Design af en amatør spiludviklingsproces

ROSKILDE UNIVERSITET

5. SEMESTER, EFTERÅR 2018

VEJLEDER: INGE-LISE SALOMON

FORFATTERE:

Michael Piepgras Neergaard	59729	mipine@ruc.dk
Bertram Pelle Metcalf	59747	bpellem@ruc.dk
Jeppe Stougaard Faber	59774	jsfaber@ruc.dk
Peter Christian Tolsager Nøhr	61002	pctn@ruc.dk
Nils Müllenborn	61462	nimu@ruc.dk

Abstract

In this paper we examine development processes in the videogame industry and attempt to use the information gathered therefrom to structure and plan a hypothetical small-scale game development process while simultaniously attempting to follow such a process. We examine how game development processes are structured and what separates game development from other forms of software development. In terms of modelling, we examine software modelling methods are used in game development and why they are used, as well as how agile development plays into game development processes. A hypothetical process was created for amateur developers, with a team size assuming that ongoing simoultaneous processes were impossible to carry out, and was structured into three separate stages to be carried out in order; pre-production, production and post-production, as inspired by litterature sources. Our examination of industry-litterature, in addition to our own experience with the followed process and the experience gained through designing our own process, lead us to conclude the following:Game development is uniquely split into three separate stages: 1. with the purpose of experimenting with the game concept and design, 2. with the purpose of implementing the game in its minimum-viable stage, and 3. with the purpose of optimizing and improving the game as implemented in the previous stage and get it ready for release. The size of the development team has a drastic influence on the activities in the game development proces since more developers require a higher level of communication and therefore more formalized and divided processes.Modelling in video games is generally meant to communicate design, but the complexity of the process and other possible factors may require the modelling to adopt a more formalized approach.Game development includes frequent re-definition and re-iteration of the design and as a result, an agile approach is almost a necessity to avoid wasting time on re-planning; if the design constantly changes, a plan-driven approach is not a viable solution since every developed plan is unreliable.

Indholdsfortegnelse

1 Introduktion **3**
1.1 Problemformulering . . . 3
1.2 Afgrænsning . . . 3
1.3 Opgavens fremgangsmåde og opbygning . . . 4
1.4 Begrebsdefinitioner . . . 5

2 Spiludvikling **9**
2.1 Agile & Scrum . . . 9
2.2 Udviklingsprocessen . . . 13
2.3 Udviklingsroller . . . 21
2.4 Modellering i spiludvikling . . . 30

3 Beskrivelse af udførte arbejdsproces og erfaringer med denne **34**
3.1 NESI . . . 35
3.2 Machinations . . . 35

4 Forslag til den fulde arbejdsproces **37**
4.1 Pre-Produktion . . . 38
4.2 Produktion . . . 42
4.3 Post-Produktion . . . 44

5 Diskussion **46**

6 Konklusion **48**

7 Perspektivering **50**
7.1 Videre arbejde med projektet . . . 50

Introduktion

Produktion og salg af videospil (i.e spil) har gennem det seneste årti oplevet en voldsom udvikling. Alene i 2017 blev der omsat for over $127 milliarder [Wijman, 2018]. Med store digitale spilbutikker som Steam er det både blevet nemmere at sælge og købe spil. Dette har medført en stor forøgelse i omsætning og antallet af spil produceret. Ser man på det nuværende marked for spil, tyder det på at omsætningen for 2018 bliver større endnu. Dermed er spil en af de største underholdningsindustrier i verdenen. De moderne blockbuster-spil, The Witcher 3[Makuch, 2015] og Metal Gear Solid 5[Lin, 2015] havde et udviklingsbudget på $80 millioner og tog over 3 år at producere. Med så store investeringer i produktioner der varer flere år, er udvikling af spil vanskeligt og risikabelt. Dermed er der opstået et behov for at have velplanlagte og struktureret udviklingsprocesser. Disse processer kan være med til at, blandt andet, mindske forsinkelser, udbedre produktionsfejl inden de bliver store problemer og vedligeholde et velfungerende arbejdssystem. Selvom mange specialister allerede har forsket i udviklingsprocesser og udarbejdet forbedrede processer, er der stadig behov for at fortsætte forskningen. Spil er unikke softwareprogrammer der har forandret sig meget, men som stadig udvikler sig på u-forudsebare måder.

I dette projekt har vi tænkt os at udforske videospilindustrien med fokus på udviklingsprocesserne bag spil. Vi vil anvende fundet teori til at gå i dybden med udviklingsprocessen og anvende den tilfundne erfaring til at skabe en 5-mands optimeret spiludviklingsproces.

1.1 Problemformulering

Hvordan kan der på basis af industristandarder for spiludviklingsforløb, struktureres og planlægges, et spiludviklingsforløb som ville kunne udføres af os.
Herunder:

- Hvilke softwareudviklingsprocesser er unikke for spiludvikling?
- Hvilken indflydelse har størrelsen af udviklingsholdet på spiludviklingsprocessen?
- Hvordan bliver der gjort brug af modellering inde for spiludvikling?
- Hvordan bliver der gjort brug af Agile arbejdsformer i spiludvikling?

1.2 Afgrænsning

Vi har i dette projekt valgt at afgrænse os fra nogle aspekter og områder, der potentielt kunne være interessant at undersøge eller diskutere for vores relevante emne. Først og fremmest har vi besluttet os for ikke at undersøge eller diskutere budgettering. Dette har vi valgt af flere grunde.

Vi mener at det bevæger sig ud i et mere økonomisk aspekt og væk fra projektets fokus. Selvom god finansiering af et spiludviklingsprojekt er en stor drivkraft for at udføre sådan et projekt, vælger vi ikke at inddrage dette fordi vi arbejder med at strukturere produktionen af et spil og ikke motivationen til at lave et. Ydermere kræver budgettering af et spiludviklingsforløb en god forståelse af firmaøkonomi, som vi ikke besidder.

Vi har også valgt at undlade kvalitetsanalyse og -kontrol fra projektforløbet. Dette har vi gjort da vi mener at projektets omfang ligger omkring processen af produktionen af spil og ikke analysering og vurdering af et færdigt spil. Derudover kræver en kvalitetsanalyse også at der udarbejdes en definition af hvad et spil af god kvalitet er, dette er igen ude for projektets omfang og formål.

Til sidst har vi også valgt ikke at inddrage risikoanalyse. Vi har valgt risikoanalyse fra, fordi det ligeledes er hårdt at bedømme hvor risikabelt sådan en produktion er, samt at det ligger ude for projektets omfang, og derfor ville tage fokus væk fra arbejdsprocessen i sig selv.

1.3 Opgavens fremgangsmåde og opbygning

For at læsere af denne rapport opnår den optimale forståelse af projektet, er det oplagt at forklare læseren den overordnede struktur af rapporten. Dertil er det første vi har lavet et begrebsafsnit. Dette har vi gjort da spil, spiludvikling og hele dets tilhørende forbruger-organisme i sig selv benytter termer som ikke altid har den samme betydning som den direkte tilsvarende danske oversættelse, og derfor vil miste sin betydning. Vi vil ikke specificere de termer der anvendes direkte i teksten, men har derimod lavet et afsnit om videospil og de tilhørende begreber.

I den redegørende del af vores rapport gennemgår vi det teoretiske grundlag for projektet. Her har vi et kapitel der indeholder beskrivelser af de teorier, metoder og modeller der anvendes i spiludvikling og som er relevante for vores projekt. Udover dette indeholder denne del også en forklaring af de forskellige roller der eksisterer i industrien.

I den analyserende del af rapporten bruger vi vores empiri fra teori og vores erfaringer til at designe og modellere en hypotetisk spiludviklingsforløb til en mindre gruppe uerfarne udviklere.
Til sidst tager vi et kritisk syn på vores model og vores empiri grundlag, for at identificere eventuelle fejl og mangler. Udover det reflekterer vi over hvordan processen ville se ud fremadrettet.

1.4 Begrebsdefinitioner

Hvad er videospil

Definitionen på det

Spil kan variere fra hinanden som solen og månen, men i stedet forskellige genrer og platformer. De kan fortælle vidt forskellige historier, på lige så forskellige måder. Hvilken definition bør et så bredt og kreativt begreb så indebære? For at besvare dette låner vi definitionen på hvad et videospil er fra Nicolas Esposito. Esposito skriver at definitionen i sin simpleste form består af følgende:

A Short and Simple Definition of What a Videogame Is [Esposito, 2005]

"A videogame is a *game* which we can *play* thanks to an audiovisual apparatus and which can be based on a *story*."

Et spil er et videospil I det at der er en form for interaktion mellem spiller og og et audiovisuelt apparat, ofte med et regelsæt som skaber en konflikt eller forhindring i at spilleren når et mål. Det kan være alt fra at få Mario til at bevæge sig og hoppe til at arrangere de rigtige Tetris-blokke på den rigtige måde. *Play* derimod kan siges at være fri lej indenfor et begrænset omfang som defineret af Zimmerman[Zimmerman, 2004]. Det audiovisuelle apparat kan være ethvert elektronisk system som modtager input som f.eks. bevægelser fra en computermus, og kan håndtere visuelle og auditoriske output, som en computerskærm og højtalere. En historie er ikke altid et krav for et videospil, mange klarer det fint uden, men de er næsten altid centreret omkring spillets mekanikker og leg frem for det modsatte[Esposito, 2005].

Begreber inde for videospil

Inde for videospilsverdenen findes der en række forskellige begreber og udtryk som benyttes til at beskrive diverse elementer inden for videospil. En af de overordnede begreber er konceptet om genrer og et videospils placering i dem. Videospil behøves ikke kun at besidde en enkelt genre, men kan derimod have mange genrer dækkende over sig. Et eksempel på dette ville være et spil med en simpel platformer-genre, hvor spilleren kontrollerer en figur til at hoppe fra platform til platform indtil de når et mål, til at være et tredje-persons rollespil, som er defineret af spillets tredje persons kamerasynsvinkel og de rollespils elementer som spillet benytter, ofte i form af koncepter såsom et gradvist opgraderingssystem af spillerens karakter, oftest set i form af egenskaber og evner.

Et spils gameplay, betegnelsen for spillerens interaktioner med spillet, er opbygget af en række game mechanics. Game mechanics er defineret til, ifølge Sicart[Sicart, 2008], at være metoder for forskellige agenter til at interagere med spillets game status, spillets tilstand. Et eksempel på dette koncept kan ses i den gameplay mechanic kaldet 'Double Jump', som vi har implementeret

i vores platformer spil.

Et andet vigtigt koncept inde for spil er idéen om at et spils gameplay kan være strukturet i levels, baner som spilleren befinder sig i. Dette er et regulær koncept inde for bestemte genre såsom platformer-genren eller FPS-genren, First Person Shooter. Med gameplayet struktureret i levels er det ofte at spilleren starter i en bestemt ende af banen og skal formå at komme videre til en slut-zone som så fører dem videre til næste level.

Der findes selvfølgelig mange forskellige definitioner for hvad spil er, fra den klassiske Game Theory[Neumann, 1944], og mere moderne tilgange indefor ludologi[Jesper, 2003, Salen, 2004]. Spil anses generelt som værende en form for system. Der findes dog uenighed om hvad præcist dette system indeholder, 'Rules of Play'[Salen, 2004], inddeler spil i et system af 4 essentielle elementer.

Example of the **Elements** involved in the **Game System** of **chess**, from Rules of Play[Salen, 2004].

- ☐ The first is *objects*-the parts, elements, or variables within the system. These may be physical or abstract or both, depending on the nature of the system.
- ☐ Second, a system consists of *attributes* -the qualities or properties of the system and its objects.
- ☐ Third a system has *internal relationships* among its objects. This characteristic is a crucial aspect [of systems].
- ☐ Fourth, systems also possess an *environment*. They do not exist in a vacuum but are affected by their surroundings.

- *Objects*: The objects in Chess are the pieces on the board and the board itself.
- *Attributes*: These are the characteristics the rules give these objects, such as the starting positions of each piece and the specific ways each piece can move and capture.
- *Internal Relationships*: Although the attributes determine the possible movements of the pieces, the internal relationships are the actual positions of the pieces on the board. These spatial relationships on the grid determine strategic relationships: one piece might be threatening another one, or protecting an empty square. Some of the pieces might not even be on the board.
- *Environment*: If we are looking just at the formal system of Chess, then the environment for the interaction of the objects is the play of the game itself. Play provides the context for the formal

Salen & Zimmerman's tilgang til et spil system bygger på regler; et spil skak eksisterer kun indenfor et specifikt system, med specifikke regler, og selvom andet kan tænkes[Costikyan, 1994], så anser vi størstedelen af spil som værende regelbaseret.
F.eks. præcisere Salen & Zimmerman deres definition på regler som værende;

- *Constituative* - De abstrakte regler der tilgrundelægger kernen af spillet.
- *Operational* - De regler som dirigere spillerens interaktion med spillet.
- *Implicit* - De uskrevne regler formet af social adfærd.

Et andet synspunkt er det af Aki Järvinen[Järvinen, 2003, Järvinen, 2004], som opstiller en topologi af spilregler og derudfra konkluderer at der er 5 grundlæggende spil elementer i spil syste-

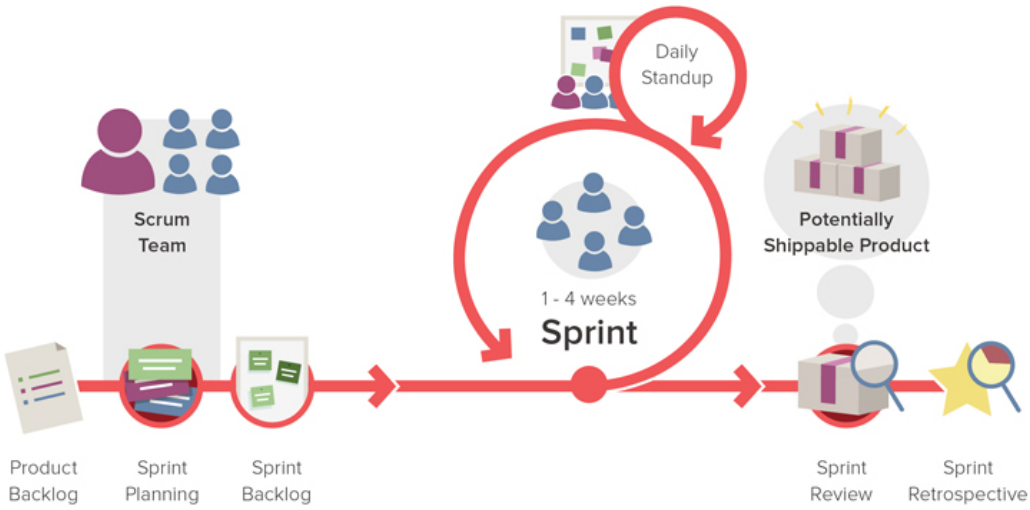
mer: *komponent, procedure, miljø, tema og grænseflade*.

En korrekt definition er altså derfor svær og vi vil derfor blot acceptere spil som systemer, af regler.

Spiludvikling

2.1 Agile & Scrum

[Schwaber and Sutherland, 2017] Scrum beskriver sig selv som "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value."[Schwaber and Sutherland, 2017]
Under sprintet må der ikke laves ændringer der bringer sprintets mål i fare eller sænker kravende til målet.



Figur 2.1: Visualisering af Scrum modelen[Kukhnavets, 2016]

Scrum Events

Sprintet

[Butler, 2018]De Agile sprint er omdrejningspunktet for Scrum development.
Et sprint er en periode på højst en måned som kulminere i en fuldt funktionelt interaktion af det endelige produkt.
Et Scrum sprint indeholder følgende Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective.

Sprint Planning

Sprint Planning er den første fase i ethvert sprint. Det er her at sprintet tilrettelægges igennem et samarbejde med hele Scrum teamet.
I Sprint Planning er der fokus på 2 ting, hvad der kan blive leveret som en del af næste Sprint Increment og hvordan arbejdet der skal resultere i næste increment bliver opnået. Dette bliver gjort ved at udvælge en række funktionaliteter fra Product Backloggen og tilføje dem til Sprint Backloggen.
Desuden bliver der også sat et overordnet mål med sprintet, et Sprint Goal som opnås ved at opfylde funktionaliteterne fra Sprint Backloggen, og målet hjælper teamet med at have en samlet vision for sprintets increment.

Sprint Review

Efter at udviklingsdelen af sprintet og sprintets increment er færdigt, holder Scrum teamet og eventuelle andre steakholders et uformelt møde kaldet Sprint Review.
Målet med dette møde er for udviklerholdet at få feedback på incrementet og at opdatere Sprint Backloggen med hvad der er, og ikke er færdigt, plus tilføjelse af nye eventuelle funktionaliteter der blev opdaget i løbet af sprintet.

Sprint Retrospective

Sprint Retrospective er den sidste fase i Scrum sprint cyklussen. I denne fase selvreflekterer Scrum teamet over hvad der gik godt og mindre godt i løbet af sprintet. Denne refleksion sker inde for både mennesker, relationer, processer og værktøjer, så de kan inkorporere disse erfaringer i efterfølgende sprints.
Dette gøres for at øge produktkvaliteten ved at løbende forbedre arbejdsprocesserne og/eller tilpasse definitionen af "Done".

Scrum Artifacts

Scrum Artifacts eller artefakter er designet til at øge gennemsigtighed for at sikre at alle involverede har den samme forståelse af informationen de indeholder.

Product Backlog

Product Backlog er en sorteret liste af alle krav, udvidelser, og fejlrettelser der skal laves til det

endelige produkt. Denne liste er ikke statisk, da den udvikler sig igennem hele udviklingsprocessen. Alle elementer på Backloggen skal have beskrivelse, rækkefølge, estimat og værdi, desuden er det vigtigt at alle elementerne er af størrelse som kan komme til et stadie af "Done"inde for et enkelt sprint.

Sprint Backlog

Sprint Backloggen er en liste af elementer fra Product Backloggen som er er blevet udvalgt til at blive implementeret i sprintet samt en plan for implementering af elementerne og det satte Sprint Goal.

Ligesom med Product Backloggen er Sprint Backloggen bliver løbende opdateret med hvad der er blevet lavet og elementer kan også tilføjes og fjernes så længe det ikke bryder med Sprint Goal.

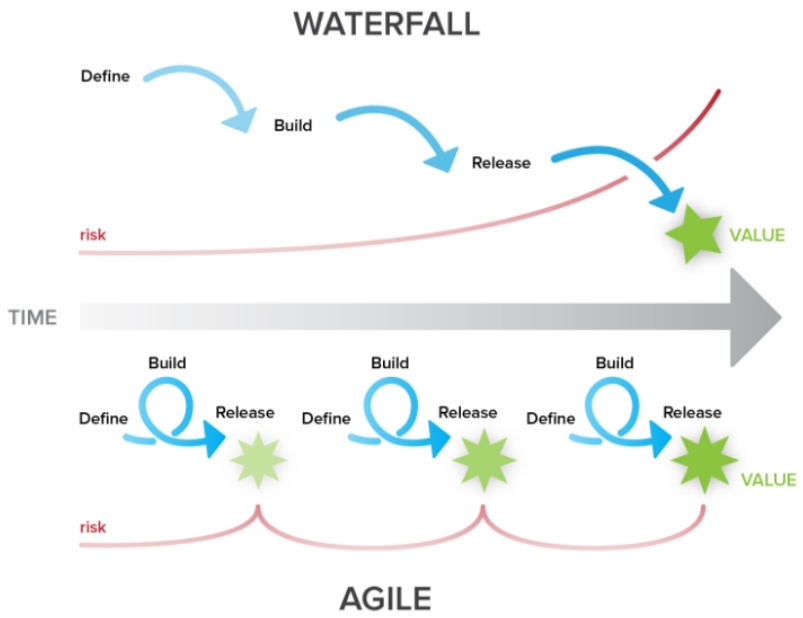
En vigtig egenskab for Sprint Backloggen er at den er ekstern synlig så personer udenfor udviklingsholdet kan følge med i fremgangen.

Increment

Et increment er resultatet af alle Product Backlog elementerne der er blevet implementeret som har tilstanden "Done"både i nuværende og alle tidligere sprint. **Definitionen af "Done"**

Alle medlemmer af Scrum teamet skal have den samme forståelse for hvornår at arbejdet er fuldført. Dette er nødvendigt for ordentligt at kunne bedømme hvor mange funktionaliteter der fra Product Backloggen kan blive fuldført i løbet af et sprint. En klar definition hjælper desuden også med at øge gennemsigtigheden af arbejdsprocessen.

Agil og Vandfalds sammenligning



Figur 2.2: Producering af værdi i Vandfald og Agil arbejdsform[Kukhnavets, 2016]

Den store forskel på Vandfalds og Agile arbejdsformer er at Vandfald er en linear arbejdsproces hvor det Agile er iterativt.
Den lineære proces, hvor alt planlægningen af egenskaber og udviklingsplanen fastlægges i starten af forløbet, kan nemt blive problematisk indenfor spiludvikling. Dette er fordi at interaktionsdesignsprocessen omkring at finde ud af hvad der er sjovt er svært at finde ud før alle dele af systemet bliver samlet. I spiludvikling er en vigtig del af processen at løbende teste og tilpasse spillets elementer og deres interaktioner. Denne proces hvor man laver løbende vurdering af idéers værdi via tests giver sig meget bedre til Agilt-styret udvikling.

Spiltest Teori

Spil ligner meget den grimme ælling i deres første version, hårde at kigge på og endnu hårdere at interagere med. De er nødt til at blive poleret og raffineret før de kan blive den smukke svane som forbrugeren ser i sidste ende, og en af de vigtigste redskaber til at opnå dette resultat ligger i at teste ens spil. Dertil findes der mange forskellige former for testning indenfor spilindustrien, som alle har deres egne pladser i et spils udviklingsproces. For eksempel, sker produktionstestning i et spils alfafase, hvorimod QA, Quality Assurance, først er i betafasen[Luis Levy, 2010].
Indenfor testning er det vigtigt at prioritere hvilke af de fundne fejl der skal løses først, samt hvilke der kan være acceptable at sætte til siden, til senere, eller endda aldrig løses. De bliver ofte sorteret i lav, mellem, høj og kritisk rangering, alt efter hvor nemme de er at løse samt hvor stor en indflydelse fejlene har på spilningen af spillet. [Luis Levy, 2010]
Inde for testning selv eksisterer der forskellige discipliner, som man skal have i mente når spillet testes.

Den første, balance testing, omhandler spillets balance, såsom sværhedgrad og hvor stærkt/svagt forskellige elementer i spillet er i forhold til andre. Det gælder om at gøre spillet fair, alt efter hvor spillets design ligger[Luis Levy, 2010].
Næste disciplin er compatibility testing, hvor testerne undersøger hvorvidt spillet kan køres på forskellige konfigurationer af PCer, konsoler, mobiler og andre enheder. Passer spillets kompabilitet med enhederne som det er designet til? Hvis ikke, så skal det tilpasses indtil det gør. [Luis Levy, 2010]
Tredje disciplin er compliance testing, som ser hvorvidt spillet overholder de design eller tekniske certifikater som der skal overholdes for at spillet kan udgives. Ofte går det ud på at undgå religiøse symboler, at alle copyright eller trademarks er tilstede osv. [Luis Levy, 2010]
Localization testing er den fjerde disciplin, og er ansvarlig for at holde øje med om at oversættelser til forskellige sprog af spillet er i god kvalitet og beholder spillets originale ånd i dets design. Det er et must for at opnå et så bredt et marked som muligt. [Luis Levy, 2010]
Den næste disciplin er nok den vigtigste for et spils succes som underholdningsmedie, og det er playtesting. Playtesting er ansvarlig for at tjekke om et spil er sjovt at spille, at det har en vis ”fun factor”. Det er også primært her at man finder og rapporterer de fejl som er mest væsentlige for spillet. [Luis Levy, 2010]
Sidste disciplin er usability testing. Usability testing går ud på at teste hvor brugbart og nemt det er at inteagere med spillet og benytte sig af de forskellige mekanikker og elementer der er i det. Det er for eksempel med til at sørge for at forskellige ting, så som spillets UI, er intuitivt for spilleren. [Luis Levy, 2010]
Grundet at vi er få udviklere betyder det at vi ikke har ressourcerne nødvendigt for at gå igennem alle disse discipliner hver gang der skal testes forskellige iterationer. Dertil har vi valgt at benytte os af playtesting som den primære testningsmetode for spillet, hvilket burde resultere i at de værste fejl og mangler bliver ordnet.

2.2 Udviklingsprocessen

Dette afsnit vil introducere og redegøre for spiludviklingsprocessen som den bliver præsenteret fra diverse kilder. Afsnittet har basis i diverse tilgængelige teorier om emnet, alle som hver især forsøger at generalisere deres beskrivelser. Derfor er det vigtigt at nævne at der, i industrien, findes så mange forskellige processer, at det er praktisk umuligt at definere en enkelt, samlet beskrivelse for disse. [Chang, 2018]

Spiludviklingsprocessen begynder med at definere spillets indledende koncept og forsøger igennem et produktionsforløb at udvide, raffinere og implementere dette til et endeligt spil. Detaljerne i processen varierer meget fra projekt til projekt alt efter faktorer såsom spillets budget, omfang og type, men processen kan oftest beskrives med en grundlæggende model. [Chandler, 2013]. Generelt set skelner man mellem 3 overordnede faser i produktionsforløbet. Hver fase har unikke mål og leverancer, og overgangen mellem dem kan være flydende. De tre

faser har mange navne, men kategoriseres ofte i litteratur som:

1. *Pre-Produktion*, er den første fase i et produktionsforløbet og har til formål at tilrettelægge spillets fokus og indledende koncept, samt udarbejde et et 'Game Design Document' og en produktionsplan.

2. *Produktion*, er den fase i produktionsforløbet hvor selve spillet udvikles. Formålet her er at producere spillets elementer i deres endelige form, baseret på konceptet som udviklet i pre-produktionen.

3. *Post-Produktion*, er den tredje og sidste fase i produktionsforløbet. Her optimeres og forbedres spillet i dets nuværende form, og klargøres til udgivelse. Fasen involverer også diverse aktiviteter efter udgivelse.

Generelt i spilproduktion findes der mange forskellige tilgange til produktionsfaserne og hermed mange forskellige aktiviteter som indgår i dem. Dette afsnit vil forsøge at redegøre for de mest centrale aktiviteter i hver fase og hvordan de hjælper med at udarbejde fasernes endelige leverancer.

Pre-produktion

Pre-produktion er den første fase i spiludviklingsforløbet og er baseret på målene om at få spillets grundlæggende koncept og retning fastlagt, samt planlægge og budgettere spillets fortsatte udvikling i produktionsfasen. Pre-produktionens varighed er afhængig af mange forskellige faktorer, men en generel tommelfingerregel er, at fasen tager alt mellem 10-25% af den totale udviklingstid.[Chandler, 2013]

Pre-produktion er oftest udført af et mindre, centralt team af udviklere, designere og andre specialister, som eventuelt kan udvides inden begyndelsen af produktions-fasen.

Konceptualisering

Spiludviklingsprocessen baseres altid på et spilkoncept. Det indledende spilkoncept er oftest bestemt inden pre-produktionen går i gang og kan være alt fra overordnede idéer om spillets temaer, til detaljerede redegørelser for game mechanics som ville være centrale for spillet. Uanset formen er det en tidlig milepæl i pre-produktionen at udvide og detaljere det indledende koncept til et stadie hvor spillets produktion kan styres efter det. Spilkonceptet tjener til formål, at fokusere teamet og kommunikere spillets endelige vision. Af denne grund vil spilkonceptet altid være en del af pre-produktionens endelige leverancer.[Chandler, 2013]

Det skal siges at spilkonceptet ikke kun udvikler sig i pre-produktionsfasen, men derimod ændres løbende gennem hele processen; Pre-produktionen tilrettelægger kun det grundlæggende

spilkoncept.

Det grundlæggende spilkoncept kan indeholde overvejelser om ting såsom spillets temaer, historie, genre, spillets centrale game mechanics og andre elementer som vil få spillet til at stå ud blandt konkurrerende spil. Hovedformålet med prototyping i pre-produktionen er at få defineret disse elementer tilstrækkeligt og at få lavet en plan over deres implementering. [Chandler, 2013]

Valg af målgruppe

Spil er underholdning. Mennesker bruger underholdning til at underholde sig selv, men ingen person synes lige godt om al underholdning. Adams[Adams, 2013] hævder at spildesignere ofte laver den fejl at ignorere deres målgruppe, og baserer deres design på hvad de selv synes er underholdende. Han mener at alle design-beslutninger bør tages på baggrund af en målgruppe, og at konceptualiseringen derfor må inddrage definitionen af en sådan målgruppe.

Kravsspecifikation

Først når spillets konceptualisering er ovre, og det grundlæggende koncept er fuldt udviklet, kan man begynde at definere spillets kravsspecifikation.
Kravsspecifikationen beskriver samlingen af de centrale illustrations-, design- og programmeringsmæssige komponenter som vil indgå i spillet, samt spillets afgrænsning og de grundlæggende tekniske- og designspecifikationer. Disse komponenter er alle udarbejdet på basis af det grundlæggende koncept. En af måderne at udarbejde kravsspecifikationen er sætte team-medlemmerne sammen og internt brainstorme og rangere forslag i "need-to-haves", "want-to-haves"og "nice-to-haves". [Chandler, 2013]

Når komponent-sættet er defineret, kan projektet planlægges med milepæle og løbende leverancer for hvert komponent. Leverancer her definerer diverse spilelementer såsom illustrationsmateriale, implementerede game mechanics, egenskaber af internt-udviklet software, etc. På denne måde forsøger man at skabe en plan over det fortsatte udviklingsforløb.

Valg af udviklingsværktøjer

Udviklingen af et spil kræver samarbejde mellem personer med forskellige kompetencer. Ikke alle medlemmer af dit team har tilstrækkelig kompetence til at forstå og producere kode. Af denne grund, benytter spiludviklingsforløb ofte en type software kaldet '*Game Engines*'. Game Engines har til formål at abstrahere implementeringen af dit spil fra koden selv, så andre teammedlemmer end programmører kan arbejde på spillet. Som andet software, kan Game Engines enten være tilpasset eller standard software. Tilpassede Game Engines er tidskrævende og dyre at lave, så oftest er brugen af disse begrænset til større produktioner. Mindre produktioner benytter oftest standard Game Engines, men udvikler udvidelser til disse hvis de har specifikke teknologikrav som softwaren ikke tilbyder.

Imens komponent-sættet defineres, undersøges tilgængelige teknologier som skal kunne opfylde de definerede krav.[Chandler, 2013]
Teknologier i pre-produktion og produktion kan afvige meget fra hinanden. Eksempelvis kan prototyper i begyndelsen af pre-produktionen udvikles med en standard Game Engine og krav til en eventuel tilpasset løsning kan 'opdages' løbende under prototyping. Den egentlige tilpassede teknologi vil så udvikles sideløbende i pre-produktionen på basis af de opdagede krav og først tages i brug senere i pre-produktionen og fremover.

Prototyping

Pre-produktionen er præget af en meget eksperimentel udviklings- og designproces i forbindelse med konceptualiseringen. Derfor gør pre-produktionen oftest brug af hurtigt-udviklede prototyper[Cerny, 2002], løbende i processen, til at undersøge de konceptualiserede komponenter:
Som komponenter bliver brainstormet i løbet af konceptualiseringen, er det ofte en god idé at afprøve dem for at vurdere udfaldet af deres implementation. Prototyper hjælper med at se komponenter i kontekst af hinanden og gør det lettere at vurdere hvordan spillets forskellige elementer fungerer sammen[Cerny, 2002].

Fullerton[Fullerton, 2014] beskriver prototyping som en en essentiel del af et godt designforløb, specielt fordi de tillader designere og udviklere at se, hvor realistiske komponenter er at implementere. Generelt set skal prototyper være fuldt interaktive, hvor de mest grundlæggende interaktioner skal indgå, men tidlige prototyper burde kun forsøge at vurdere få komponenter på samme tid. Ifølge Fullerton er der i de tidligere prototyper ingen grund til at implementere de mere krævende komponenter såsom fuldendte illustrationer og lyddesign.
Cerny[Cerny, 2002] beskriver en anden tilgang til prototyping i pre-produktionsfasen. Han foreslår at enhver prototype burde udvikles præcist som et fuldkomment spil, men bare på en mindre skala; Alle delelementer, som indtil videre er definerede, burde indgå i deres fuldendte form i hver prototype, men omfanget af prototypen burde indskrænkes til en lille delmængde af det færdige spil. Dette, mener Cerny, tjener pre-produktionen idet de giver et meget mere realistisk billede af hvordan komponenterne spiller sammen, og derved øger kvaliteten af pre-produktionen som helhed. Derudover foreslår Cerny at prototyper bliver lavet designmæssigt fra bunden i hver iteration under pre-produktionen. Dette forhindrer dårlige vaner og antagelser fra at gå igen i hver prototype og tillader nye tilgangsmåder til allerede løste problemer. Selvom disse metoder er meget tidskrævende, mener Cerny at de kan betale sig for at undgå problemer i den kommende produktionsfase.

Leverance: Game Design Document & Macro Design

"*A design document is a bible from which the producer preaches the goal, through which the designers champion their ideas, and from which the artists and programmers get their instructions*

and express their expertise."[Ryan, 1999]
Spiludvikling er en proces med mange kreative elementer som oftest kan misforstås. Mange af disse beslutninger kan oftest tolkes på forskellige måder af forskellige personer og disse beslutninger kan oftest glemmes senere i udviklingsforløbet. Spiludvikling er samtidig en proces som involverer mange parter; Større spiludviklingsforløb kan let komme til at råde over flere hundrede personer, som alle skal arbejde mod det samme mål. Af denne grund udarbejder man løbende et såkaldt Game Design Document.

Et Game Design Document(GDD) er et stykke dokumentation som udvikles og benyttes under spillets produktion til at kommunikere spillets design til resten af teamet. GDD'et burde starte ud med at indeholde de centrale designmæssige beslutninger som bliver taget undervejs i konceptualiseringen. Eksempelvis kunne et GDD indeholde informationer om spillets fastsatte målgruppe, beskrivelser af spillets centrale features, tematiske overvejelser, etc.[Ryan, 1999].
Forskellige spilprojekter kræver forskellige GDD'er, og der er stor fluktuation i helt præcist, hvilke elementer som indgår i dem og hvor detaljeret de er beskrevet.[Adams, 2013]
Cerny [Cerny, 2002] mener at alt for detaljerede GDD'er er hæmmende for spiludviklingsprocessen; Eftersom spillets design forventes at ændre sig i løbet af produktionsfasen på basis af de erfaringer som man oplever i den, giver det ingen mening at gå i detaljerne før man har haft de erfaringer.

Cerny [Cerny, 2002] foreslår en alternativ måde at se GDD'et på, som han kalder et Macro Design. Et Macro Design GDD indeholder kun de mest overordnede beslutninger om spillets design, eks: Hvad er de centrale interaktioner som spilleren skal kunne udføre, hvordan er spillet delt op i sektioner, etc. Detaljerne af alle disse beslutninger er udeladt, så produktionen har mulighed for at definere dem igen, når de kommer til at skulle implementere det i produktionen.

Leverance: Produktionsplan

Produktionsplanen er hvor alle logistiske informationer bliver trukket sammen. Teamets producer går i spidsen for at forberede ting såsom spillets budget, tidsplan, deadlines og mandskabskrav på basis af de erfaringer som pre-produktionsteamet har gjort sig indtil videre. Når produktionsplanen er udarbejdet, bliver den gennemgået af teamet som skal sikre sig at de kan udføre planen i produktionsfasen. Hvis ikke, kan det være at planen o,g eventuelt kravsspecifikationen skal ændres. [Chandler, 2013]

Preproduktionen afsluttes

Preproduktionen er en meget vigtig del af et spiludviklingsforløb. Konceptualiseringen kan være en fase som meget hurtigt går overbords, og som resultat kan spillet blive praktisk talt umuligt at producere med de tilgængelige ressourcer. I sidste ende handler pre-produktionen om at lægge fundamentet for spillet, og at være realistisk omkring spillets produktionsplan og design. Det er her man får en solid forståelse for, det produkt man vil producere, hvilken type mennesker som man vil imødekomme, og hvilke typer oplevelser man kan lave til dem.
Prototyping har givet et overblik over hvilke teknologier som ville skulle til for at implementere

spillet og prototyperne gør det muligt at genere interesse i mulige investorer idet man kan vise en konkret delmængde af spillet.
Produktionsplanen har givet dig en indledende slagplan for hvordan produktionen skal sparkes i gang og Game Design Dokumentet har givet dig et overblik over hvad du gerne vil lave.
Et spil kan aflyses i alle stadier af udviklingsforløbet, men en af fordelene ved at have pre-produktion som et separat forløb er at man har tid til at vurdere om spillet skal aflyses. Derfor er slutningen af pre-produktion i større projekter, præget af risiko- og markedsanalyser som undersøger om spillet har en chance for at være profitabelt når det udgives.

Produktion

Beskrivelse af fasen

Efter pre-produktionsfasen er afsluttet arbejdes der, i produktions-fasen, videre med det ny lavede skelet af hvordan spillet skal se ud. Det er her hvor det intensive arbejde begynder i fuldt sving, med større hold af producere, designere, kunstnere og programmører på banen.[IGN, 2006] Selve arbejdet i produktionsfasen foregår ofte i flere cyklusser hvori de forskellige hold kommer med nye og gradvist bedre iterationer af spillet og dets elementer. Disse cyklusser er derfor ofte opdelt i de respektive roller som de forskellige medarbejdere har, såsom Design Produktion Cyklussen og Kunst Produktion Cyklussen, så holdene kan arbejde parallel. Kommunikation og samarbejde er nøgleordene.[Chandler, 2013]

Cyklussernes gennemgang

Cyklusserne i sig selv kan variere i deres arbejdsmetoder fra hold til hold og studie til studie, men den generelle idé består af at gentage disse tre underfaser igen og igen, nemlig design, udvikling og testning. Det er en essentiel del af at opnå et produkt som kan nå de specifikationer og krav der er sat i pre-produktionsfasen.

Holdenes arbejde

Producerne er ansvarlige for at koordinere produktionens forskellige roller, at de alle har den samme forståelse for hvad de arbejder efter/på, samt at alle følger arbejdets skema.[IGN, 2006]
Designerne er derimod ansvarlige for at designe elementerne fra design dokumentet bliver sat ind korrekt, samt at komme med nye designløsninger når der opstår enten fejl eller mangler i det nuværende design. Et eksempel på dette kunne være at implementere en game mechanic som var ansvarlig for at putte spilleren tilbage indenfor det besluttede spilleområde i tilfælde af at de finder en vej ud af banen.[IGN, 2006]

Kunstnere arbejder samtidig på at få lavet animationer samt artwork, ofte i animationsprogrammer såsom 3D Studio Max. De modellerer spillets omgivelser, objekter og karakterer samt de grafiske elementer af spillets brugergrænseflade. I visse tilfælde bliver der koordineret med et motion-capture team i samarbejde med at lave animationer for karakter modeller.[IGN, 2006]

Programmører skal også i gang med at kode spillets biblioteker, spillets forskellige mekanikker, spillets AI, samt eventuelt spillets game engine, alt efter om det bliver lavet eksternt med en tredjeparts game engine eller internt med en som studiet selv har udviklet. [IGN, 2006]

Undervejs i denne proces er produktionsholdet med til at optimere de forskellige aspekter af spillet, og sørge for at det sammen kan køre på det hardware som spillet er designet til. Dette inkluderer blandt andet at vælge om der skal reduceres polygoner, ændres effekter, eller sletning af elementer som forhindrer spillet i at opnå dets tekniske mål, såsom 30 eller 60 frames per sekund, hvor heftig på CPU'en spillets AI skal være.[IGN, 2006]

Samtidig med det, skal spillets forskellige prototyper og builds også testes for fejl og mangler af QA. Det er et kritisk element at sørge for at spillet fungerer og kan nå de mål der er sat for det. Oven i det, kan playtesting også være en ressource for spillets designere idet at testernes feedback kan hjælpe med at inspirere dem til at finde alternative- eller nye løsninger i spillets design. [Chandler, 2013]

Produktionsoversigt og Ledelsen

Under produktionsfasen er det vigtigt at der gradvist bliver tjekket om projektet følger den udviklingsplan som blev fastlagt i pre-produktionsfasen. Begrundelsen for dette ligger i at der ofte er en deadline sat for hvornår spillet skal udgives fra ledelsen, hvilket kan betyde at der enten skal sættes flere ressourcer til rådighed eller at der skæres forskellige game mechanics fra, som der ellers var planlagt at have med. Den værste konsekvens der kunne ske ville være at spillet bliver annulleret hvis det er for langt bag ud i forhold til udviklingsplanen, så ressourcerne kan benyttes til andre projekter som er mere lovende.[Chandler, 2013]

Den anvendte metode for at undersøge er ofte i form af en projektanmeldelse. Projektanmeldelsen tillader de relevante parter, ledelsen, projektlederne, samt udgiveren at kunne få et overblik over projektets mål, og hvor i arbejdsprocessen det befinder sig. De undersøger hvorvidt projektet er bagud i forhold til planen, om udviklingsholdene er stødt ind i vejspærringer undervejs, anskue hvilken risiko der er i at fortsætte udviklingen af projektet samt hvorvidt de skal opdatere dokumentationen, såsom status rapporter og udviklingsplaner. [Chandler, 2013]

Produktionen færdiggøres

Når projektet endelig har opfyldt de kriterier der var stillet, er det på tide at gennemteste og ordne så mange af de fejl der kan findes inden spillet skal udgives. Need-to-haves og helst want-to-haves er på plads, og der kan anskues om hvilke nice-to-haves der enten kan arbejdes på i de sidste uger af produktion eller eventuelt tilsidesættes til post-produktion.

Post-produktion

Post-produktionen starter

Nu er projektet ved at nå mållinjen og en ende er til at se ude i horisonten. Produktionen er ovre og der er få uger tilbage inden den deadline hvor spillet bliver udgivet nås. Alpha'en er klar, koden er fråset, spillet bliver gennemtestet synder og sammen, fejl bliver ordnet og marketing går i fuld sving med at promovere spillet til offentligheden. Vi befinder os nu i post-produktionsfasen.

Vores definition på Post-Produktion

Selve definitionen på hvad post-produktionsfasen består af kan variere meget fra studie til studie og person til person. Chandler anser for eksempelvis at post-produktionsfasen består kun af den del af arbejdsprocessen som foregår efter spillet er udgivet og på hylderne [Chandler, 2013], hvorimod Fullerton har en mere løs definition hvor produktionsfasen slutter før QA og vedligeholdelse [Fullerton, 2014]. Derfor har vi valgt at definerer produktion til at slutte ved Alpha, og at post-produktion så består af de resterende elementer af udviklingen.

Post-Produktion i praksis

I post-produktionsfasen lægges der vægt på at polere og vokse spillet indtil det skinner. Meget lignende produktionsfasen bliver der på samme vis testet for fejl og mangler, men siden alt koden er fråset bliver der ikke længere langt vægt på at udvikle nye features. QA sætter sig sammen med spildesignerne og programmørerne for at udarbejde fejlene og få de 70% af spillets kvalitet på plads. [Fullerton, 2014]

Undervejs i processen bliver der ofte holdt hvad der kaldes "triage" møder, hvor udviklingsholdet sætter sig sammen, ser på kodens status og prioriterer fejlene alt efter hvor alvorlige de er. Programmørerne kører databaserne igennem systematisk, og samtidig finder ud af hvilke af fejlene der kræver input fra de andre hold, såsom de visuelle designere eller endda den juridiske afdeling. Når der endelig ikke er første prioritets fejl tilbage er spillet anset for at være i beta, hvorefter målet så er at rette fejl og features indtil spillet er klar til at blive sendt ud til fabrikkerne som en Gold Master. [Fullerton, 2014]

Vedligeholdelse og Post-Produktion

Efter spillet så bliver udgivet er udviklingen anset for at være hovedsageligt færdig, hvorefter de fleste ressourcer bliver sat til side til andre projekter. Det kan dog variere på basis af hvor mange fejl og mangler det endelige produkt har, såvel som de planer der er lavet med hensyn til fortsat udvikling på spillet. Dette er ofte i form af patches, opdateringer og DLC. [Fullerton, 2014]

2.3 Udviklingsroller

Inden for spiludvikling findes der et utal af roller som, hver især, står for et eller flere af de mange delelementer der findes i spil. Dette afsnit vil ikke indeholde en komplet gennemgang af hver rolle, og dens tilhørende funktion, men snarere en klargørelse af hvad der sker i de meste generelle afdelinger af en spiludviklingsprocess.
Beskrivelsen af disse roller laves ud fra den erfaret viden[Chandler, 2013] og et 'optimalt' synspunkt, da der selvfølgelig altid vil forekomme utallige små variationer, ved hver spiludvikling, når der arbejdes med unikke koncepter og holdsammensætninger hver gang et nyt spil skal produceres.

Ledelsen

Ledelses afsnittet skrives med henblik på de personer, som er relevante for det øverste og vigtigste niveau af planlægningen af et spils udvikling og generelle koncept.
Ledelsen i en spiludvikling består af ofte kaldet 'Lead Game' -Designer, -Developer, -Artist osv. Under en spiludvikling er det ledelsen der traffer de meste essentielle valg, for spillets endelige sammensætning, ledelsen er ofte en samling af relativt få personer, som dækker over alle de fagligheder, som det specifikke spil der udvikles, kræver. Ledelsen består af de bedst kvalificeret personer som har erfaring med alle de specifikke spil elementer, som der skal bruges.

Et hvert spil starter med en spil-ide, som beskrevet i 'pre-produktions' afsnittet.
Når en spil-ide skal bringes til live, er det indbyrdes op til ledelsen, at få en fælles forståelse for det endelige mål, således at de kan dele sig ud i flere forskellige afdelinger og lede mange andre personer mod et fælles mål.

Ledelsens formål er simpelt, men som det er med mange ting er det lettere sagt end gjort.

Ledelsen danner afdelinger af personer med erfaringer, som er relevante indenfor den specifikke afdeling, de skal arbejde i.
Disse afdelinger har hver sin funktion og dækker tilfælles over alt hvad der skal til for at bringe spil-idéen til live.

Det er ofte sådan at afdelingerne arbejder sideløbende, men det kan absolut forekomme at afdelinger starter før eller efter, andre afdelinger.

Spil Artist

Indenfor spiludvikling dækker begrebet Game Artist over mange roller. Artisterne sætter blandet andet det visuelle og lydlige præg på et hvert spil.
Koncept artister er ofte de første artister som går i gang med at arbejde.

Koncept Artist

Koncept artister har til opgave at starte den visuelle proces ved at at håndtegne 'essensen' af spilidéen. Grunden til dette gøres er for at bygge et fundament, hvor alle andre der er involveret i spiludviklingen, kan få en fornemmelse for hvad der sigtes efter, samt for at fange problemer med konceptet så tidligt som overhovedet muligt i processen.

Koncept artister står gerne for to forskellige aspekter af det visuelle, det karakteristiske og det miljømæssige.

Figur 2.3: Eksempel på 2D karakter koncept kunst af Liza Ivushkina

Figur 2.4: Eksempel på 3D miljø koncept kunst af Francis Tneh

Disse to koncepter danner to forskellige indtryk som, igen, er med til at danne en forståelse for, og gi inspiration til, resten af afdelingerne.

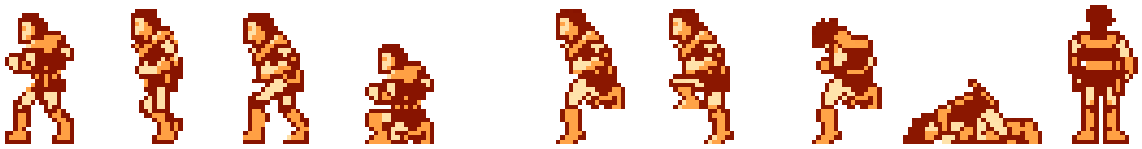
Animators 2D & 3D

2D

Når et koncept er fastlagt går animators i gang med at overføre de statiske billeder, og deres inspiration fra koncept artisterne, til den digitale verden.
Animationen af et spil kan foregå på mange måder og er fuldkommen afhængigt af hvilket slags spil der sigtes efter. F.eks. er det en stor beslutning om spillet skal foregå i 3D, 2D, eller en blanding derimellem.

Der findes mange forskellige måder at producere både 2D og 3D.

Produktionsprocessen af 2D animation er ofte 'unik' for hver individuel animator, da produktionen af 2D ikke kræver mere end en blyant (som faktisk er en spil stil), men ofte benyttes et utal af programmer, som ofte kan modificeres for at tilpasse sig brugeren så godt som muligt. Disse programmer er gerne strømlinet for hurtigt at kunne producere mange 'sprites' af hvad der ønskes animeret for til sidst at sammensætte et 'spritesheet'.

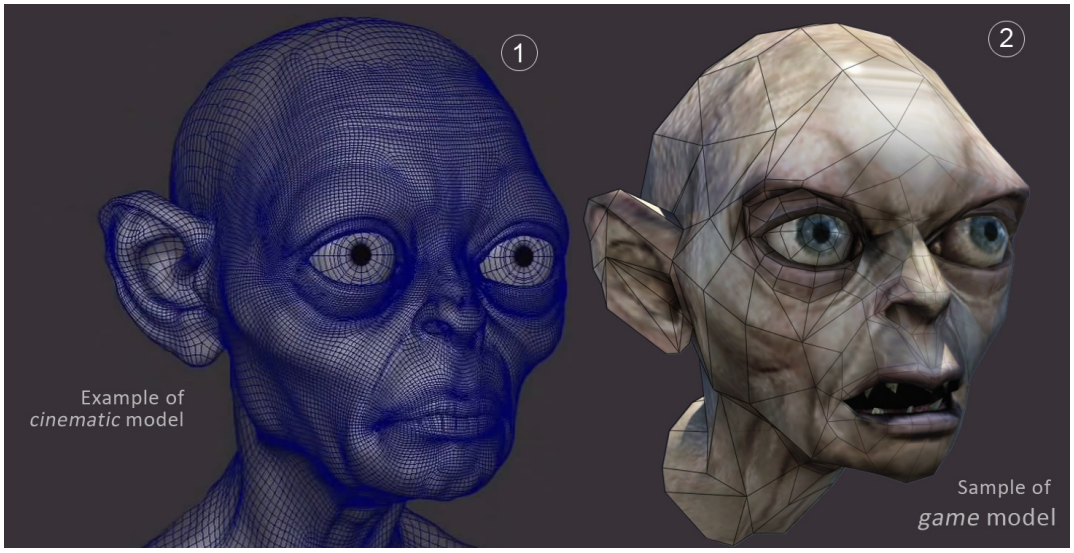


Figur 2.5: delvist Spritesheet af Simon Belmont, fra Castlevania (1986). Disse sprites er (fra venstre) at gå, dukke sig, hoppe, dø og 'gå ind af dør'.
[Castlevania, 1986]

Et spritesheet som vist i 2.6 bruges derefter til at animere ved at skifte mellem de individuelle sprites i et passende tempo, for at give illusionen af bevægelse.

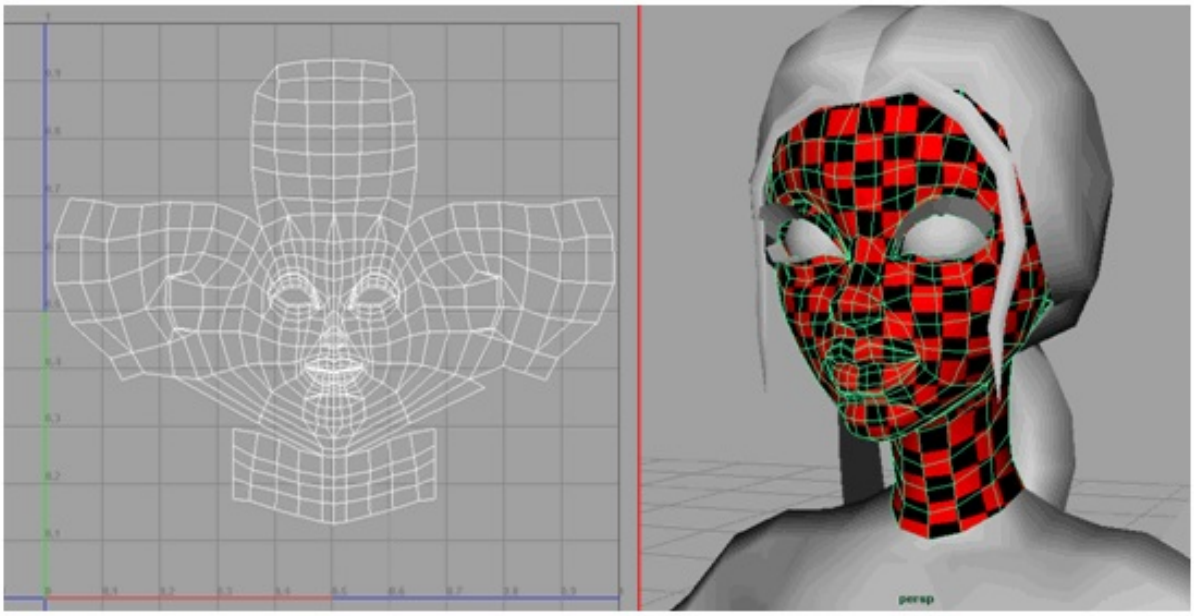
3D
Alt 3D animation sker med software, der findes mange specialiseret programmer som kan animere alt hvad man kan forestille sig til.

Produktionen af 3D animation foregår generelt ved at en animator udarbejder en 'mesh' (ofte ud fra koncept kunst eller egne sketches), som er en sammensætning af polygoner, disse polygoner kan tage formen af hvad som helst og er med til at skabe entiteter man kan placere i sin spilverden.



Figur 2.6: Eksempel på forskellen mellem en 'cinematic' og 'game' mesh af karakteren Gollum fra Ringenes Herre. Bemærk at meshen til højre allerede har en tekstur.
[**?**]

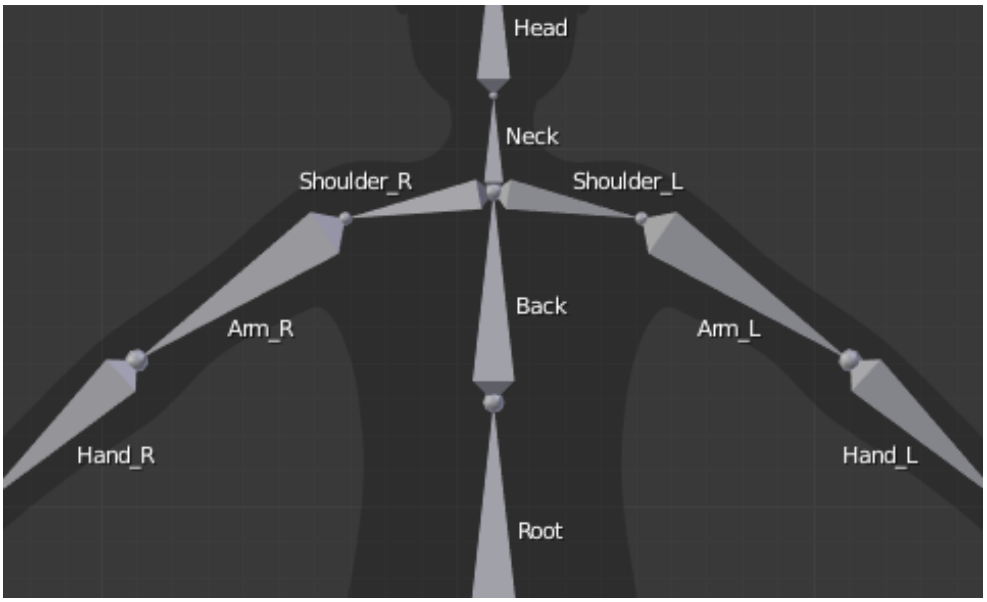
Efter at en mesh er produceret udføres 'uv-mapping' på meshen, det er en teknik, hvor man tilføjer to yderligere akser separat fra meshen, som bruges til at definere formen af de teksturer som man ønsker i et 2D miljø, se 2.8



Figur 2.7: Eksempel på uv-mapping af et ansigt.

[Intel, 2011]

Mesh'en giver dimensionerne og formen af hvad man animere, uv-mapping giver overfladen grafiske detaljer, men for rent faktisk at animere er det nødvendigt at 'rigge' den, det vil sige at man tilføjer et 'skelet' til en model, som binder 'knoglerne' af dette skelet til overfladen af meshen, som giver en mulighed at animere de bevægelser man ønsker.

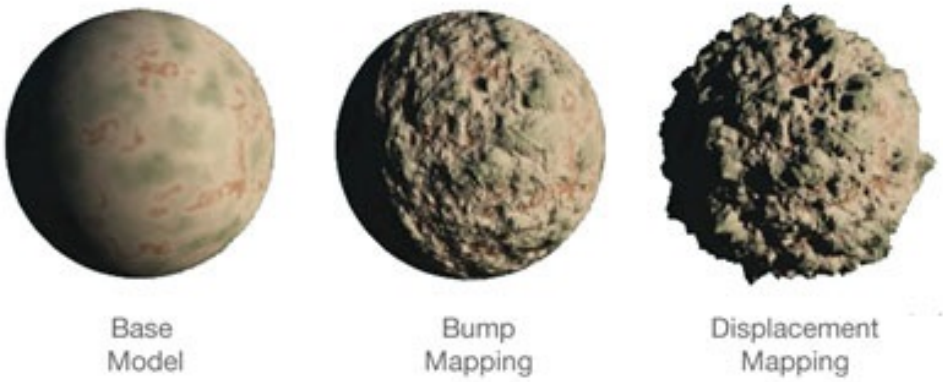


Figur 2.8: Eksempel på rigging af en vilkårlig menneske-formet mesh. Skulle man 'hive' i knoglerne af denne animation ville animationen bevæge sig tilsvarende.

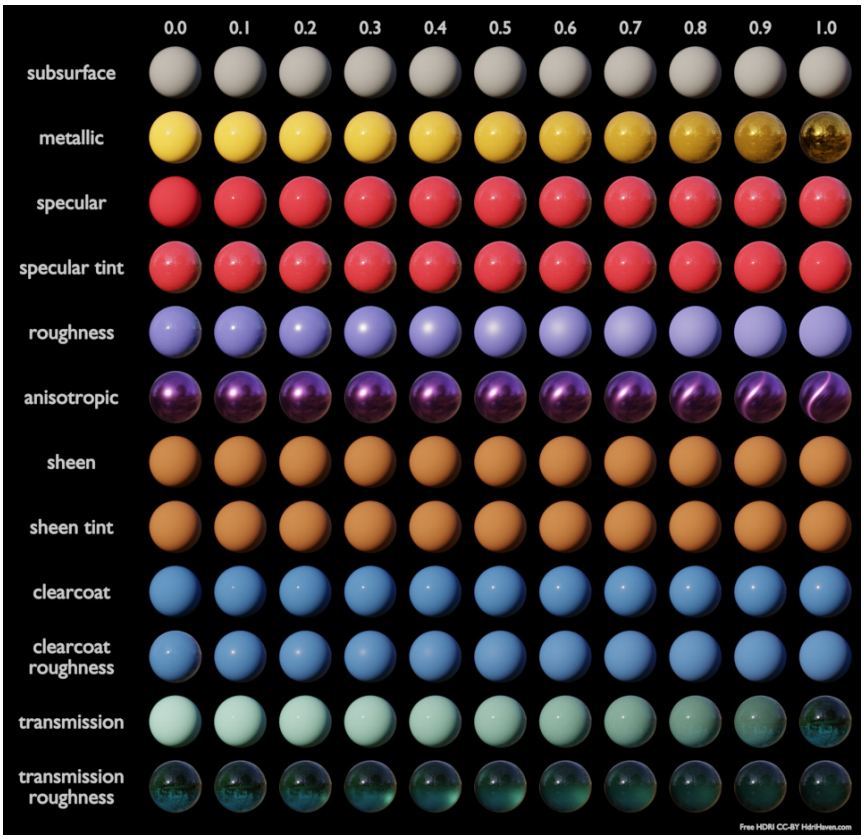
[Blender, 2018a]

Disse teknikker er de meste centrale for at lave en simpel animation. Der findes mange yderligere ting som man kan tilføje til sin animation, men vi vil undlade de fleste og blot illustrere nogle få

metoder man kan vælge at benytte.



Figur 2.9: Bump mapping og displacement mapping er to teknikker til at bringe dybde til ens materialer [Forum, 2013]



Figur 2.10: 'Shaders'. Tilføjes af specielle effekter til materialer, for at give en model en specifik materiale-lignende effekt, så som glas, metal, plastik, etc.[Blender, 2018b].

Som det ses kan en animator benytte mange metoder til at efterligne visuelle effekter som vi selv oplever i 'virkeligheden'.

Lyd

Ligesom at detaljen for det grafik kan gøre en stor forskel for atmosfæren i et spil, er lyd ligeså meget med til at danne en stemning.
(Næsten) Alle spil der bliver lavet nu til dags benytter lyd på en eller anden måde, om det er enorme symfoni orkester eller enkelt mands produktioner, så er de til at finde i spilverden.

Vi vil ikke lave en komplet gennemgang af lyd, musik og det generelt lydlige indenfor spil, da det er en kunst i sig selv at forstå hvad der gør lyd i spil 'god'.
Dog vil vi nævne nogle af de metoder som benyttes ofte indenfor spilindustrien for at give et særligt lydlige indtryk:

1. Positionel imitation af 3D lyd, der giver lyd en 'retning', så giver man selv med kun 2 lydkilder (f.eks. et sæt admindelig hovedtelefoner), kan give illusionen af at en lyd kommer 'fra etagen under en' eller 'rundt om hjørnet'.

2. 3D lyd, modsat ovennævnte kan man hvis man har 3D-lyd højtalere eller hovedtelefoner, og et spil der er designet omkring denne feature, afspille lyd på en sådan måde at det giver livagtig manipulation af lyd.

3. Modulær musik, der flydende skifter stil udfra hvad der sker in-game.

Specialiteter

Der findes mange niece metoder til at opnå noget af det ovenfor nævnte arbejde, vi vil ikke fokusere på dem, men blot nævne dem for at pointere at de findes samt at der er mange af dem som er brugbare f.eks.:

1. I stedet for at lave manuel 'rigging' af et en model, vil man i stedet bruge 'motion-capturing' for at overføre bevægelsen fra en skuespiller til et digital rig. Dette kan være med til at give meget mere livagtig bevægelse.

2. I stedet for manuel uv-mapping, kan der benyttes specialiseret værkertøjer til det, som oversætter et tryk på en 3D model, til en direkte placering af et materiale på en 2D overflade. F.eks. benytter programmet ZBrush, en 'spray-paint' lignede arbejdestil, hvor man belægger sine modeller i materialer ved at 'sprøjte det på'.

3. Det er fornyligt blevet muligt for den gennemsnitlige forbruger at købe hardware, som er i stand til udføre såkaldt 'real life ray-tracing' så hurtigt at det kan benytte i spil, det betyder at man i stedet for at manuelt håndtere skygger og lys, blot i stedet placerer en lyskilder i sin spilverden som man behager og lader 'lys strålerne' danne skyggerne, som derefter danner dybden og detaljen i et utal af ting.

4. 3D-scanning. Hvor man i stedet for at bruge tid på at lave formen af et mesh 'i hånden', lader en maskine skanne formen at et fysisk objekt, som enten allerede eksistere, eller at man skulpturere noget ikke eksisterende og skanner det efterfølgende.

Designere

Designer er en bred term, og indenfor spiludvikling er det ikke anderledes, der findes mange design roller inden for alle felter af spiludvikling. De mest almindelig er:

- *Design director*, er ansvarlig for det overordnet design og den kreative retningen for alt der produceres og designes til spillet.
- *Game designer* er en meget overordnet betegnelse for en designer som arbejder i 'alle' afdelinger af spiludvikling.
- *Level designer*, sætter hver en blomst og bygning i ens spilverden og designer (men ikke implementere) specifikke spil-events. De bygger den verden spillet foregår i.
- *Content designer*, designer materiale til specifikke ting i ens spil. F.eks. vil de lave historien, NPC dialogen, teksten til bogen osv. De vil også være tæt involveret i optagelsen og implementationen af lydmateriale, som f.eks. stemmer.
- *UI designer*, sørger for at designet af brugergrænsefladen.
- *System designer*, er ansvarlig for et eller flere spilelementer, f.eks. kan de være ansvarlige for at det følelse 'rigtigt' at køre en racerbil, eller alt spillets menu system er flydende og intuitivt.
- *Teknisk designer*, er delvist en designer og programmør, de sidder ofte på hegnet mellem design- og implementeringsafdelingerne, for at give en flydende proces fra hvad designer vil, til hvordan programmøren skal implementere designet.

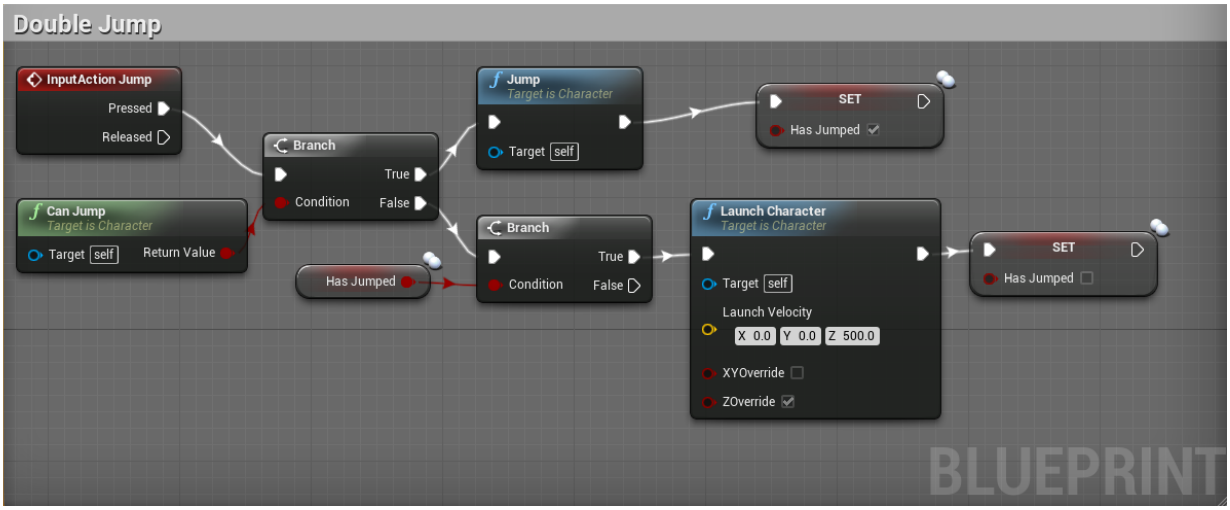
Programmører

Før at designerne og kan bringe artisternes arbejde til live, er det nødvendigt at programmørerne implementere kode som er i stand til at håndtere alle spilelementer som er nødvendige for det specifikke spil, deres roller kan være mere eller mindre essentielle og opdeles gerne ud fra hvad de udvikler.
Det er kun fantasien der sætter grænserne når det kommer til valg af miljø man vil programmere sit spil i, men med det sagt er det selvfølgelig foretrukket at udvikle hurtigt og effektivt, med etableret procedurer, fremfor at genskabe alting fra bunden gentagende gange.
Inden for spiludvikling skelner man gerne mellem 'niveauer' i programmeringssprog, disse niveauer skal ikke sammenlignes med de niveauer der omtales i hardwarestruktur. Det er også ofte nødvendigt at koordinere flere sideløbende programmeringssprog, f.eks. benytter man gerne et

lav-niveau programmerings sprog til at håndtere grafik, hvor i mod man (måske) hellere vil designe sin brugergrænseflade i et højere niveau sprog.

- *"Lav niveau"* - F.eks. programmeringssproget C. Benyttes til at udvikle de mest essentielle og beregningsmæssigt intensive processor, f.eks. grafiskarbejde. Alt kan programmeres i et lavere niveau sprog, men det er en enorm arbejdsbyrde.
- *"Mid niveau"* - F.eks. programmeringssproget C++. Bruges gerne til udvikling af værktøjer, som f.eks. Unreal Editor og Unity, men kan også benyttes til at programmere alle tænkelige spilelementer, dog bruges de oftest på de meste essentielle ting, f.eks. at håndtere bevægelse og kollision, og undlades til udvikling af ting som håndteres i et højere niveau sprog.
- *"Høj niveau"* - F.eks. programmeringssproget JavaScript. Benyttes f.eks. til at lave en grænseflade, håndtere simple overlægning, eller at scripte spil begivenheder.
- *"Visuelt"(Ultra højt niveau)* - F.eks. Unreal eller Unity 'blueprint' programmering, hvor man visuelt sammensætter modeller der oversættes til kode som en editor vil kunne forstå, se 2.11.

Med dette sagt er det et frit valg hvad man vælger at udvikle i. Men generelt er det værd at overveje om den ydeevne og kontrol man får ud af at udvikle i et lavere niveau sprog er det værd i forhold til tempoet man kan udvikle i, i et højere niveau sprog.
Især bør det fremhæves hvor langt væk fra reel programmering man kommer hvis man benytter et såkaldt 'blueprint' programmerings sprog, som nu er implementeret i både Unity og Unreal. Disse blueprint 2.11 sprog kræver minimal programmeringserfaring og oversættes så effektivt at alt på nær de mest krævende spil kan håndteres.



Figur 2.11: Simpelt eksempel på game featuren 'Double Jump' implementeret i Unreal.

QA & Support

Quality Assurance er en afdeling som er mere eller mindre essentiel varierende ud fra omfanget af ens spil projekt. QA har generelt til opgave at udføre tests på og omkring et specifikt spil som er under udvikling. Rollerne indebærer:

- *Spil tester*, har til opgave at spille et spil 'i stykker', decideret finde så mange fejl og mangler som overhovedet muligt og dokumentere dem, på sådan en måde, at problemer kan udbedres i andre afdelinger.
- *Teknisk tester*, udfører gerne systematisk fejlfindingen på et mere generelt plan end spil testeren og er gerne i tæt kontakt med supporten.

Support er hvad der ligger i ordet support; det omkring liggende og støttende, som f.eks. tilhørende faciliteter; spil specifikt forum, hjemmeside, salg, promotion osv. Support er igen ikke en nødvendighed, men det er værd at overveje at man på et eller andet tidspunkt på traffe en ikke spil-specifik beslut og håndtere den.

2.4 Modellering i spiludvikling

Proces Model

Som det fremgår af forgående afsnit er proces model benyttede i alle 3 faser til at modelle diverse processer som eksisterer i spil. I Sommerville står der:

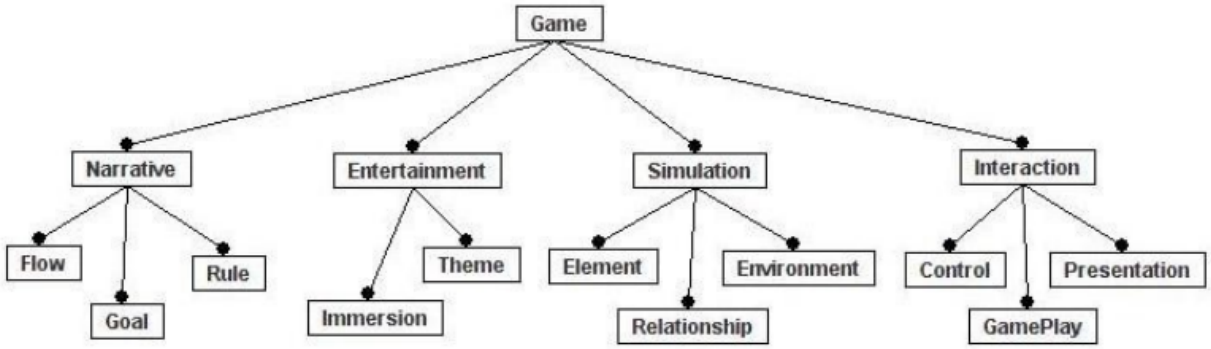
> Ian Sommerville 'Software Engineering'[Sommerville, 1982].
>
> *Process model*
> An abstract representation of a process. Process models may be developed from various perspectives and can show the activities involved in a process, the artifacts used in the process, constraints that apply to the process, and the roles of the people enacting the process.

Inden for spiludvikling er proces modeller vigtige, da de indgår i alle spiludviklings forløb[Aleema, 2011, O'Hagan, 2014]. De holder et meget højt abstraktionsniveau, er hurtige at lave og de kan hurtigt ændres, hvilket betyder at en proces model for et givent spilprojekt ikke nødvendigvis minder om et andet et af slagsen, alt efter hvilke karakteristikker processerne har. Samt dette reflekterer den flydende processer som spil glimrende.

NESI Feature Model

En feature model er en model over et eller flere feature diagrammer. Disse feature diagrammer opstilles hierarkisk, som et træ, der indeholder features som det specifikke system, der modelleres, indeholder.

Mere specifikt benyttes NESI feature modeller[Sarinho, 2014], NESI modeller kommer af sammentrækningen af ordene *Narrative, Entertainment, Simulation* og *Interaction*. Modellen er især brugbar inden for spiludvikling da den giver et nemmere overblik over alle tænkelige spil features. Dette kommer i kraft af en simplificering af de systemer hvor spil foregår, nemlig at kategorisere alle spil elementer under de 4 før nævnte karakteristika.



Figur 2.12: Eksempel på et NESI Feature Diagram

Som det ses i figur 2.12 er et NESI feature diagram forholdsvist simpelt, i det at der blot undergrupperes features under en af de specifikke NESI karakteristika.

Machinations

I løbet af vores undersøgelse af hvilke former for modellering der benyttes indenfor spiludvikling opdagede vi Machinations, som er et modelleringsframework.
Dette framework er på nuværende tidspunkt under en lukket beta, som vi har heldigvis har fået adgang til.

Machinations er et Turing-komplet modelleringsværktøj som er i stand til at simulere modelleringer, samtidig med at de bygges. Spil er generelt set alle sammen resurse baseret og er derfor oplagt at benytte et værktøj som Machinations.

Machinations er under udvikling og der findes ikke særlig meget dokumentation, men vi har på nuværende tidspunkt haft adgang til det længe nok, til at kunne danne os et billede af hvordan det fungere.

Machinations promovere sig selv som den 'bedste' måde at modellere spil på da man kan præsentere flere forskellige modelleringsmetoder samtidigt, samt simulere ens modeller omgående.

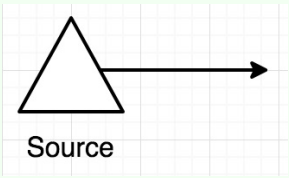
På nuværende tidspunkt er det svært at bedømme om Machinations er bedre end andre modellerings metoder, det er stadig fejlfyldt og hele arbejdsprocessen, i sig selv, kan være ganske

frustrerende. Med det sagt, så er Machinations lovende, det er forholdsvist intuitivt og i løbet af kort tid kan man forstå det meste basale og begynde at lave interaktive modeller. Nedenfor at vist og illustreret de mest fundamentale nodes af Machinations.

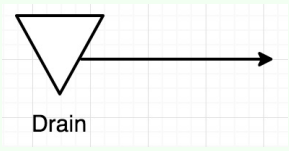
Nodes types in Machinations [Machinations, 2018].

- *Pools*

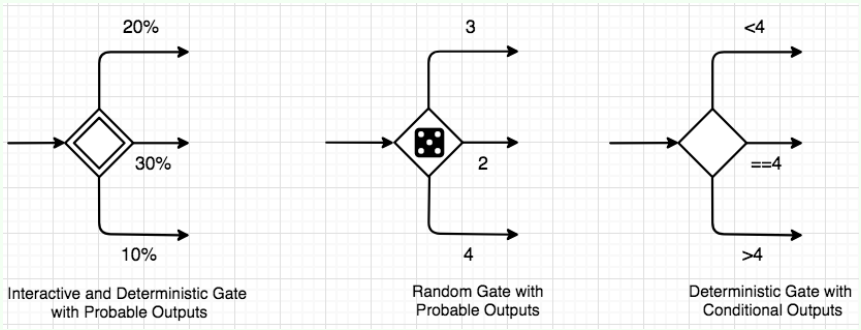
- *Sources*



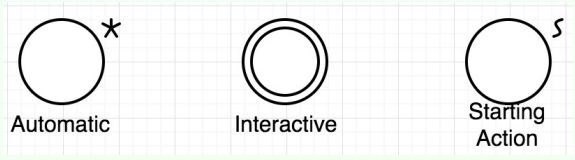
- *Drains*



- *Gates*



- *Modes*



Machinations bygger på ideen om spil som resursebaserede systemer.
Disse ovenfor nævnte noder; pools, drains, converters og pools danner basen for Machinations.

Pools er de meste centrale nodes i Machinations, de har til funktion at holde på mængder. F.eks. vil en spiller i et spil, blive repræsenteret af en pool som indeholder spillerens liv, penge, ammunition og lignende.

Sources er kilden hvor ressourcerne kan udspringe fra.

Drains afleder ressourcerne.

Gates, basale logiske gates, som er i stand til at lave deterministisk og tilfældige udfald.

Yderligere kan visse nodes sættes til forskellige *modes*, som f.eks. gør en pool interaktiv, så man kan aktivere den imens der simuleres og derved manipulere med ens opstilling.

Disse enkelte noder kan opbygge meget komplekse systemer. Vi har eksperimenteret lidt med Machinations i vores procesforløb.

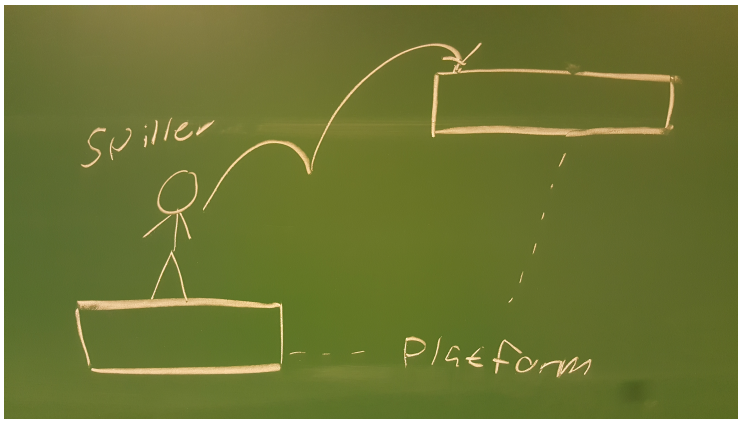
Beskrivelse af udførte arbejdsproces og erfaringer med denne

Dette afsnit vil redegøre for vores udførte arbejdsproces, herunder den overordnede struktur af forløbet og hvilke central valg vi tog.

Processen startede ud med en række møder dedikeret til planlægning og udvælgelse af spillets koncept. Her blev genre, Game Engine og vores definition af minimumsproduktet specificeret, herunder en indledende kravsspecifikation. Dette indledende arbejde gav os et godt overblik over hvordan spillet skulle være, og det tillod os at designe spillets prototype med samme indgangsvinkel.

Da dette var på plads, kunne udvikling af prototypen begyndes. Processen startede med en lille periode dedikeret til at lære den udvalgte Game Engine at kende, samt planlægning af den første prototype. Prototypens grundlæggende form blev implementeret i den valgte Game Engine, og derefter udvidet iterativt i sprints. Vores sprints blev sat til at være 2 uger lange, med sprintplanlægning i begyndelsen af hvert sprint. Daglige Scrum onlinemøder blev holdt dagligt hvor gruppemedlemmerne orienterede hinanden om deres arbejde og eventuelle spørgsmål kunne stilles undervejs. Sprintplanlægningen bestod af en fælles design-session hvor spillets features blev overvejet og opgaver blev uddelt til individuelle gruppemedlemmer. Spillets design blev diskuteret i uformelle, interne samtaler, og ændringer blev inkorporeret dynamisk af enkelte gruppemedlemmer uden for formelle møder.

Ingen dokumentation blev udviklet undervejs i processen og modellering blev kun benyttet under gruppemøderne til at kommunikere designet i gruppen. Ingen formel modellering blev benyttet undervejs i processen.



Figur 3.1: Eksempel på uformel model brugt under gruppemøder. Her er et 'dobbelt-hop' modelleret, hvor spilleren kan hoppe midt i luften for at kommer højere op.

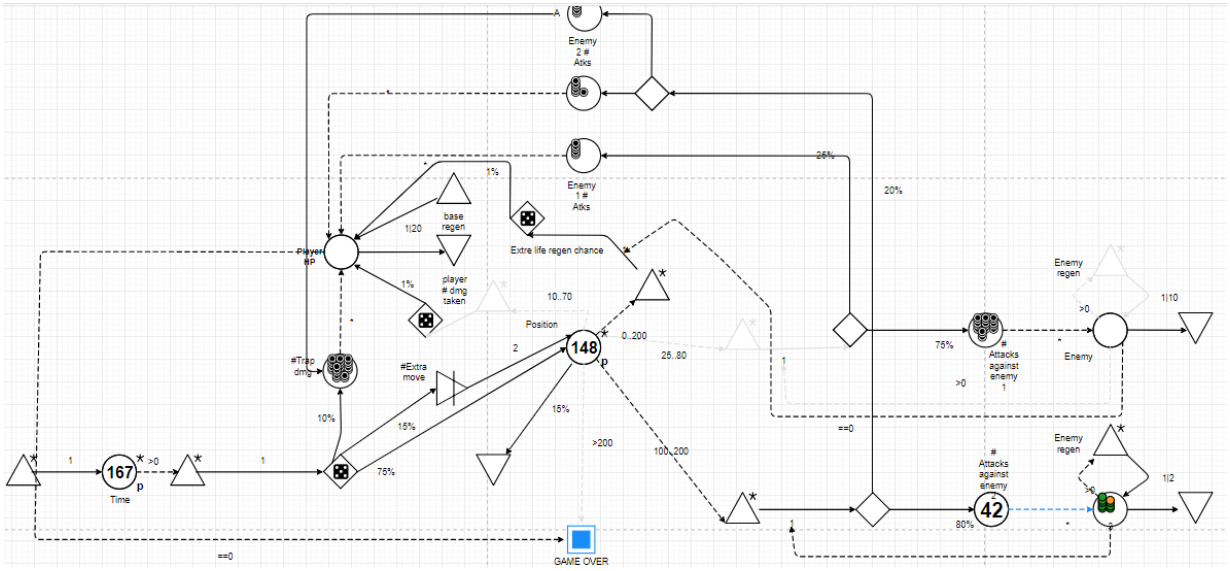
Vi valgte Unity som vores Game Engine, da denne er generelt anset som en brugervenlig og fleksibel Game Engine som fungerede på både Windows og MacOS styresystemerne. Prototypen blev udviklet med pre-fabrikede assets for at simplificere udviklingen, så vi kunne lægge vores kræfter i spillets design. Spillets design blev hovedsagligt baseret på genre-klassisker features, med enkelte originale features indblandet.

3.1 NESI

3.2 Machinations

Det bør gøres klart at vi først fik adgang til Machinations meget sent i vores projektforløb. Vi har derfor ikke haft nok tid til at danne os et realistisk billede over hvor brugbart Machinations er under en reel spiludviklings proces.
Machinations er på et lukkede beta stadie hvor der stadig findes mange fejl og mangler. Vi vil undlade at kommentere på disse og blot klargøre at det ikke virker helt korrekt.

Grundet den sene introduktion har vi ikke brugt Machinations til nogen form for reel modellering under vores projektforløb, men for at danne os et billede af hvad det kan bruges til har vi alligevel prøvet at modellere en simpel opstilling af et metroidvania spil. Opstillingen ser ud som vist i figure 3.2



Figur 3.2: Overblik over vores implementation af en metroidvania i Machinations.

Diagrammet er som det ses, svært at få et overblik over. Især i det at modellen i sig selv er 'interaktiv' og ændrer sin form som den simuleres, er det svært at pointere præcist hvad der foregår igennem modellen, vi henviser til **BILAG** hvor der er en større model.

Denne Machination starter i nederste venstre hjørne. Det hele starter med en *source* som producerer ressourcer i form af 'tid' eller 'ticks'.
Hvert tick kan relateres til en handling fra spilleren. Denne handling sendes gennem en *random gate* som skal illustrere forskellige spiller udfald; spilleren har 10%, 15% og 75% chance for enten at, træde i en fælde, bevæge sig dobbelt, eller blot bevæge sig.
Efter denne del af modellen er der to veje, som ressourcerne kan tage. Enten går der ressourcer til en fælde-'pool som trigger 'spiller'-poolen til at smide en resurse, dette er det tilsvarende til at miste 1 'liv' i en spil verden.
Modsat kan ressourcer fortsætte mod centrum af diagrammet, som er en pool, vi har valgt at kalde 'Position'. Der bliver gradvist sendt flere ressourcer til denne center pool. Dette skal repræsentere at spilleren bevæger sig igennem den modelleret spil verden.
Samtidig med at der bliver fyldt flere ressourcer i positions poolen, bliver der åbnet op for den første ekstra 'vej' som spilleren kan tage. På denne vej findes den første fjende i banen. Fjendes overvindes gennem at spiller noden og fjende noden mister liv indtil at en af dem 'dør'. Skulle det ske at spilleren vinder over fjenden, så vil der først blive tilføjet en permanent livsgenererings til spilleren, som har x antal skridt tilbage før at vejen for den anden fjende bliver åbnet.
Denne fjende er betydeligt sværere at besejre, efter 10 simulationer så vi den kun vinde en enkelt gang og det var rent held.

Det er en lidt svær proces at forestille sig, da den der meget visuel.

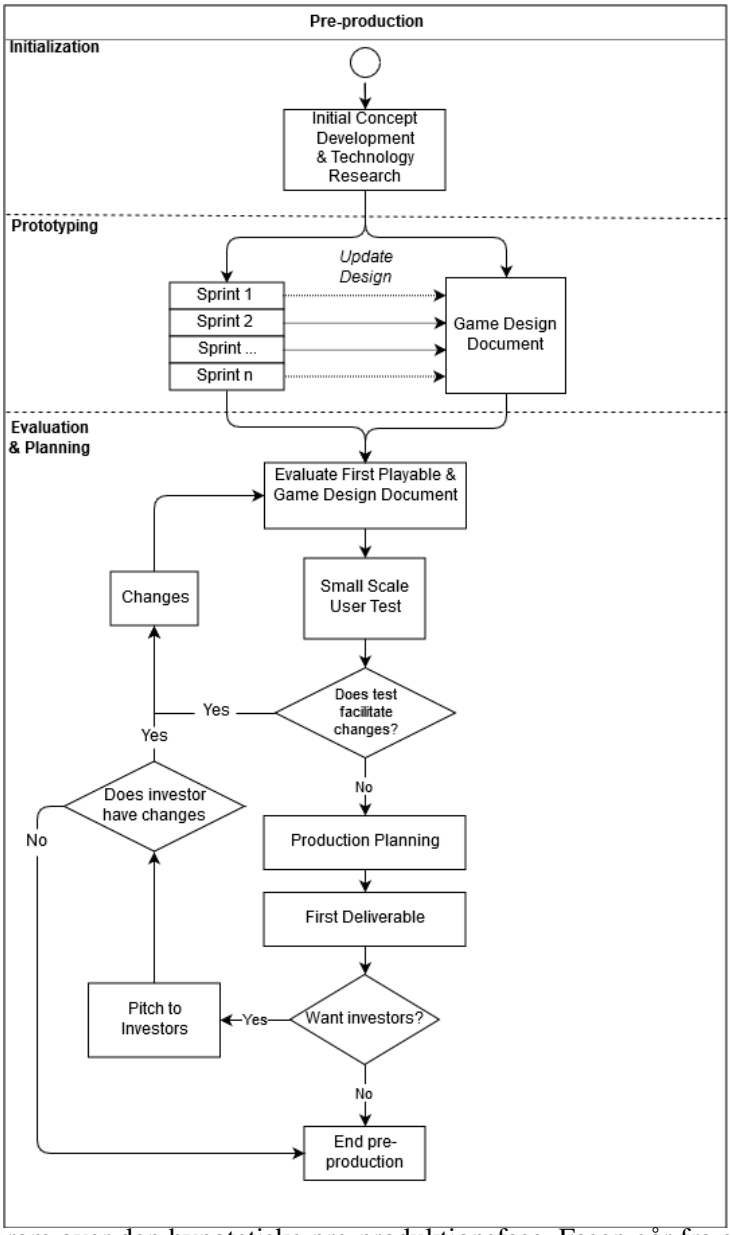
Forslag til den fulde arbejdsproces

På basis af teorien som beskrevet i tideligere afsnit, samt vores erfaring med spiludvikling i dette semester, har vi udarbejdet en foreslået arbejdsproces som andre spiludviklere ville kunne følge. Dette kapitel er dedikeret til en redegørelse for denne arbejdsproces.

Arbejdsprocessen er rettet mod mindre udviklingsgrupper med manglende erfaring inden spiludvikling. Derfor, er processen struktureret ud fra antagelsen, at gruppen ikke er i stand til at udføre adskillige større processer i parallel. Processen antager at gruppens medlemmer ikke er specialister og forsøger at strukturere arbejdet på en sådan vis, at alle gruppemedlemmer deltager flydende i alle arbejdsopgaver. Processen antager ingen fastsat tidsperiode for udviklingsprocessen og lader administration og planlægningen af denne op til gruppen selv. Processen forholder sig overordnet, da processens detaljer er afhængig af faktorer som gruppens opbygning, projektets kompleksitet og spillets genre.

Processen er, ligesom i teorien, opdelt i faserne *Pre-produktion*, *Produktion* og *Post-produktion*. Hver fase har en unik struktur og råder over en række aktiviteter som vil blive uddybet i følgende afsnit.

4.1 Pre-Produktion



Figur 4.1: Procesdiagram over den hypotetiske pre-produktionsfase. Fasen går fra et indledende koncept og eksperimenter og udvider dette løbende. Undervejs i fasen bliver produktionsplanlægning og user testing udført og til sidst bliver design dokumentation kogt ned til et overordnet macro design, som produktionsfasen kan baseres på.

Pre-produktion er den første fase i processen. Formålet med denne er at udvikle og konkretisere spillets struktur og temaer, samt give en indledende forståelse for hvordan spillets implementation skal foregå. Pre-produktionen handler overordnet om eksperimentation på spillets koncept og antager at gruppen besidder et indledende koncept inden fasen sættes i gang.

Pre-produktion er opdelt i tre under-faser med hvert sit formål:

- Initialization: Hvor spillets koncept udvides og ideer til spillets features bliver genereret.

- Iteration: Hvor ideerne genereret i *initialization* bliver afprøvet i diverse prototyper og konceptet konkretiseres.
- Evaluation: Hvor pre-produktionens endelige koncept evalueres, testes og eventuelt præsenteres for potentielle investorer. Her bliver produktionens plan også udviklet på basis af viden fået fra pre-produktionen indtil videre.

Initialization

Pre-produktionens første fase er initialization. Hovedfokus i initialization er at danne et grundlag hvorpå at iterationer af prototyper kan udføres og funktionalitet tests kan bygges. Fasen består udelukkende af design- og orienteringsmøder som fokuserer på udviklingen af dette udviklingsgrundlag dvs. konceptet og dets understøttende elementer, indledende tanker om Spillets genre, overordnede tema og mulige kravsspecifikation til testning. Derudover skal der også lægges op til det rent tekniske aspekt af prototyping, så fasen vil også involvere undersøgelse og udvælgelse af en Game Engine til udvikling af prototyperne.

Vi har valgt at definere initialization som en separat fase fra iteration fordi vi mener at gruppen bør gøre sig tanker om spillets koncept, før prototyper begynder at udvikles. Dette tillader gruppen at aftale og samles om udgangpunkt inden det praktiske arbejde, og gør det lettere for gruppen som helhed at acceptere eventuelle ændringer i designet.

Når initialization-fasen er ovre, burde gruppen stå tilbage med en indledende baglog, hvori deres features fundet i løbet af fasen er blevet nednoteret. Desuden sal der være taget en beslutning om hvilke værktøjer der vil benyttes i processen, herunder Game Engine, og programmer til konstruktion af visuelle og auditive elementer. Disse værktøjer fastlægges i initialization-fasen for at udviklingsholdet kan få erfaring med dem gennem prototype udviklingen i Iteration fasen.

Iteration

Når initialization er overstået kan iterations-fasen begyndes. Formålet med iterationsfasen er at undersøge det praktiske aspekt af de elementer som eksisterer i kravsspecifikationen og løbene udvide den med flere. Kravsspecifikationen bør forholde sig meget levende, med hyppig udskiftning af ideer i takt med deres undersøgelse. Pointen er at udvælge de interaktioner som spiller bedst sammen og som fungere bedst i forbindelse med spillets temaer og målgruppe.

Overordnet består fasen af diverse iterationer, kaldet *sprints*, som hver især har til formål at videreudvikle på den centrale prototype. Det første sprint dedikeres til at implementere og undersøge de de fundne interaktioner i en prototype, og de efterfølgende sprints bygger videre på denne. Hvis udviklingen viser sig at kræve specielle teknologier vil disse skulle noteres og, afhængig af deres kompleksitet, vil det være en idé at udvikle prototyper til disse teknologier sideløbende.

De sidste sprint benyttes på at få omdannet prototypen til et stadie hvor den er et sammenhængende level det kan testes som helhed. Efter dette sprint skulle prototypen have form af en lille sektion af det endelige spil, som reflektere spillets centrale interaktioner og ideer.

Iteration-fasen er, i store træk, hovedformålet med pre-production, så størstedelen af pre-production burde bruges på denne fase.

Sprints i pre-production

Hvor deres endelige mål er at udvikle et færdigt og fuldt funktionelt første level af en kvalitet så det kunne optræde i det endelige produkt. Desuden skal der også udvikles et design dock der indeholder et macro design for spilet og dets struktur samt en baglog der indeholder en liste over testede features der skal bruges i det endelige spil.

Game Design Document

Løbende som elementer frasorteres og nye kommer til, vil spillets design ændre sig, formodentlig meget drastisk i visse tilfælde. Disse ændringer skal noteres og Game Design Document'et, som begyndte sin udvikling i initialization, vil dermed løbende skulle re-defineres og udvides.

Godkendelse af leverancer

Når iteration-fasen er ovre står gruppen helst tilbage med 2 leverancer: Pre-produktionens Game Design Document, og prototypen i sit endelige stadie, kaldet First Playable. Evaluation starter når det sidste sprint er udført og begynder med en vurdering af disse to leverancer; Gruppen går sammen i et par møder og diskutere hvorvidt de er tilfredse med de to leverancer i deres nuværende form. Afhængigt af udfaldet, kan spillet blive godkendt, iterationsfasen blive udvidet med flere sprints, eller spillet helt aflyst.

Første Bruger Test

Efter leverancerne er godkendt, bør First Playable-prototypen blive brugertestet for at finde eventuelle mangler i designet. Eventuelle mangler og deres resulternede ændringer i leverancerne, bør overvejes og de ændrede leverancer bør diskuteres igen i endnu et godkendelses-møde.

Ideelt burde pre-produktion indeholde flere bruger-tests, specielt undervejs i Iteration, men denne proces er designet til mindre grupper, som ikke forventes at have det tids- eller budgetmæssige overskud til at dedikere tid til formelle bruger-tests midt i prototype-udviklingen.

Produktionsplanlægning

Når testene er udført og leverancerne endeligt godkendt, bør gruppen gå i gang med udviklingen af produktionsplanen. Overvejelser som mandskab, budgettering, milepæle i produktionen og eventuelle ændringer i udviklingens teknologisæt, burde alle kommer ind i dette stadie. Det er formålet her at lægge en slagplan for produktions-fasens udvikling. Alt efter gruppens tilgængelige ekspertise og ressourcer inden for emnet, bør der også foretages risiko og kvalitetsundersøgelser i dette stadie.

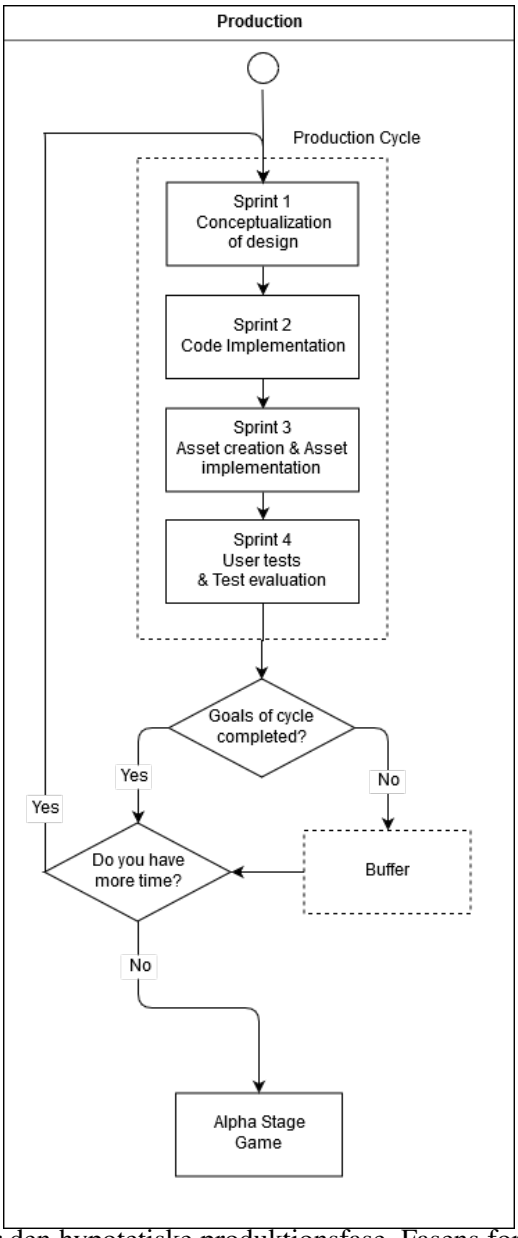
Pitch til investorer/publisher

Hvis produktionensplanlægningen viser at spillet vil kræve ekstra investering, kan gruppen på dette tidspunkt tage kontakt til eventuelle investorer. Hvis dette har været i tankerne undervejs i udviklingen, kan Game Design Document'et og First-playable prototypen bruges som præsentation for spillet. Gruppen bør være forberedt på eventuelle krav og ændringer fra potentielle investorer og disse bør tages op til godkendelse.

Afrunding af pre-produktion

Pre-produktionen afsluttes med en forberedelse af produktionsfasen. Vores foreslåede produktionsfase baseres på overordnet design dokumentation, så de designmæssige detaljer kan implementeres i eventuelle ny og forbedret form undervejs i produktionen. Derfor koges Game Design Document'et ned til et macro-design. Eventuelle licenser til produktionsværktøjer og teknologier anskaffes, og begyndelsen af produktionsfasen skematiseres.

4.2 Produktion



Figur 4.2: Procesdiagram over den hypotetiske produktionsfase. Fasens formål er at implementere spillet baseret på det koncept som blev udviklet i pre-produktionsfasen. Fasen struktureres som en række cyklusser som hver håndtere enkelte features. Spillet bliver udviklet iterativt til spillet indeholder mindst alle de meste centrale features, og eventuelt flere hvis der er tid.

Produktion er den anden fase i processen og formålet med denne er konkret at implementere spillets elementer i deres fulde form. Pre-produktion har givet os et macro design document som gruppen kan basere udviklingen på og planlægge iterationer efter. Derudover er der blevet udarbejdet en produktionsplan hvor vigtige milepæle er noteret som tillader gruppen at holde styr på hvor langt i processen de er nået. Der arbejdes på en enkel version af spillet, som iterativt bliver udvidet i løbet af produktionen. First-playable prototypen, som udviklet i pre-produktion, bliver ikke ført videre i produktions-fasen da denne ofte vil indeholde svagheder og fejl fra pre-

produktionens hektiske udviklingsforløb.

Fejl under produktionen kan potentielt være katastrofale hvis de ikke bliver opdaget i tide. Vores produktionsforløb inkludere hyppige bruger-tests, så gruppen har en større chance for at opdage fejl løbende, helst inden anden funktionalitet eller design bygges oven på det fejlagtige.

Produktionens cyklus

Produktions-fasen består udelukkende af en række produktionscykluser. Hver cyklus dedikeres til et specifikt område eller feature af spillet og består af 4 separate sprints med hver deres fokus i relation til iterationens overordnede mål. De 4 sprints er navngivet efter deres fokus, men i praksis forventer vi ikke at udviklingen kan separeres på en så sekventiel måde. Hvert af tre første sprint vil uundgåeligt involvere alle aspekter af udviklingen, men sprintets fokus vil sætte rammer for hvad der forventes færdigt i slutningen af sprintet. Macro-designet fra pre-produktion tillader os at lave micro-designet løbende i cykluserne og dette er også fokus af det første sprint i hver cyklus
Iterationernes strutkur er som følger:

- Konceptsprint; Det første sprint, hvor iterationens aftalte fokus konceptualiseres og detaljerne i dets design defineres.
- Implementationsprint; Det andet sprint, hvor konceptet implementeres i produktionens valgte Game Engine.
- SPRINTNAVN; Det tredje sprint, hvor implementationen fuldendes med illustrationer og lyd tilføjet. Alternativt kan dette bruges som catch-up sprint hvis de forrige sprints falder bagud.
- Testsprint; Det fjerde og sidste sprint, hvor den nyudviklede feature brugertestes.

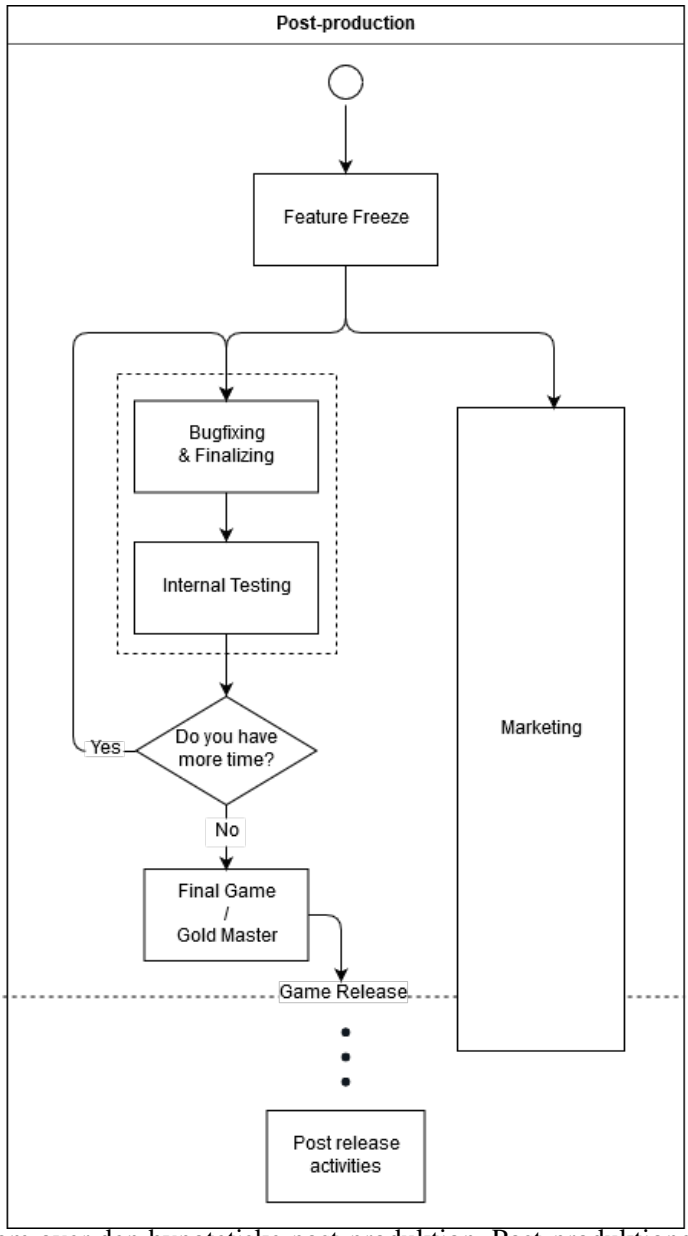
Buffer

Tidsmæssigt anbefaler vi mindst en 'buffer' på mindst 10% af den totale tid anvendt på produktionen. Der vil næsten uundgåeligt være diverse forsinkelser undervejs i processen, så buffertiden skal afsættes til at udbrede disse.

Spillet i alpha

Produktions-fasen afsluttes med en enkelt leverance: Spillet i dets alpha-stadie. Spillets alpha-stadie burde indeholde alle de mest centrale elementer i designet, samt eventuelle udbygninger og ekstra features som gruppen havde overskud til at udvikle under produktionsfasen.

4.3 Post-Produktion



Figur 4.3: Procesdiagram over den hypotetiske post-produktion. Post-produktionens formål er afrunding af spillet, hvor spillet i dets nuværende stadie optimeres, forbedres og fejl udrettes. Dette fortsætter indtil spillet udgives. Sideløbende køre en marketing-proces.

Post-produktion er den sidste fase i udviklingsprocessen og er dedikeret til at bringe produktet fra dets alpha-stadie til et stadie hvor det kan udgives og sælges. Produktionens intensitet burde falde en smule i denne fase, for at dedikere overskud til eventuel sideløbende marketing.

Når spillet er færdigtestet bliver det udgivet på udvalgte platforme.

Fokusset i denne model ligger på de præ-udgivelses aktiviteter som beskrevet nedenfor. I industrien fortsætter post-produktionen også som en proces efter spillet udgivet, men dette har vi ikke valgt at modellerer.

Feature freeze

Post-produktionen indledes med en såkaldt 'feature freeze', hvor arbejde på nye features stoppes og produktionsholdet skifter fokus til at fikse fejl i koden, samt generelt at optimere og forbedre spillets eksisterende features.

Bugfix & Testing

Post-produktionen er derfor opbygget af en central cyklus; Et sprint med testing og derefter et sprint hvor fejlene fundet i testene bliver udbedret.

Undervejs i produktionsfasen burde det nuværende design have været testet grundigt igennem og post-produktionen starter med et funktionelt produkt, så testing i post-produktionen har udelukkende til formål at finde utilsigtet opførsel i koden. Fejl burde prioriteres, så de mest kritiske fejl bliver fikset først.

Marketing

Hvis gruppen er i samarbejde med en eller flere investorer, kan man oftest overlade marketing til disse. Hvis ikke, er det en nødvendighed for gruppen at sørge for at visse mennesker ved at deres spil eksistere. Derfor bør en del af post-produktionens ressourcer afsættes til marketing, sideløbende med den centrale cyklus. Marketing kan betyde mange ting og oftest menes der en krævende proces som mindre udviklingsgrupper ikke har overskud til på dette stadie. Alligevel, vil vi foreslå et minimum af marketing over sociale medier og tage kontakt til eventuelle online-personligheder som kan sprede viden om spillet.

Diskussion

Vores opgave og udarbejdede proces bære præg af vores manglende personlige erfaring med det meste af selve processen. Grundet projektets begrænsede tidsrum, kom vi ikke længere i vores udførte arbejde end til tideligt i pre-produktionsfasen. Derfor er store dele af processen baseret på vores læste teori og sammenflætning heraf.

Hvorfor er vores tilegnede erfaringer fra processen relevante for dette projekt?

Vores udførte arbejdsproces var baseret på teori om eksisterende spiludviklingsprocesser i industrien og vores tilegnede erfaringer tillod os dermed at reflektere over teorien.

Hvorfor valgte vi at prøve spiludvikling selv i stedet for andre metoder

Selve det at udføre spiludviklingen selv lå i at få sat teorien i praksis. Derfra fik vi brugbar empiri i form af første hånds observationer af hvordan arbejdsprocessen fungerer, selvom den ikke nødvendigvis var ideel. Det skal dog siges at en alternative måde kunne godt have bestået i at lave anden kvalitativ og kvantitativ undersøgelse såsom interviews eller spørgeskemaer. Dette bør nok stadig gøres til at supplere empiri og derfor yderligere raffineret vores hypotetiske udviklingsproces.

Giver det god mening at vi forholder os så abstrakt og ukonkret i vores model

Forskellige spiludviklingsprojekter lægger sig op til forskellige processer. Udover forskelle grundet i faktorer som bemandskab og ekspertise, kan spil i forskellige genre kan have vidt forskellige elementer involveret og dermed vidt forskelligt fokus fra hinanden. Eksempelvis lægger visse spil fokus på et visuelt og lydligt indtryk, mens andre lægger fokus på engagerende interaktioner. Sådant skelnene elementer gør det svært at lave beskrivelse på lave abstraktionsniveau som stadig ville dække generelt.

Projektets fokus kunne alternativt have været på en proces opbygget specifikt til vores udvalgte type spil, men vi mente ikke at denne type proces-design ville være brugbar nok.

Hvorfor er initialization blevet lagt i en separat fase fra pre-produktions iterations fase

Vi har valgt at lægge initialization før iteration fordi den tillod os at tage en række vigtige valg omkring spillets design og teknologiretning. Havde vi derimod ikke taget disse beslutninger inden vi gik i gang med kodningen kunne man risikere at lave noget arbejde forgæves eller direkte forkert hvis der var uenighed eller forekom ændringer i det grundlæggende design. Derudover

oplevede vi også at det var nemmere at lave en plan over de sprints der skulle afholdes i den følgende fase. Vi fastlægger valg af teknologi i starten, da vi vurdere at tidligt erfaring med programmerne, giver mere værdi til produktionen, end at have en længere teknologi-udvælgningsproces. I og med at vores erfaring med at have en separat initialization fase i starten af pre-production var hovedsageligt positiv mener vi at det er gyldigt at udforme modellen på den måde.

Hvorfor har vi kun indsat en enkelt user test i pre-produktionen?

Ideelt set ville vi indføre flere user tests undervejs i pre-produktionen, men vi mente at dette ville tage for mange resurser fra den rapid prototypeing processen, da tid ofte skal afsættes til analyse af sådanne user tests; Pre-produktionen er relativt kort og fokuset i fasen er på eksperimentering og afprøvning af ideer. Dertil er vores proces overordnet designet med en mindre gruppe uerfarne udviklere i tankerne og vi mener derfor at løbende user tests på dette plan ville tage en for stor del af den tilgængelige tid. Vi mener dog at mindst én user test skal udføres, så designet kan relateres til brugeroplevelser og eventuelle ændringer laves før produktionen går i gang.

Hvorfor er Buffer elementet placeret hvor det er i modelen

Præcist hvor i vores produktions-model at vi skal placere vores buffer, er vi usikre på. Vi har valgt at placere den udenfor produktionsforløbet, men den kunne have været placeret inden i.
Grunden til vi har placeret bufferen udenfor produktionsforløbet, er for at symbolisere at den ikke på noget tidspunkt reelt skal blive en fast del af produktionsforløbet, bufferen er tiltænkt det der ligger i ordet. Den benyttes udelukkende i tilfældet af at man slutter produktionsforløbet, på sådan en måde at det er nødvendigt at tage buffer tid, for at kunne nå et minimal viable product. Vi ligger deruodver stort fokus på hyppige user-tests i vores produktionsforløb og vil derfor ikke lade bufferen udskyde testene.

Hvorfor er produktionscyklus sprintsne ikke mere eller mindre opdelt.

Længden på produktionscyklussen er ikke mindre opdelt da hele cyklussen er konstrueret i forhold til user-test. Derfor er de vigtigste aspekter får testet fokuseret i starten af cyklussen. På den anden side ville en total opdeling heller ikke være ideal da spillet mellem de samlede aspekter og deres samlede indtryk også vigtigt for user-testen.

Hvorfor beskæftiger vi os ikke med post-release activities.

Vi undlader at beskrive post-release forløbet fordi de aktiviteter der forgår efter udgivningen af et spil er en hel anden form for softwareproces. I og med at det ville kræve et andet teoretisk og metodisk grundlag bevæger det sig også ud af projektets omfang og formål. Derudover er et post-release forløb ikke en nødvendighed for at producere et spil.

Konklusion

Hvordan kan der på basis af industristandarder for spiludviklingsforløb, struktureres og planlægges, et spiludviklingsforløb som ville kunne udføres af os.

I dette projekt har vi udviklet og modelleret et spiludviklingsproces fra læste industristandarder. For at gøre dette måtte vi svarer på følgende underspørgsmål:

Hvilke softwareudviklingsprocesser er unikke for spiludvikling?

Spiludvikling er primært et interaktions-drevet medie. Spil bedømmes derfor primært på deres interaktive egenskaber og dette er som følge et langt større fokus i selve udviklingsprocessen. Tests af design forekommer hyppigt i spiludviklingsprocesser og design-revisioner undervejs i udvklingsprocessen er næsten garanterede. Interaktionsdesign betyder så meget for spiludvikling, at strukturen for den overordnede proces drejer sig om det; Unikt for spiludvikling, og samtidigt generelt for spiludvikling, er opdelingen i de tre faser *pre-production*, *production* og *post-production*. Samtidigt er unikke implementationer af interaktion også vigtigt; Den 'optimale' interaktion er ikke et begreb man snakker om inden for spil, derimod forsøger spil at skille sig ud med forskellige typer for interaktion. Hele pre-production-fasen er til for at tillade eksperimentation inden for interaktionen og eventuelt opdage nye, spændende måder at strukturere interaktion. Derudover besidder spiludvikling også deres egne discipliner inden for QA-testing, såsom balance-testing.

Hvilken indflydelse har størrelsen af udviklingsholdet på spiludviklingsprocessen?

Teorien og vores personlige overvejelser i henhold til proces-design har gjort det tydeligt at udviklingsholdets størrelse har en markant indflydelse på processens overordnede struktur; som udviklerhold vokser er det muligt at specialisere designere og udvikling i flere og flere kategorier. Som følge kræver større udviklingshold også mere komplekse og formaliserede styringsprocesser; specialiserede udviklere tillader leverancer at deles op i flere del-leverancer af forskellige typer. Hver af disse forskellige typer af leverance kræver deres egen proces, og flere processer kræver mere koordinering mellem processer.

Hvordan bliver der gjort brug af modellering inde for spiludvikling?

Som koordineringens kompleksitet stiger, bliver der også brug for mere dokumentation, herunder modellering som kommunikationsmiddel: I mindre udviklingshold har udviklere ofte meget

direkte kontakt med hinanden, hvilket tillader kommunikation af design gennem uformel kommunikation, men i større udviklingsforløb går denne direkte kommunikation tabt, så designere er afhængige af indirekte kommunikation i form af dokumentation og forklarende modeller til at kommunikere deres design, både internt og eksternt til eventuelle investorer.

Hvordan bliver der gjort brug af Agile arbejdsformer i spiludvikling?

Som beskrevet i teorien, udvikles størstedelen af spillet som produkt i produktion-fasen, men grundet spiludviklingens fokus på interaktionsdesign, præges selv produktion oftest af store design-beslutninger undervejs. Derfor er detaljeret forhånds-planlægning af produktionen oftest en nytteløs opgave; hvis spillets design konstant gennemgår ændringer, må planen ligeledes ændres løbende for at reflektere det, hvilket i sidste ende gør den dybdegående planlægning upålidelig og til et konstant tidskrav. Agile arbejdsformer tillader derimod denne type udvikling; hvis spillet udvikles iterativt tillader det planlægningen kun at fokusere på en enkelt iteration ad gangen. Eventuelle fejl fundet i designet i en iteration, kan dermed kun ugyldiggøre den indeværende iterations planlægning, frem for hele processens plan som helhed.

Perspektivering

7.1 Videre arbejde med projektet

Det ville være af stor værdig for vores opstillede udviklingsproces at få den ordenligt bedømt enten ved selv at følge processen til ende eller ved at få af nogen inden for industien.
At interviewe en industri insider i forhold til spiludvikligsprocesser ville desuden også være et stor plus for vores kildebase da alt vores nuværende kilder er udelukkende litteratur.
Oplagt for videre arbejde med projektet kunne også være at inkorporere i vores model, de elementer vi har afgrænset os fra.

Modellen udvidet med ting fra afgrænsningen En oplagt måde at arbejde videre med dette projekt kunne være udvidelsen af vores nuværende hypotetiske arbejdsproces med de elementer som vi har valgt at udelade gennem afgrænsningen. Risikoanalyser, markedesanalyser og budgetteringsmetoder kunne redegøres for og inkorporeres som en del af den udviklede model. Der er allerede blevet gjort nogen indledende tanker omkring inklusionen af disse i form af hvornår i processen vi nuværende ville vurdere de skulle forekomme.
Risikoanalyse starter i pre-prodution -prodution planing og udføres kontinuert som en del af sprints indtil release.
Markedesanalyse starter i pre-prodution -prodution planing og udføres kontinuert indtil release især i sammenhæng med marketing.
Budgettering udføres i pre-prodution -prodution planing.

Post-release activities

Vi har i vores model udeladt post-release activities som Post Mortems og generel maintenance. Disse kunne være interessante at udforske og udvide modellen med.

Litteratur

Litteratur

[Adams, 2013] Adams, E. (2013). *Fundamentals of Game Design*. Pearson Education.

[Aleema, 2011] Aleema, Capretz, A. (2011). Game development software engineering process life cycle: A systematic.

[Blender, 2018a] Blender (2018a). https://docs.blender.org/manual/en/latest/rigging/armatures/bones/editing/naming.html Sidst besøgt 18. december 2018.

[Blender, 2018b] Blender (2018b). https://docs.blender.org/manual/en/latest/render/cycles/nodes/types/shaders/principled.html Sidst besøgt 18. december 2018.

[Butler, 2018] Butler, N. (2018). "What is Scrum? The Product Owner primer". https://www.boost.co.nz/blog/2018/03/what-is-scrum-product-owner-primer.

[Castlevania, 1986] Castlevania (1986). https://www.spriters-resource.com/nes/cv/sheet/8477/ Sidst besøgt 18. december 2018.

[Cerny, 2002] Cerny, M. (2002). "D.I.C.E. Summit 2002 - Method". https://www.youtube.com/watch?v=QOAW9ioWAvE, *Besøgt 18. december 2018*.

[Chandler, 2013] Chandler, H. M. (2013). *The Game Production Handbook, 3rd Edition*. Jones & Bartlett Publishers.

[Chang, 2018] Chang, T. (2018). Things to know about working in the gaming industry. https://medium.com/cup-of-tea/working-in-games-4de9b0a5df7e *Besøgt d. 14/12 2018*.

[Costikyan, 1994] Costikyan, G. (1994). I have no words and i must design: Toward a critical vocabulary for games. http://www.costik.com/nowords2002.pdf *Besøgt 18. december 2018*.

[Esposito, 2005] Esposito (2005). A short and simple definition of what a videogame is. http://www.utc.fr/~nesposit/publications/esposito2005definition.pdf Sidst besøgt 18. december 2018.

[Forum, 2013] Forum, S. (2013). https://superuser.com/questions/674843/what-is-the-difference-between-tessellation-and-displacement-bump-mapping Sidst besøgt 18. december 2018.

[Fullerton, 2014] Fullerton, T. (2014). *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. AK Peters/CRC Press.

[IGN, 2006] IGN (2006). The game production pipeline: Concept to completion. https://www.ign.com/articles/2006/03/16/the-game-production-pipeline-concept-to-completion *Besøgt 18. december 2018*.

[Intel, 2011] Intel (2011). https://software.intel.com/en-us/articles/creating-uvs-for-characters-in-autodesk-maya Sidst besøgt 18. december 2018.

[Jesper, 2003] Jesper, J. (2003). *The Game, the Player, the World: Looking for a Heart of Gameness.*

[Järvinen, 2003] Järvinen, A. (2003). Making and breaking games: a typology of rules. https://www.researchgate.net/publication/221217603_Making_and_breaking_games_a_typology_of_rules *Besøgt 18. december 2018.*

[Järvinen, 2004] Järvinen, A. (2004). A meaningful read: Rules of play reviewed. http://www.gamestudies.org/0401/jarvinen/ *Besøgt 18. december 2018.*

[Kukhnavets, 2016] Kukhnavets, P. (2016). "Agile vs Waterfall: Pros and Cons, Differences and Similarities". https://blog.ganttpro.com/en/waterfall-vs-agile-with-advantages-and-disadvantages/.

[Lin, 2015] Lin, E. (2015). Metal gear solid v cost over $80 million to develop, says nikkei report.

[Luis Levy, 2010] Luis Levy, J. N. (2010). *GAME DEVELOPMENT ESSENTIALS, GAME QA TESTING*. Delmar, Cengage Learning.

[Machinations, 2018] Machinations (2018). Machinations documentation. https://docs.machinations.io/ Sidst besøgt 18. december 2018.

[Makuch, 2015] Makuch, E. (2015). This is how much the witcher 3 cost to make.

[Neumann, 1944] Neumann, M. (1944). *Theory of Games and Economic Behaviour*.

[O'Hagan, 2014] O'Hagan, O'Connor, C. (2014). Game development software engineering process life cycle: A systematic review.

[Ryan, 1999] Ryan, T. (1999). The anatomy of a design document, part 1: Documentation guidelines for the game concept and proposal. http://www.gamasutra.com/view/feature/131791/the_anatomy_of_a_design_document_.php?page=2, *Besøgt d.14/12 2018*.

[Salen, 2004] Salen, Z. (2004). *Rules of Play: Game Design Fundamentals.*

[Sarinho, 2014] Sarinho, A. (2014). A feature model proposal for computer games design. https://www.researchgate.net/publication/228755345_A_Feature_Model_Proposal_for_Computer_Games_Design *Besøgt 18. december 2018.*

[Schwaber and Sutherland, 2017] Schwaber, K. and Sutherland, J. (2017). The scrum guide. *Scrum Alliance*, 21.

[Sicart, 2008] Sicart, M. (2008). Defining game mechanics.

[Sommerville, 1982] Sommerville, I. (1982). *"Software Engineering*

.

[Wijman, 2018] Wijman, T. (Apr. 30, 2018). Mobile revenues account for more than 50 % of the global games market as it reaches $ 137.9 billion in 2018. *Newzoo*, newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/ .

[Zimmerman, 2004] Zimmerman, E. (2004). *Narrative,Interactivity, Play, and Games.*