi skal kunne interagere med. De forskellige brygningsmaskiner har hver en *GameSprite* associeret, som er et gennemsigtigt bitmap element, som kan tegnes ovenpå gameCanvas og håndteres deri. De visuelle backend dele er opsummeret i nedenstående UML klassediagram.

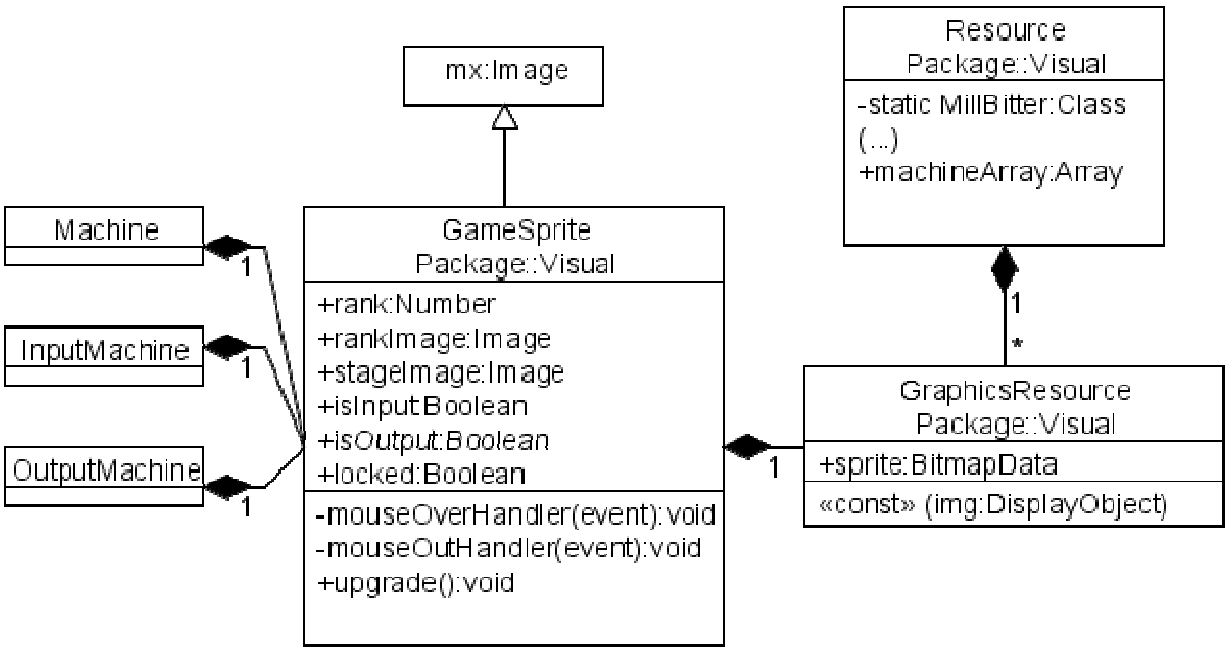


Fig. 9 - UML klassediagram over Visual package

En GameSprite er en specialisering af den indbyggede Image klasse, således at vi kan drage fordel af de indbyggede metoder i denne, og kan bruge gameCanvas.addChild uden konflikter. GameSprite indeholder endvidere en enkelt GraphicsResource, som er vores egen udgave af Image, der fungerer som en "hack", så vi kan få transparente bitmaps i spillet - hvis vi brugte Images direkte blev baggrunden hvid, hvilket ikke var optimalt.

De enkelte maskiner fik vi tegnet af en ekstern grafiker, som ud fra vores anvisninger tegnede repræsentationer af visse ølmaskiner. Fra et programmeringssynspunkt er disse blot bitmaps, hvorfor der ikke er den store forskel fra vores oprindelige simple abstrakte symboler. Det eneste forskel er størrelser, da systemet skulle tage højde for bitmaps med forskellige størrelser, da alle tegningerne ikke er samme højde*bredde forhold. For brugeren er de nye bitmaps dog meget bedre:

Fig. 10 - gammel masher ny masher

Model

Den underliggende model til vores spil, går ud på at man skal sende Beer objekter fra en InputMachine igennem op til 6 forskellige maskiner, som imprægnerer deres smagsegenskaber på objektet og til sidste sender det til en OutputMachine, som sælger ølobjekterne, så spilleren for en højere indtjening end det han bruger på at købe råvarer (de "tomme" Beer objekter) og for vedligeholdelse af maskinerne.

Et Beer objekt indeholder et Number for hver af de fem smage vi har repræsenteret i spillet: bitter, salty, sour, sweet og umami. En maskine går konkret ind og højner denne værdi. F.eks. vil en Sweet Mill tilføje 1 til sweet variablen. Endvidere, vil maskinen ændrer øllets completedStage variabel, som bruges til at udregne øllets endelige pris når det skal sælges.

Hver gang der startes et nye spil genereres der en "perfect flavor", som er en vilkårlig blanding af smagstyper ud fra et budget på mulige antal stages (6).

OutputMachine.as

```
public function generatePerfectFlavor():void
{
        var budget:Number = 6;
        while (budget > 0)
        {
                var taste:Number = Math.floor(Math.random() * 5);
                perfectFlavor[taste] = perfectFlavor[taste] +  1;
                budget--;
        }
}
```

Når en øl er kommet igennem til OutputMachine udregnes en værdi ud fra hvor tæt på den perfekte smag og hvor mange stages den her været igennem.

OutputMachine.as

```
private function findPrice(beer:Beer):Number
{
        var avgQuality:Number = beer.qualitySummed / beer.completedStage;
        var optimalBasePrice:Number = beer.completedStage * 5 + 5;
        var tasteDifferential:Number = 0;
        tasteDifferential += Math.abs(beer.bitter - perfectFlavor[0]);
        tasteDifferential += Math.abs(beer.salty - perfectFlavor[1]);
        tasteDifferential += Math.abs(beer.sour - perfectFlavor[2]);
        tasteDifferential += Math.abs(beer.sweet - perfectFlavor[3]);
```

```
            tasteDifferential += Math.abs(beer.umami - perfectFlavor[4]);

            var price:Number = optimalBasePrice - tasteDifferential;

            price = Math.round(price * avgQuality);
            return price;
}
```

Til at holde styr på spillerens net worth, har vi lavet et Model objekt, således vi kan værdierne logisk kan være adskillelige fra controlleren. Oprindeligt var de gemt som instansvariable direkte i vores controller, men for at undgå at det bliver et *blob* anti-pattern, som samler alle funktioner i sig selv[34], uddeler vi ansvaret til andre objekter i vores program.

Controller

Vores controller klasse bruger singleton mønsteret, da vi på alle tidspunkter kun skal bruge en enkelt controller. Controlleren modtager de events som sendes fra UIet, når der bliver lavet mus-interaktioner med spillet. main.mxml modtager disse events først, og sender dem direkte videre til controller, som så udfører de relevante ændringer.

Da vores primære tilgang til programmering er pragmatisk, er det dog ikke slået fast med syv-tommer søm, at controller-objektet skal håndtere alle events, da visse er ligeså fint styret fra main.mxml klassen, som indeholder de umiddelbare modtagere af events. I disse tilfælde fungerer main som controller for den specifikke funktion.

Refleksion

Vi valgte bevidst et lille browserspil som målsætning, da vi vidste vores tid og kundskaber udi spilproduktion begge var begrænsede. Vi mødte dog alligevel grænsen for begge da det kom til nogle områder af spillet – ambitionen om at udvide perspektivet udover det enkelte fabriksgulv ville f. eks. have krævet for meget arbejde – den nuværende form af spillet ville kunne forbedres meget på mange områder, ikke mindst grafisk, hvor animeret indhold ville gøre en stor forskel, og gå meget godt i spænd med den industrielle kulisse, og browserspillets æstetik. Menuerne kunne ligeledes dekoreres bedre, så det kunne supplere resten af look-and-feel bedre. Gameplaymæssigt ville det være godt hvis maskinerne havde en byggetid, som det tog når man købte en maskine. Dette ville være et boost for realismen, i forhold til den nuværende form hvor maskiner bare bliver sat ind øjeblikkeligt. Udover disse ressourcemæssige begrænsninger, har vi også overvejet de følgende elementer, som kunne være interessante at implementere. Vi vurderer her de mulige konsekvenser.

Vi overvejede at implementere opgraderingen af maskiner med en "X-factor", hvor en opgradering ville fungere som en joker i forbindelse med at ramme den "perfekte opskrift". Jo flere opgraderinger man købte, jo mere sikker var man på at ramme denne opskrift, og jo mindre ville man behøve at købe og sælge maskiner, for at eksperimentere sig frem. Vi vurderede dog, at det ville være en kraftig "Dominant Strategy", som defineret af Rollings & Morris, og derfor ville føre til ensporet adfærd. På den anden side ville adfærden med at købe og sælge maskiner muligvis være frustrerende for spilleren, når motivationen bag spillet nu er at maksimere sit udbytte af en begrænset mængde råvarer, og salget af maskiner

[34] Horstmann (2004), p. 181.

medfører et tab i forhold til deres købsværdi. Vi kunne eventuelt genoptage denne overvejelse i fremtiden, hvis det viste sig at X-factor-ideen medførte færre ulemper.

Det kunne være et interessant supplement til interfacet hvis maskinerne hver var associeret med en lyd, der startede når maskinen startede. Kombinationen af maskiner kunne så modsvares af en behagelig kombination af lydene, for eksempel kunne de passe sammen i en akkord. lydsiden kunne justeres til at stige i frekvens eller takt når man regulerede på maskinerne, og takterne/tonerne kunne svare til hvornår maskinerne passede sammen i arbejdstempo, så man kunne høre sig frem til hvornår takterne passede sammen, eller tonerne var i harmoni, og på den måde have en auditiv ledetråd til at optimere sin værdikæde. Dette ville dog være umuligt at lave i Flash, så det ville nødvendiggøre en ”port” af spillet til et andet udviklingsmiljø, da flash kun kan klare temmelig begrænset håndtering af flere lydkanaler, samt synkronisering af rytmer, etc.

Siden vi har fulgt OOP i vores programmering, ville det ikke være svært at lave flere forskellige inkarnationer af "fabriksgulv"-modellen. det ville således være muligt at have et større spil med flere baner, hvor hver bane udgjorde et led i en større værdikæde, og hvor man skulle forfine et produkt hele vejen igennem systemet. forskellen mellem en mineskakt og et R&D-kontor ville således være overfladisk, og gameplay-metoden ville være nogenlunde den samme. vi kom dog frem til at vi hverken havde tid eller ressourcer til sådan et projekt, da der ville skulle laves unik grafik til hver afskygningen af den underliggende model, og passende differentiering af gameplayet til at det ikke blev *for* ensformigt. Desuden ville medieringen af så forskellige miljøer som mineskakter og designkontorer til et universelt firkantet område med flytbare firkantede elementer være, uanset hvor dekorativ grafikken var, for stor en abstraktion i forhold til det romantiske, virkelighedsnære paradigme, vi havde bundet os til, i form af den kritiske rationalisme.

Priserne på de forskellige elementer i spillet kunne med fordel justeres for at optimere balancen i gameplayet. dette ville bedst opnås i en iterativ proces, hvor gentagne brugertests ville anvise det optimale prisniveau for maskiner, råvarer og opgraderinger. Vi vil gennemføre sådanne brugertests i tidsrummet mellem den skriftlige aflevering og det mundtlige forsvar, og fremlægge eventuelle resultater der.

Vi er endt med et enkelt udseende spil på overfladen, som til gengæld er grundigt baseret på mange forskellige teorier om computerspilmediet, narrativ og den kreative proces. Vores mål har fra starten af processen været at udforske computerspilmediet. Vi synes at vi har formået at kombinere et teoretisk fornuftigt grundlag og implementere det i en produktion på en helstøbt måde. Der er selvfølgelig plads til forbedringer, men vores designvalg har generelt været i god overensstemmelse med det teoretiske fundament.

Litteraturliste

Barthes, Roland (1968), Death of the Author, USA: Aspen #5-6, hentet 6/2/2009 fra http://www.ubu.com/aspen/aspen5and6/threeEssays.html#barthes

Brookshear, J. Glenn (2005): *Computer science – An overview,* 9th edition, Pearson.

Horstmann, Cay (2004). *Object-Oriented Design & Patterns*. USA: John Wiley & Sons.

Kirby, Alan (2006) Death of Postmodernism And Beyond, London: Philososphy Now, hentet 6/2/2009 fra http://www.philosophynow.org/issue58/58kirby.htm

Larman, C. (2006). *Applying UML And Patterns.* Westford: Pearson Education, Inc.

Lye, John (2000), The 'death of the author' as an instance of theory, USA: Brock University, Hentet 6/2/2009 fra http://www.www.brocku.ca/english/courses/4F70/author.php

Neitzel, Britta – Narrativity in Computer Games i: Handbook of Computer Game Studies, USA: MIT Press, ISBN 0-262-18240-8

Porter, Michael E., (1980) "Competitive Advantage: Techniques for Analyzing Industries and Competitors" Qvortrup, Lars (1998) Den Nye Eftermodernitet: Mennesket ud af Centrum, kapitel 6 og 7 i: Det Hyperkomplekse Samfund, København: Gyldendal, ISBN 8-700-45508-3

Rollings, A. & Morris, D. (2004) Game Architecture and Design, Indianapolis:New Riders Publishing

Saussure, Ferdinand de (1916) Course in General Linguistics. Eds. Charles Bally and Albert Sechehaye. Trans. Roy Harris. La Salle, Illinois: Open Court

Turner, Tom (2008) Essays on Cities and Landscapes, USA: Garden And Landscape Books, hentet 6/2/2009 fra http://www.gardenvisit.com/history_theory/library_online_ebooks/architecture_city_as_landscape/after_post_postmodernism

Zimmerman, Eric – Narrative, Interactivity, Play and Games: Four Naughty Concepts in Need of Discipline i: First Person – New Media as Story, Performance, and Game, USA: MIT Press, ISBN 0-262-23232-4