Forside

Indholdsfortegnelse

[Specialisering – Proces 2](#_Toc369814340)

[Indledning 2](#_Toc369814341)

[Specialiserings Plan 2](#_Toc369814342)

[Opstart på projektet og planlægning 2](#_Toc369814343)

[Concept 2](#_Toc369814344)

[Forklaring af Første koncept som endte i skraldespanden: 2](#_Toc369814345)

[Forklaring af hvorfor det første koncept skrottes: 3](#_Toc369814346)

[Forklaring af Koncept nr. 2: 3](#_Toc369814347)

[Concept - Vurdering af arbejds – og læringsprocessen 3](#_Toc369814348)

[Concept - Refleksion af eget udbytte 3](#_Toc369814349)

[Concept - Konklusion 3](#_Toc369814350)

[Elaboration 4](#_Toc369814351)

[Opbygning af scener og arkitektur i Unity 4](#_Toc369814352)

[Opstart på projektet i Unity med 3D objekter 4](#_Toc369814353)

[Opsætning af spillet med GUI elementer 5](#_Toc369814354)

[Praktisk Problem med opsætning af GUI elementer 5](#_Toc369814355)

[Opbygning af Scene 6](#_Toc369814356)

[Scene problem med at have forbindelse til de forskellige scripts. 6](#_Toc369814357)

[Game manager objektet 6](#_Toc369814358)

[metoder der kan opretholde persistens på game manageren 7](#_Toc369814359)

[Yderligere GUI med terninger 8](#_Toc369814360)

[Arbejdet lidt med trello i slutfasen 8](#_Toc369814361)

[Elaboration - Vurdering af arbejds – og læringsprocessen 8](#_Toc369814362)

[Elaboration - Refleksion af eget udbytte 9](#_Toc369814363)

[Elaboration – Konklusion 9](#_Toc369814364)

[Konklusion 9](#_Toc369814365)

Bilag:

B1 - Specialiserings plan

B2 – Spilleregler

# Specialisering – Proces

## Indledning

Denne rapport skal klarlægge Processen i mit specialiserings projekt på 4. semester. Først beskrives processen omkring specialiseringsplanen, som skulle udføres inden projektet kunne gå i gang. Projektet har jeg delt op i opstart og planlægning, Concept og Elaboration. Opstart og planlægning består af at finde en Design Model til at udvikle spil med, som jeg kunne arbejde ud fra og lave en tilhørende planlægning ud fra den model. I Concept er spillets koncept, design og ide udviklet. Elaboration skal ideen fra Concept udføres

Dertil har jeg så haft nogle delmål i de forskellige faser, som jeg har kunnet følge op på.

## Specialiserings Plan

Inden projektet startede blev der udarbejdet en specialiserings plan som skulle beskrive hvad det var for et arbejde der skulle udføres og klarlægge mine mål for opgaven. Som hoved emne havde jeg at udgive et spil til android devices fra Unity 3D, helt specifikt på Samsung Galaxy S2, som er den android Device som jeg har tilgængelig nu. Som delmål til spillet skulle der laves en AI så det er muligt at gennemføre et singleplayer spil. Et andet delmål var at lave/ designe en god State machine som er mere lig den undervisning vi fik på sidste semester. Næste delmål var at udarbejde en forretnings strategi, i forbindelse med udgivelse af spillet på Google Play med henblik på at senere kunne lave et fritids firma hvor jeg kan udgive spil og tjene lidt penge på det.

## Opstart på projektet og planlægning

Til at udvikle spillet vil jeg gerne have et design værktøj, som jeg kan bruge igen efterfølgende til flere spil eller til videre udvikling af spillets videre forløb. Derfor har jeg søgt sådan et værktøj. Det som jeg har fundet er en game designer på nettet som har lavet en blog omkring game design, som er simpel og nem at forstå. Det er hans model eller template som jeg har taget og vil arbejde ud fra. Bloggen kan findes på: http://gamedesigntools.blogspot.nl/

Fra templaten har jeg valgt at bruge Concept og Elaboration.

## Concept

### Forklaring af Første koncept som endte i skraldespanden:

Det første koncept jeg lavede var et spil hvor der er 2 spillere som skal spille mod hinanden, enten spiller mod spiller eller spiller mod AI. Hver spiller har et tårn som er spillerens liv eller hitpoints og tårnet er så samtidig modspillerens mål for at vinde spillet. For at kunne opnå det mål, har hver spiller et sæt kanoner som man kan skyde med for at ødelægge modspillerens tårn. Hver spiller skulle kunne vælge en karakter af 3 forskellige slags, altså en avatar for spilleren, som løbende får mere erfaring og åbner for forskellige skills jo højere man stiger i level. På de højere niveauer kan spilleren også vælge flere forskellige kanoner. Spillet skal være tur-baseret og for at spilleren kan bruge sine kanoner eller karakter skills skal han bruge nogle begrænsede ressourcer, som kunne være i form af for eksempel krudt. Meningen med gameplayet er så at spilleren i hver tur, skal tage en strategisk beslutning om hvad for nogle skills eller kanoner der er bedst at bruge ressourcer på, for at have størst chance for at vinde kampen.

Planen med at der skulle være 3 forskellige karakterer, som man kan vælge i mellem, var for det første at variere gameplayet ved at de forskellige karakterer har deres egen unikke skills, som giver forskellig oplevelse af gamplayet og dermed forlænger spillets holdbarhed. Et andet mål med det var at jeg gerne vil lave noget digitalt salg på Google Play, hvor man så kunne købe udstyr til sin karakter og til kanonerne kunne der laves forskellige skins som der også kunne sælges. Dog med henblik på at spillet ikke må være pay to win, eller ikke virke som det. De features som man skal kunne købe er kun visuelle forbedringer af spillerens karakter og nye skins til kanoner.

### Forklaring af hvorfor det første koncept skrottes:

Efter at have stiftet bekendtskab med Unity 3D fandt jeg ud af at der er mange ting som ville være meget tidskrævende i forhold til de mange animationer og grafiske objekter der skal laves til spillet. Og jeg ville ikke kunne nå at få ret meget af spillet færdig til aflevering af specialet. Samtidig kunne jeg se på mine skitser af spillet at det havde en høj kompleksitet, som ville gøre det svært at se på mit target device, Samsung Galaxy S2. det som jeg gjorde for at sikre mig at det var for småt, var at hente et spil på Google Play som havde en lignende kompleksitet. Jeg kunne tydeligt se at spillet var alt for komplekst til det format og man kunne ikke rigtig se hvad der foregik på skærmen. Men ideen ville måske passe godt til tablet størrelse. Derfor har jeg valgt at droppe dette første koncept og lave et mere simpelt koncept, hvor jeg kan nå at få de vigtigste hovedfeatures til spillet færdig.

### Forklaring af Koncept nr. 2:

For bedre at kunne opnå en form for funktionalitet i spillet er jeg kommet op med Koncept nr. 2 som er en mere simpel udgave. Spillet skal bestå af 2 spillere og en spilleplade med 6 terninger på. Spiller nr. 1 starter med turen og kan kaste terningerne ved at trykke på en knap. Når spilleren er færdig, er det spiller nr. 2 tur til at kaste terningerne og sådan fortsætter det til en af spillerne har vundet. For yderligere oplysninger af reglerne til spillet henvises der til B2 – Spilleregler bilaget. Forskellen fra det første koncept er at mange af de 3D modeller som jeg skulle have haft med i stedet for skal laves af GUI elementer. Det vil sige at spilleren, terninger og spillepladens score er repræsenteret i GUI elementer. Det eneste der bliver sat op i form af 3D modeller er en kanon, et tårn og selve spillepladen. Hvis der bliver tid til det, vil jeg lave en animation hvor kanonen skyder på modstanderens tårn, når man giver modstanderen minus point. Den opgave der skal løses som går ud over de 2 kurser der hører til specialet, vil så være udvikling af arkitektur og opbygning af spillet i Unity. Med disse mere simple features er det muligt at opnå et færdigt produkt til tiden.

### Concept - Vurdering af arbejds – og læringsprocessen

Den arbejds- proces som jeg har været igennem i koncept fasen, har været en trial and error proces, da jeg ikke har prøvet at udvikle et spil koncept før. Det har været en god proces fordi at jeg har prøvet at lave 2 forskellige koncepter, hvor af det første koncept ikke kunne udføres pga. tidsbegrænsningen af projektet. så jeg var nød til at lave en iteration mere i koncept fasen, hvor jeg så udviklede koncept nr. 2 som er et mere simpelt spil, men dog stadig med masser af muligheder for at udvikle det til et større koncept, ved for eks. At tilføje animationer.

### Concept - Refleksion af eget udbytte

### Concept - Konklusion

## Elaboration

### Opbygning af scener og arkitektur i Unity

### Opstart på projektet i Unity med 3D objekter

Som en lille indledning til opstarten af spillet i Unity, vil jeg lige forklare hvad det er for nogle objekter og funktioner, som spillet skal bestå af. Dette er for at vise hvad det er jeg skal udvikle. Spillet skal have en start menu, hvor man kan se 2 bokse, som hører til hver sin spiller. I hver boks kan man indtaste sit spiller navn. Når man har gjort det kan man trykke start game, som starter spillet. Spillet består af en spilleplade med 6 terninger placeret i midten. I hver ende af spillepladen er spillerne placeret. På hver spiller er et spiller ikon med hver sin farve, så man kan kende forskel. Hver spiller har en score og et navn så man se hvor mange point man har og sin spillers navn. Ved terningerne er der en knap til at kaste terningerne med. Når teringerne bliver kastet bliver samlet en score spillepladen som bliver vist på en spilleplade score. Denne score skal så trækkes fra spillerens point alt efter hvad score de lander på ved slutningen af turen. Når en af spillerne har vundet spillet, skal der vises en besked der skriver hvilken spiller som har vundet spillet og derefter skal main menuen loades igen. sådan som det er nu er der ikke nogle tilstande eller objekter der skal vedvare efter at et spil er slut. Dog er det meningen at jeg i en senere udgave vil gemme en historik på hver spiller, der viser hvor mange spil de har vundet. Fra hovedmenuen kunne der så laves en knap til en scene der viser en top ti liste over de spillere der har vundet flest spil. For yderligere oplysninger omkring spillereglerne se bilag B2 – SpilleRegler.

Når man har cirka ingen erfaring med Unity, kan det være svært at vide hvor man skal starte. Men som udgangs punkt har jeg det bedst med at lære nye ting ved at bare at gå i gang med at arbejde med det og forsøge mig frem. Erfaringer hænger bedre fast når jeg bruger dem i stedet for at lave en masse teoretisk research først, også gå i gang med arbejdet bagefter. Nogle gange kan det selvfølgelig være nødvendigt at slå nogle teoretiske ting op, men det tager jeg løbene som jeg får brug for det. Så derfor har jeg startet med at oprette en scene i Unity som skal fungere som spillepladen i spillet. Med det viden jeg havde om Unity på dette tidspunkt, vidste jeg at man kunne finde og hente 3D modeller og sætte dem ind i Unity. Så jeg gik i gang med at søge på nettet efter modeller på blandt andet www.turbosquid.com og Unitys egen Asset store. Efter have fundet de modeller som jeg mente at jeg skulle bruge, begyndte jeg at sætte dem ind i Unity. Det gik sådan set nemt nok da man kunne importere direkte til Unity fra websiderne af. Nu havde jeg så fundet en spilleplade i form af en tønde, 6 terninger og 2 Menneske modeller, en kanon og et tårn. Det som jeg så troede at jeg skulle i gang med var lige at lave et par animationer til de forskellige modeller. For eks. Var planen at jeg ville lave så når man kastede terningerne, skulle de flyve lidt op i luften, rotere i luften og ud fra en random værdi mellem 1-6 skulle terningen så lande med den værdi op ad. Når en spiller gav modspilleren minus point ville jeg gerne have at en kanon skød modspillerens tårn. For at kunne lave disse animationer vil de skulle laves i et eksternt animations program, som kan være dyre at anskaffe sig. Eller også skal man bruge Unitys fysik til at manipulere objekter med. Uanset hvad kunne jeg godt se at det ville være alt for tidskrævende med den tidsramme jeg har til rådighed. Samtidig er det mere arbejde for en grafiker at lave den slags. Så jeg bestemte mig for at jeg ikke ville bruge tiden på det. Dertil kan det også tilføjes at de modeller som man finder ikke nødvendigvis har alle de ting der skal bruges for de for eksempel lave et terning kast.

Hvad gør jeg nu ? for at kunne nå at lave noget funktionalitet af spillet i Unity, var jeg nød til at komme op med ny plan, som matchede den tidsramme der var til projektet. Så jeg besluttede mig for at lave det meste af spillet i GUI elementer, så hver terning for eks. Er en GUI button med et billede af en terning på. Og spilleren skulle også repræsenteres af GUI elementer med textlabel til score, navn og et billede til spiller ikon. Det eneste der skal være en 3D model er spillepladen. Hvor spilleren og terningerne vises på. Med den nye plan var det så på tide at sætte det i værk.

### Opsætning af spillet med GUI elementer

Efter fjernelsen af alle 3D modellerne, kunne jeg gå i gang med at lave GUI elementer i stedet for. Heldigvis er Unitys Dokumentation omkring GUI elementer godt beskrevet og der er tilmed et nogle simple tutorials med de helt basale elementer som man hurtigt kan komme i gang. Efter at have læst dokumentationen var det nemt at finde ud hvad for nogle elementer jeg skulle have til de forskellige objekter i spillet. Til terningerne har jeg fundet et billede som skal vises på en knap. For ikke at hardcode stien til billederne af terningerne. Søgte jeg i Unitys dokumetation fandt en Resources klasse som netop kan håndtere dette. Jeg fandt ud at man skal oprette en Resources mappe i sin Assets mappe, i den mappe hvor projektet er oprettet. Når det er gjort kan man lægge sin billed filer i den mappe også i Unity kan man bruge metoden Resources.Load(ressource navn) hvor man giver den navnet på recourcen som input. Dermed returneres den ressource så til unity, i mit tilfælde et billede af en terning.

### Praktisk Problem med opsætning af GUI elementer

I forbindelse med opsætning af de forskellige GUI elememter stødte jeg en i et problem som kostede lidt ekstra tid. Når man sætter en GUI.Box op for eks. Skal den have et Rect objekt med x,y, længde og bredde. Altså et rektangel med x og y som nulpunkter og længde og bredde på rektanglet. det gav et problem da jeg havde lavet et build af mit spil som var på en skærm størrelse 800x600 og det build som kører i editoren kører med den skærmstørrelse som editoren kører på, i mit tilfælde 1920x1080. Det resulterede i at build jeg havde lavet var mange af mine GUI elementer ikke at se. Jeg fandt ud det af fordi jeg lige skulle teste mit spil i et build hvor jeg valgte en mindre opløsning end den der kørte i editoren. Så jeg lavede et nyt build med en større opløsning også kunne jeg se alle mine GUI elementer igen. Men problemet var ikke løst, forspillet skulle jo gerne kunne køre på flere opløsninger. Så igen tilbage til Unitys forum og finde ud af hvad man gør for få en relativ skærmplacering. Mission succes, Unity har en metode som returnerer skærmens længde og bredde. Ved at bruge disse metoder kunne jeg placere GUI elementerne relativt til den skærm buildet kører på. Metoderne er Screen.Length og Screen:height. Også har jeg divideret dem med 2 for at lægge alle elementerne ud fra præcis i midten af skærmen, på den måde vil det altid være centeret i midten af skærmen på alle skærm størrelser.

### Opbygning af Scene

Opbygningen af de forskellige objekter i spillet har jeg lavet ved at tilføje tomme game objekter, som jeg har kaldt for eks. Player og dice også oprettede jeg scripts som jeg så kunne tilføje på de game objekter. Da jeg skal have 6 terninger i spillet, lavede jeg så først en terning, som jeg så kopierede 5 af. Til terningerne lavede jeg så et script, som vidste en Gui button, med et billede af terningen. Af testmæssige formål lavede jeg det så at hver terning bare vidste værdierne 1-6, så jeg kunne se at knapperne viste et billede af en terning. Tilsvarende havde jeg lavet til player objekterne. 2 game objekter med scripts som viser en Gui Box på skærmen med et billede af et spiller- ikon, navn og score.

Nu kom jeg så til at tænke på at de game objekter jeg havde sat ind er mere eller mindre ens og skal udføre de samme opgaver. Det gælder både player og dice objekterne. jeg har hørt lidt om noget der hedder prefabs, men jeg vidste ikke hvad var og hvad man kunne med det, men havde en ide om at jeg kunne bruge det til at lave nogle objekter med dynamisk. Jeg besluttede mig for at slå det op på Unitys dokumentation, hvor der igen var god afklaring af hvad prefabs er og kan. Jeg vil kort beskrive hvad prefabs er, for at vise hvad jeg kan bruge det til og hvorfor det er smart. Prefab er en type af genbrugelig game objekt, som kan sættes ind i n antal scener, flere gange per scene. Når man tilføjer en prefab til en scene laver man en instans af den prefab. Alle prefab instanser er linked til den originale prefab og er stort set kloner af den. Lige meget hvor mange instanser der er i projektet, kan man lave ændringer til den originale prefab, vil blive tilføjet til alle de instanser der er lavet af den. Det må siges at være noget jeg kan bruge, så i stedet for at sætte 6 instanser af terninger ind i min scene, oprettede jeg i stedet en prefab af en terning. På den prefab, tilføjede jeg så scriptet til terningerne. Nu kunne jeg sætte 6 instanser af min terning ind. Nu er det smarte så at hvis jeg skal ændre noget i mit terninge script, vil ændringen slå igennem på alle instanser af den. Det samme gælder for player objektet, der oprettede jeg jeg også en prefab. Med disse ændringer vil det også være nemmere at udvide spillet, i og med at det er nemt at tilføje flere terninger og flere spillere.

### Scene problem med at have forbindelse til de forskellige scripts.

Nu skal jeg i gang med at lave funktionaliteten i spillet. Altså binde alle delene sammen og vise det på skærmen. Så at en spiller kan kaste terninger for eks. Samtidig skal jeg også have lavet en så at man startet fra en main menu, hvor spillerne indtaster navne og trykker start game, også skal deres navne loades og vises på deres GUI Box i scenen som viser spillepladen. Det vil sige at jeg de to navne som bliver indtastet fra main menuen skal jeg have ført videre til spilleplade scenen og gemt dem på player objekterne. player objekterne har en string variabel, hvor navnene skal gemmes i. Den måde som jeg har opbygget scenerne på i denne iteration, har jeg ingen player objekter på main menuen, da den består af et main menu GUI script med knapper og tekstfelter. Det har så ledet mig til at tænke på at jeg ikke har adgang til de objekter som ligger i spilleplade scenen, da de objekter først bliver instantieret når scenen bliver loadet. Det har ledt mig til at jeg gerne vil have en klasse som står for ansvars fordeling af de forskellige objekter, og som står for instantiering af dem. Så jeg kan styre hvem der opretter dem og hvornår de bliver oprettet. Derfor er jeg gået i gang med at lave en game manager klasse.

### Game manager objektet

Hermed en lille forklaring af hvordan jeg vil lave mit game manager script og hvordan det er gået med det. Til at starte med har jeg lavet et tomt GameObject i unity som jeg har kaldt GameManager. Denne klasse er kernen i spillet og skal binde det hele sammen. Klassen skal stå for at oprette og at oprette et game objekt, player objekter og dice objekter. Game objektet skal indeholde alle de oplysninger som spillet skal holde styr på. Det vil sige spilleplade score, hvilken spiller har turen og hvem har vundet spillet osv. Derfor skal game manageren altid have forbindelse til gameobjektet og den skal hentes som singleton instans, da jeg skal sikre mig at det altid er den samme instans jeg får af game objektet, fordi at jeg gemmer oplysninger fra game manageren, som jeg vil sikre mig ikke bliver smidt væk før jeg selv lukker instansen af game objektet. I game manageren har jeg så lavet 2 lister, en til at indeholde player objekter og en til at indeholde dice objekter. Når game manageren starter har jeg lavet et loop der opretter objekterne og tilføjer dem til de lister, så jeg har nem adgang til dem i game manageren. Dermed er de også tilføjet dynamisk, så hvis jeg skal udvide til flere terninger og spillere er det bare at ændre de 2 loops. Samtidig skal den også står for at binde det hele sammen, så det bliver funktionelt. Det vil sige funktioner til at kaste terninger, tilføje spillere, have en reference til en liste af spillere, en reference til en liste af terninger og en kø af spillere som skal have turen. Jeg gik så i gang med at kode game manageren som medførte at jeg fik en masse fejl. Grunden til dette var at jeg ikke havde helt styr på mine gameObjekt navne og mine script navne. For eks. Havde jeg navngivet mine scripts det samme som det tilsvarende game objekt, som i game objekt der hed GameManager og så et script, dertil som også hed GameManager. Det gjorde det svært for mig fordi at når jeg skulle oprette et game objekt tilføjer jeg et script til objektet i samme ombæring hvilket gjorde at jeg fik castet retur typen forkert. Når man får et gameobjekt i stedet for et script, er der ikke ret meget der hænger sammen og det resulterede i at jeg fik en masse nullpointers. Efter at have navngivet mine scripts på ny, fandt jeg så fejlen og det var som sagt en forkert retur type.

Efter følgende ville jeg så lave en test for at se om min game manager blev oprettet som den skulle og at den oprettede en instans af game objektet. Så for at teste det lavede jeg nogle tekst labels og satte på spillet som skrev variabler ud fra game objektet, for at teste om objektet var instantieret og at der var tilskrevet nogle default værdier på variablerne. efter at jeg havde lavet ændringerne med script navnene kunne jeg se at de rigtige variabler blev printet ud på skærmen og det vil så sige at jeg med succes havde oprettet et game objekt på game manageren, som jo som sagt er det objekt der skal stå for at opretholde alle oplysninger om spillet. Bagefter lavede jeg den samme test med spiller og terning objekterne. Her brugte jeg bare det GUI som jeg rent faktisk havde lavet til dem. På spillernes GUI fik jeg skrevet de det navne ud som blev tastet ind fra main menuen og på GUI til terninger fik jeg vist en terninge værdi, som var blevet rolled på game manageren ved opstart af spillet.

### metoder der kan opretholde persistens på game manageren

For at kunne overføre spillerens navn fra main menuen til spil scenen, har jeg valgt at oprette Game manageren fra Main menu scenen, når der bliver trykket på start game knappen. For at kunne loade en anden scene i Unity, har jeg søgt i unitys officielle dokumentation og fundet metoden Application.LoadLevel som loader en scene ved at tage et scene navn eller scene index som input og derfra finde den specifikke scene. Men for at det kan virke skal hver enkelt scene i spillet tilføjes til projektets build settings, som der også står i dokumentationen. Dette gøres ved at vælge Filer -> Build Settings i Unity som åbner et vindue hvor et index af de scener som er tilføjet til buildet vises. Hver enkelt scene kan så vælges i projekt mappen og i build settings vælger man add current og nu bliver scenen tilføjet til listen af scnener i buildet. I dokumentationen står der yderligere at alle objekter der tidligere er oprettet eller loaded, vil blive destrueret når man loader en ny scene. Dette vil sige at man skal håndtere dette hvis ikke man ønsker at alle objekter bliver destrueret.

Nu hvor at jeg opretter Game Manageren i Main menu og skal videreføre den til hoved spil scenen skal jeg så finde ud hvordan dette gøres, da Application.loadlevel jo som sagt destruerer alle objekter. Til netop dette formål har jeg fundet en anden metode på unitys officielle forum som hedder Object.DontDestroyOnLoad. I beskrivelsen af metoden står der at hvis man kalder denne metode, på et game object eller component vil objektet vedvare med Application.Loadlevel. Dog er der nogle ting man lige skal være opmærksom på. Man skal være sikker på at de variabler som man gerne vil skrive til er instantieret før man skriver til dem. I forbindelse med min research på Application.Loadlevel og Object.DontDestroyOnLoad faldt jeg over en metode som hedder awake. Det som awake metoden gør, er at initialisere variabler eller objekter før spillet starter. Awake bliver kaldt efter at alle objekter er initialiseret så man sikkert kan skrive mellem objekter. Awake bruger man til at sætte referencer op mellem scripts også bruger man start til skrive værdierne mellem de forskellige objekter. Så i min game manager har jeg tilføjet denne awake metode som så skal oprette de objekter som jeg skal bruge. Det vil sige at oprette listen af spillere før jeg rent faktisk kan tilføje en spiller til den. Det er lige netop det som at jeg gør når jeg kalder min startstate på game manageren fra main menu scenen. Så ved at jeg har awake metoden hvor jeg instantierer listen af spillere, kan jeg oprette et player objekt og tilføje det til listen af spillere. Og alt dette sker selvom jeg ikke er i den scene endnu.

### Yderligere GUI med terninger

Efter at jeg havde fået referencerne og oprettelse af objekter til at virke. Gik jeg i gang med at tilføje funktionalitets metoder på game manageren. Her opstod dog en ny problemstiling. Den knap der skal kaste terningerne skal kalde game manageren som ændrer terninge værdierne med en ThrowDIces metode. Problemet er bare at man ikke kan kalde en metode direkte i GUI elementerFor når man kalder en metode fra OnGUi bliver den kaldt flere gange per frame. Det resulterede i at jeg selvom jeg ikke engang trykkede på knappen, skiftede terningerne billede flere gange i sekundet. Jeg har lige i sidste øjeblik fundet ud af at man skal bruge metoden med et Event fra GUI også skulle det virke. Men jeg har ikke nået at teste det endnu.

### Arbejdet lidt med trello i slutfasen

I de sidste par uger af projektet fandt jeg ud af at de opgaver jeg skulle løse var for store, altså jeg lavede for mange ting på engang. Et eksempel var at jeg lavede alle scripts og kodede funktionalitet på engang. Det viste sig så at når jeg så skulle teste det og det ikke virkede, havde jeg svært ved at finde fejlene fordi jeg havde lavet så meget uden at teste. Her begyndte jeg så at bruge trello. Så lavede jeg en lille opgave, som jeg skulle have færdig og testet inden jeg kunne gå videre med den næste opgave. En af de opgaver var for eks. Opret game objekt på game manageren og udskriv en værdi på GUI. Dette gjorde at min fejlfinding var væsentlig forbedret og der var kortere imellem succes oplevelser.

### Elaboration - Vurdering af arbejds – og læringsprocessen

Den måde jeg lærer bedst på er ved at prøve ting af, altså learning by doing. Og det fungerer fint når jeg skal lære at bruge et nyt værktøj som Unity. Det som der er det gode ved at gøre det på den måde i denne sammenhæng er at jeg får en masse erfaring med værktøjet, som sidder fast på rygraden og som virkelig er noget erfaring som jeg kan bruge fremover til at udvikle videre på spillet og evt. flere spil i fremtiden. Det som jeg kan notere mig fra arbejds- processen er at fordi at det er ukendt stof som jeg arbejder med, altså Unity. Skulle jeg have været bedre til at begrænse projektet fra start af, i stedet for at sætte urealistiske mål i forhold til at man skal udvikle et spil med et nyt værktøj som man ingen erfaring har i. der ud over skulle jeg så have været bedre til at stoppe op også fået lavet en ny planlægning, så hurtigt som muligt, efterhånden som man finder ud af at ens mål er urealistiske. Måske kunne jeg også have overvejet at konferere med en som har erfaring med et ukendt værktøj, inden jeg sætter mine mål for det jeg skal arbejde med. Så ville jeg måske have haft et bedre udgangspunkt til at sætte de rigtige mål og rent faktisk nå at gennemføre dem alle.

Ulempen ved learning by doing metoden er at man ikke ved hvor lang tid tingene tager når det er ukendt stof man arbejder med. Dog ville jeg kunne hjælpe mig selv en hel del ved at lave mindre delmål, så der bliver kortere mellem succes historierne. Som jeg gjorde i den afsluttende periode, ved hjælp af trello, hvor man kan sætte de delmål op.

### Elaboration - Refleksion af eget udbytte

Det som jeg har fået ud af arbejdet med spillet, er en masse erfaring i at bruge Unity, som var et af mine vigtigste personlige mål for det her projekt. Få så meget erfaring som overhovedet muligt med Unity, så jeg er godt klædt på til fremtidige spil projekter. Lige på det punkt synes jeg bestemt at jeg er kommet i mål, i og med at jeg har prøvet så mange forskellige arbejds metoder af i Unity. Har jeg nu en god forståelse af hvordan man kan lave en arkitektur i Unity, som er nem at vedligholde og udvide på. Og det er i min bog mange point værd. For hvis man kan lave et simpelt program der er nemt at udvide og vedligeholde. Er man et skridt videre til at kunne lave større og mere krævende projekter som skal håndtere mange flere objekter.

Jeg har fået den erfaring at når man skal arbejde med et ukendt værktøj som unity, skal jeg være meget mere skarp på at få sat nogle realistiske mål op for projektet. Dette kan jeg kan jeg bruge hvis jeg skal arbejde med et ukendt værktøj igen. For så kan jeg måske gøre noget aktivt for at finde ud hvilke mål der ville være realistiske at stille op for projektet. Det kunne være for eks. At snakke med en som kender og har erfaring med det ukendte værktøj. I forhold til planlægningen af elaboration fasen fandt jeg ud af jeg lavede for store komplekse ting på en gang og en anden gang jeg skal arbejde med nye værktøjer vil jeg starte med at lave små opgaver som virker, også fælge op på dem med det samme i den overordnede plan.

### Elaboration – Konklusion

Her i slutningen af elaboration fasen kan jeg konkludere at for at lave et spil i Unity kræver det meget erfaring med at bruge det og stor indsigt i alle programmets funktioner. Samtidig skal man også have styr på hvordan man bygger arkitekturen op afhængig af hvilken type spil det er man vil udvikle. Når man udvikler et spil i et ukendt udviklings miljø er det vigtig at opstille små og simple delmål, for bedre at kunne løse større komplekse problemer og fordi at det er motiverende for den videre udvikling.

## Konklusion