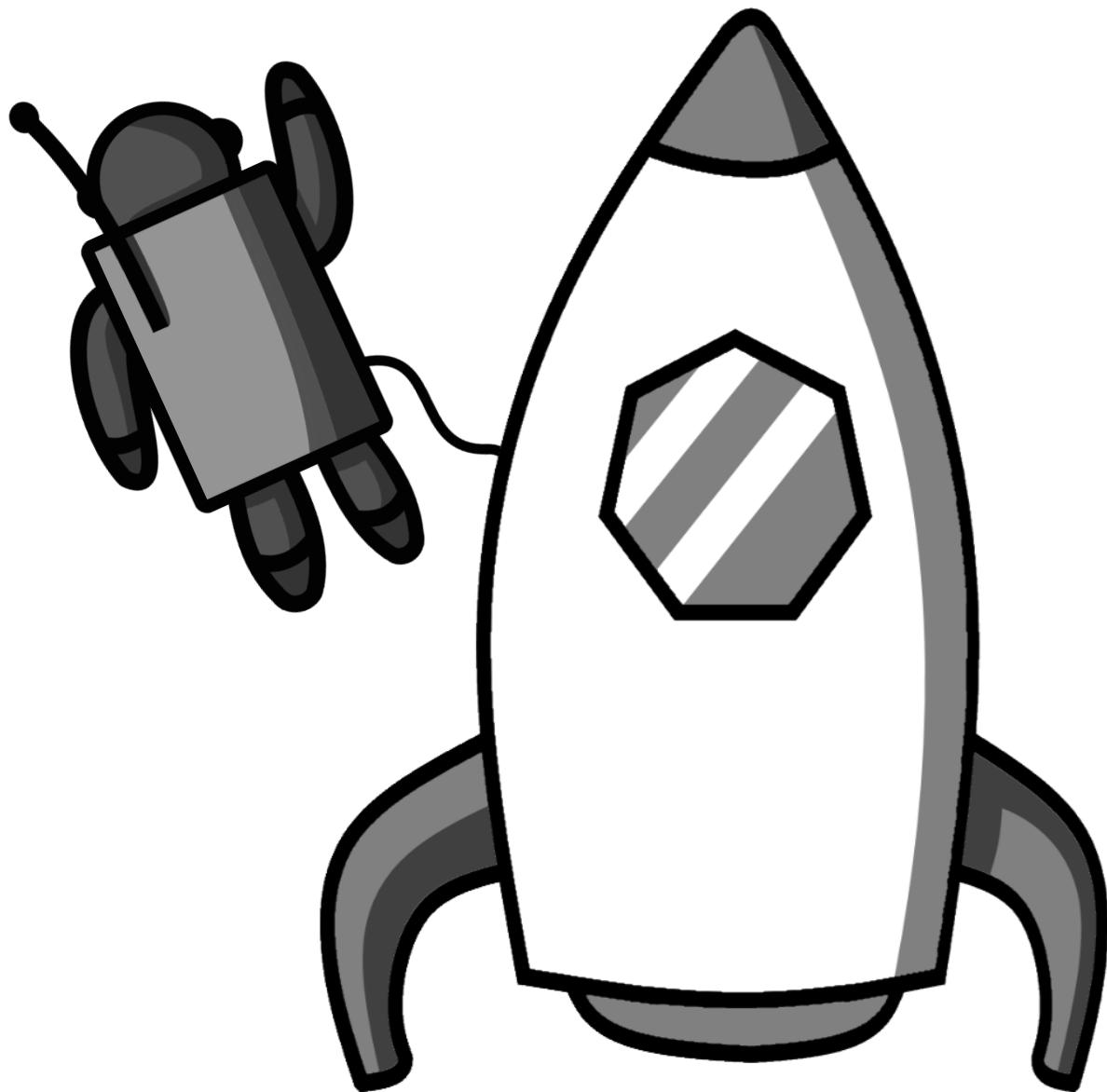


# **Matematik spil til folkeskolen**



**Eksamensprojekt i Teknik A  
Af Hans Heje, 3.z 15-5-2019**

**Lærer: Mads Søgaard og Tobias Dahlsgaard Larsen**

## Abstract

This paper explores why the Danish school system underperforms compared to other nations, and the future implications associated with the lack of motivation to get better test results in math. More precisely it is explored why school children in the early grades have a lack of motivation and interest in mathematical sciences, and what could be a solution to this problem. To help solve this problem there has been devised a product in form of a game composed of more than 1000 lines of code to make math more fun. Furthermore, every step of the process from the conception of said game to the final game is documented, and explained in detail. In conclusion, the final product certainly helps school kids get better at math, whilst making the subject considerably more engaging and fun.

# Indholdsfortegnelse

Abstract	2
Indledning	4
Nøgleproblem	4
Problemanalyse	4
Problemformulering	12
Projektafgrænsning	12
Spilteori	15
Hvad er et spil?	15
Motivationsteori	16
Interaktionsdesign	17
MVP (Minimum Viable Product)	18
HCG (Hyper Casual Games)	19
Arbejdsspørgsmål	20
Designkrav	20
Løsningsforslag	21
P/V skema	21
Implementeringsfase	22
Første iteration	23
Anden iteration	26
Tredje iteration	29
Postremum testfase	33
Resultater og data behandling	34
Tidsplan	37
Diskussion	35
Konklusion	36
Perspektivering	36
Litteraturliste	38
Bilag	40

## Indledning

I forhold til østasiatiske lande som Kina og Hongkong kommer de danske skoleelever fagligt længere og længere bagud. Det skyldes til dels mangelende motivation i den danske folkeskole til at øve sig i diverse fag, dels for stillestående undervisningen. (Jacobsen, 2013)

Matematiske kompetencer bliver vigtigere og vigtigere, og fremtiden vil blive mere og mere baseret på matematik og matematiske kundskaber. Fremtidens arbejdsmarked efterlyser flere og flere programmører, personer der kan arbejde kvalificeret med statistik, og som generelt kan anvende matematik i avancerede sammenhænge. Så hvis der ikke findes medarbejdere, der er dygtige til matematik, så vil det danske velfærdssamfund muligvis have et problem. (TheTopTens, 2019)

Det ses også ud af de 20 uddannelser som giver højst løn: 17 ud af 20 uddannelser kræver matematik på højt niveau. (Nygaard, 2018)

Det er ikke bare ude på jobmarkedet, at de matematiske kompetencer er vigtige. Det er også i dagligdagen. Alle forhold i livet involvere matematik i et eller andet omfang.

Derfor er det vigtigt at elever ikke falder bagud, og som minimum forstår de mest basale begreber i matematik, og er i stand til at benytte dem (Pi day, 2018)

Derfor har jeg valgt at udvikle et produkt, der på én gang kan styrke eleverne fagligt og løse problemet vedrørende skolelevernes manglende motivation.

## Nøgleproblem

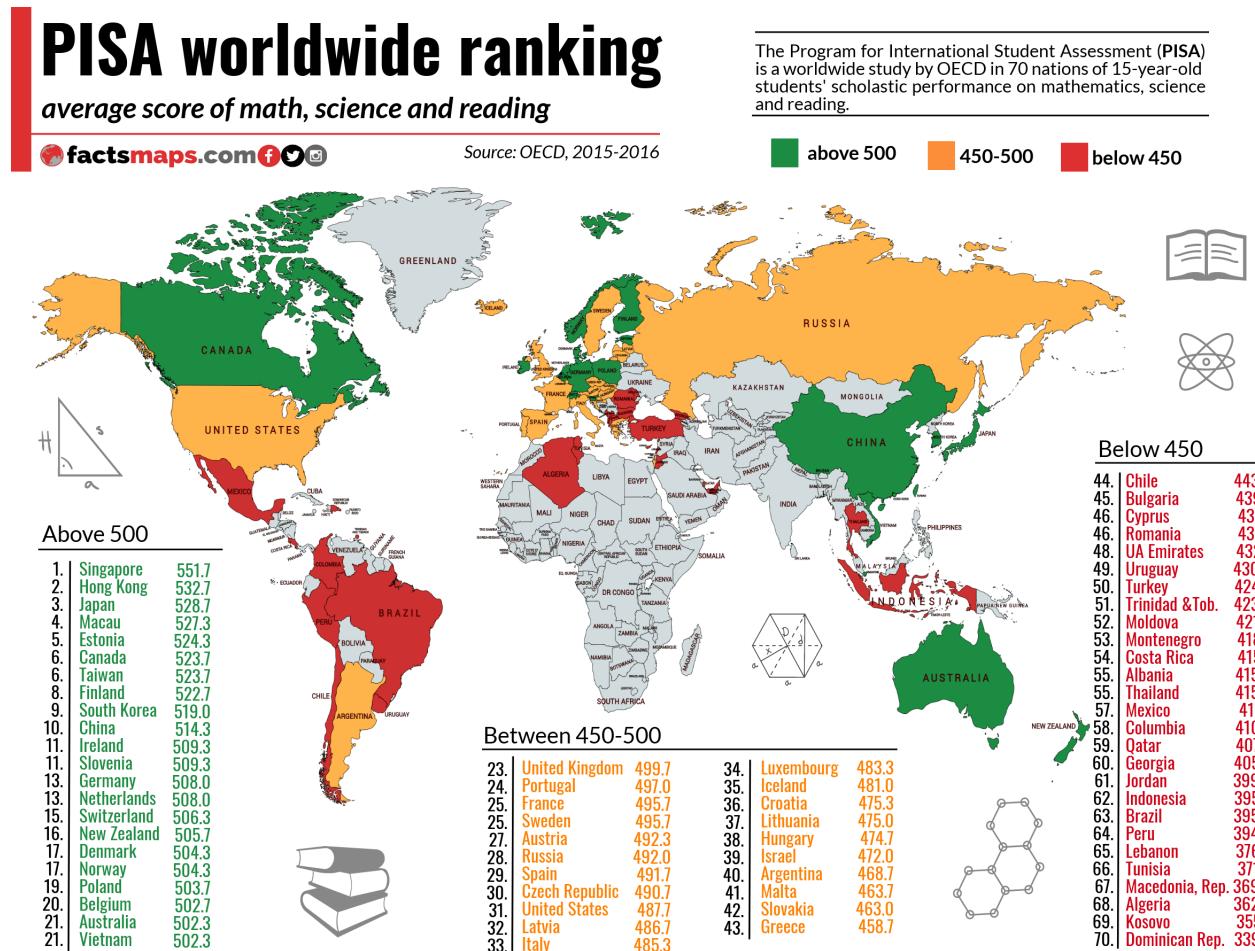
Det er et problem, at børn og unge mangler motivation til læring af basale faglige kompetencer.

## Problemanalyse

Danske elever kommer længere og længere bagud i forhold til andre østasiatiske lande.

Det bliver eksemplificeret i DRs dokumentarserie "9.z mod Kina" hvor danske skoleelever får faglige tæsk i alle de forskellige konkurrencer. (Jacobsen, 2013)

Ifølge PISA (The Program for international Student Assessment), så ligger Kina på en tiende plads, og lande som Singapore, Hong Kong og Japan er i top tre, mens Danmark relativt set ryger længere ned og nu er på en 17 plads. Figur 1 nedenunder visualisere de forskellige test resultater.



Figur 1, (PISA, 2016)

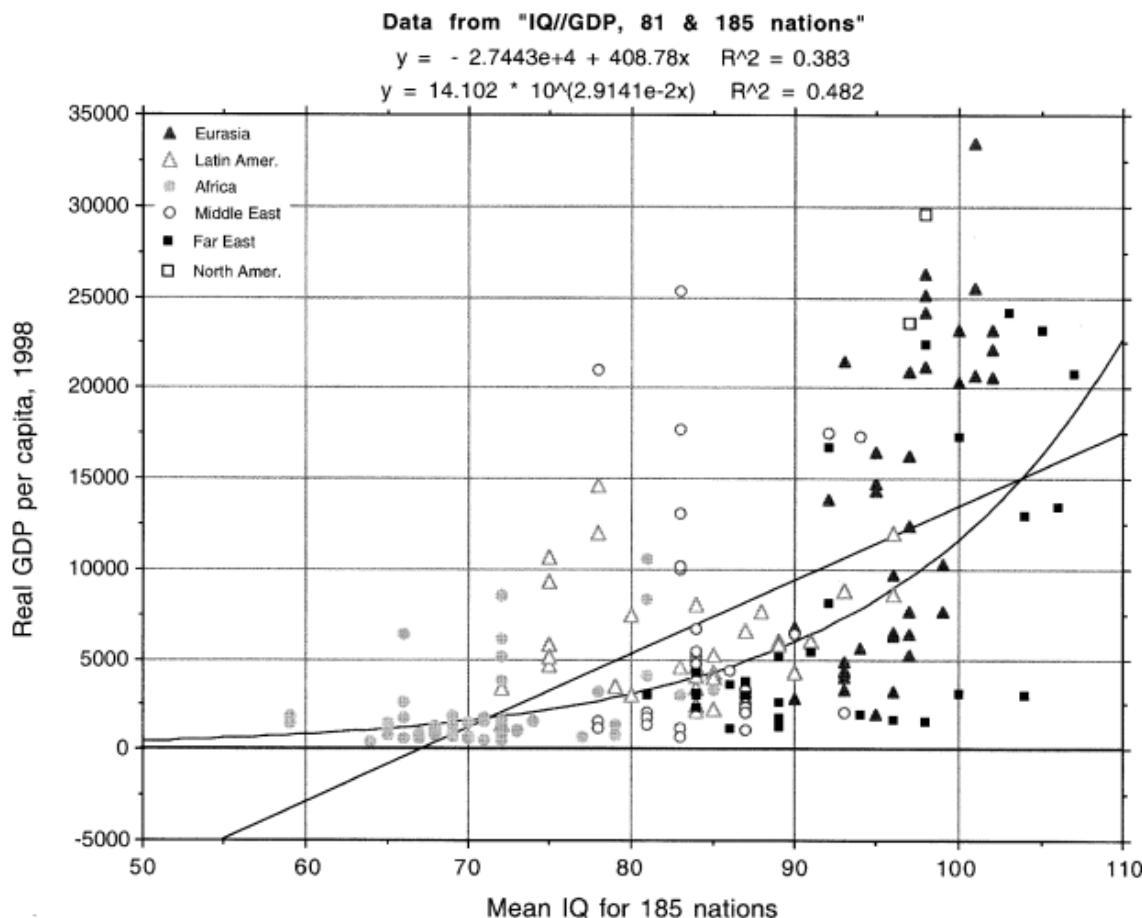
En af grundene til, at de danske folkeskoleelever bliver gennemtæsket fagligt af kineserne, er blandt andet manglen på motivation i den danske folkeskole. Mens de kinesiske skole elever er rigeligt motiverede i kraft af forældrepres, og høje forventninger om at ens børn præstationer, så er kulturen i Danmark anderledes. I

Danmark er en af vores vigtigste kulturelle værdier friheden til at gøre hvad vi vil, og dermed også forældrenes uforbeholdne og ubetingede støtte til vores beslutninger.

Danske Skoleelevers Formand Agneta Vienberg Hansen mener, at danske skole elever mangler motivation i timerne. Dog skal vi ikke kopiere den frygtbaserede og rigide kinesiske mentalitet der baserer viden på udenadslære og ikke kreative kompetencer. I stedet mener hun, at vores motivation skal komme fra et andet sted, så vi kan blive dygtigere, uden at vi mister vores danskhed. Hun foreslår blandt andet, at vi i Danmark kombinere det faglige med noget praktisk, så læringen bliver mere håndgribelig.

Desuden skal ambitionsniveauet og motivationsniveauet løftes ved eksempelvis at gøre det at lære sjovere og mere spændende. (Jacobsen, 2013)

Motivationsfaldet kan også ses af vores gennemsnitlige IQ. Kinesernes gennemsnits IQ er på 105, mens Danskerne er på 98 (Brainstats, 2019). Godt nok er der ikke nødvendigvis en relation mellem IQ og hvor meget arbejde vi lægger i de diverse opgaver, men hvor kun de få klogeste i Kina får undervisning, så får alle i Danmark undervisning. Det betyder så, at når vi tager sådan IQ test, så rangerer vi lavere. Det skal ses som en fordel, da der mange der rent faktisk får undervisning i Danmark. Men det betyder også, at den traditionelle tavle undervisning ikke er for alle, da nogen har svært ved at få et intellektuelt udbytte af den. Når alt kommer til alt, så er IQ en måde at måle hvor god du er til at løse problemer. Så hvis du ikke lærer disse værktøjer i skolen, så kan det blive et problem for den enkeltes fremtid. Der er en videnskabelig artikel lavet af Richard E. Dickerson, som konkluderer at der er en mindre men betydningsfuld korrelation mellem et lands gennemsnits BNP og det samme lands gennemsnits IQ (se figur 2, nedenunder).



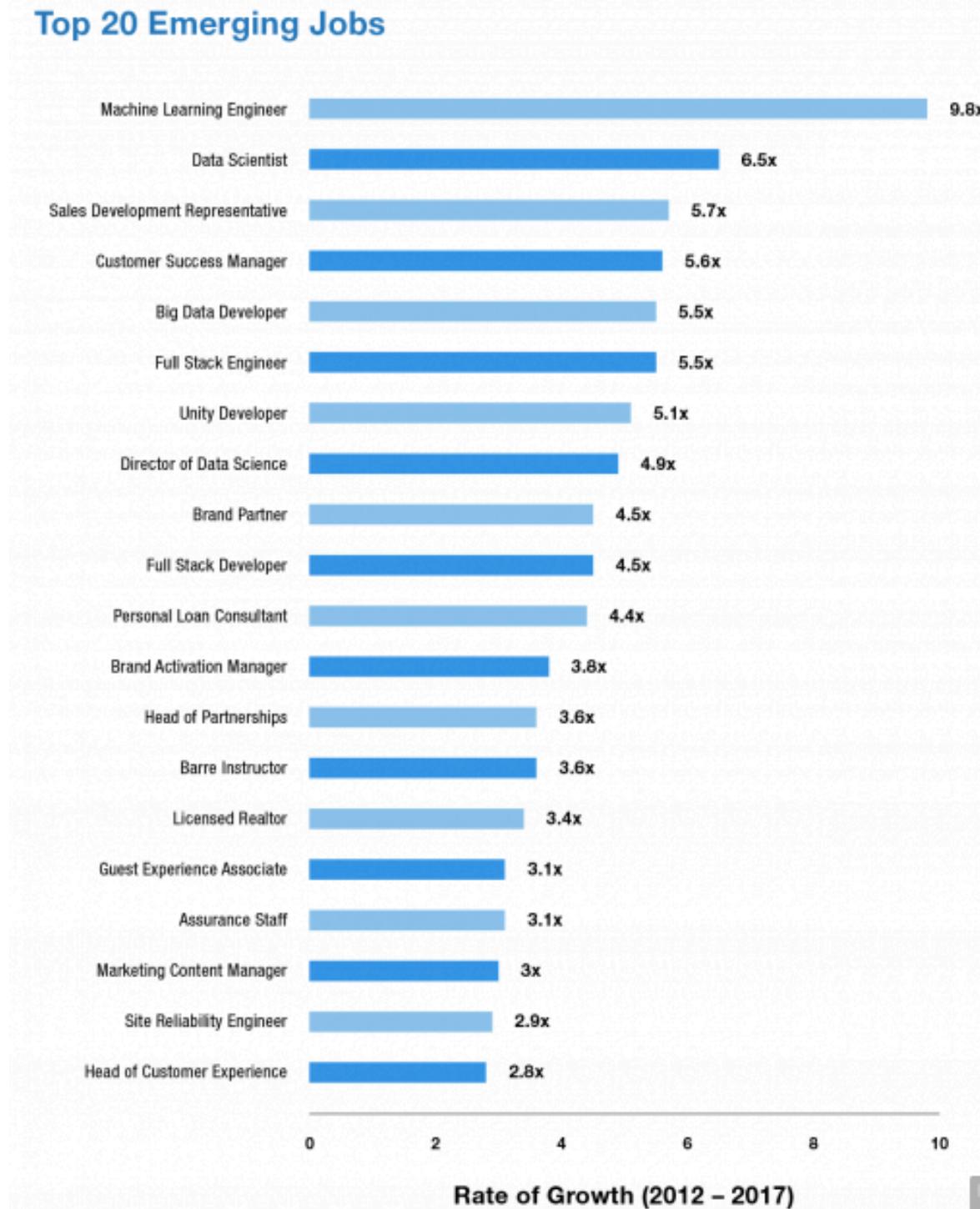
Figur 2 (Dickerson, 2005, s. 292-294)

Hvis vi fortsat skal vokse som nation, både økonomisk og intellektuelt, så er det vigtigt, at alle i skolen får det rigtige udbytte og dermed de rigtige værktøjer. Det betyder også at elever skal motiveres mere af undervisningen, i stedet for bare at døse hen i ren kedsomhed.

De danske myndigheder erkender da også, at det er vigtigt at løfte skoleelevernes faglige niveau. De har nemlig indført at ens gennemsnit skal være minimum på karakteren 5 for at kunne søge ind på et gymnasie. Så i elevernes fremtids navn skal de have gode faglige kompetencer, og ikke bare være ligeglade eller kede sig ihjel over undervisningen.  
 (Undervisnings Ministeriet, 2019)

Særligt matematik bliver vigtigere og vigtigere i nutidens verden. Derfor skal eleverne motiveres til at lære og forstå den matematiske verden. Over halvdelen af de hurtigst

voksende jobmarkeder involverer enten basis eller avanceret matematik (se figur 3, nedenunder).



Figur 3, (Stensdal, 2017)

Men selv hvis matematik ikke er elevernes primære interesse, så kræver næsten alle professioner en god forståelse for matematik. Selv de mest simple aktiviteter, som at

bage, eller at betale nede i Netto, fordrer en fundamental matematisk forståelse. Mennesker som er i stand til at benytte sig af simpel matematik, har nemmere ved at komme til tiden og planlægge ting, i det de kan forudse hvor lang tid de behøver til alle opgaverne for at nå det hele. Der ud over viser det sig at mennesker med en god matematisk forståelse normalt har nemmere ved at holde styr på privatøkonomien. Ifølge forskning udarbejdet af dr. Tanya Evans fra Stanford University, har mennesker med en god matematisk forståelse nemmere ved at tage beslutninger og løse problemer, end mennesker som ikke kan finde ud af matematik. (Pi day, 2018) Det er derfor utroligt vigtigt, at matematik ikke bliver et fag for hvilket skoleeleverne mister motivationen.

Især for drengene gælder at deres motivation falder år for år. Folkeskolen er skruet sammen til at gavne piger, ikke drenge. Det betyder, at drengene ikke føler sig grebet eller motiveret. Ifølge rapporten ”piger og drenge i skolen” af Niels Egelund, så falder drengene tilbage i de teoretiske fag, der er præget af tavle undervisning. Hans konklusionen er, at skoler som har flere håndgribelige opgaver i undervisningen skaber mere motiverede og dygtigere drenge. Det kan være alt fra at lave et eksperiment til at lave noget hvor de selv kan prøve sig frem.

(Aisinger, 2019)

Fordi drengene udgør halvdelen af befolkningen og der ved arbejdsmarkedet er der et alvorligt problem. Så er det skidt, hvis halvdelen af vores befolkning mangler motivation og derved falder bagud. Et studie lavet af ”The Office for National Statistics” konkluderer at jo højere ens uddannelse er, jo mere tilfreds og glad er man i hverdagen. Derfor er det imperativt at mennesker ikke dropper ud lige efter folkeskolen, grundet manglende motivation. Det giver nemlig gennemsnitligt mindre glade borgere.

(Bingham, 2012)

Så hvis Danmark som land også i fremtiden skal opleve vækst og mere tilfredshedude på arbejdsmarkedet, så er det vigtigt at elever ikke falder bag ud allerede i folkeskolen.

Der findes fem grundlæggende måder at motivere eleverne på:

1. Vidensmotivation: Det er når eleverne naturligt bliver motiveret af ren intellektuel interesse i faget, og får lyst til at lære mere. Eksempelvis hvis du er en

stor bogorm, så vil du nok finde naturlig interesse for litteratur delen af dansk timerne.

2. Mestringsmotivation: Det er når eleverne kan løse en opgave og får lyst til at løse flere opgaver, fordi det giver tilfredsstillelse.
3. Præstationsmotivation: Det er når eleverne oplever at en ekstra indsats giver pote, fordi de derved får ros eller højere karakterer. Det medfører at de får lyst til at præstere mere, så de kan se deres karakterer blive højere eller tilsvarende.
4. Relationsmotivation: Det er når eleven føler at være en del af det sociale aspekt i klassen. Altså oplevelsen af at læreren ser og hører efter vedkommende.
5. Involveringsmotivation: Det er når eleverne er med til at bestemme emner i undervisningen. Det kunne være igennem åbne læreprocesser eller gennem projektopgaver hvor der er mere frihed.

(Barse, 2015)

For at styrke elevernes præstation i timerne så skal vi helst kunne opfylde alle 5 måder mest muligt. Hvis du derfor ikke har vidensmotivation kan du stadig få relationsmotivation og så videre. Nærmest som et sikkerhedsnet for motivation i timerne. Relationsmotivation kommer sædvanligvis naturligt i skolen hvis der er et godt miljø. Vidensmotivation kommer naturligt hvis læreren er interessant, eller hvis emnet fascinere eleven. Involveringsmotivation kommer også ved at tale med læreren. Men sådan noget som mestringsmotivation og præstationsmotivation, kommer ofte ikke naturligt i folkeskolen. Hvis læreren ikke kan motivere eleven på nogle af de andre måder, så skal der findes en alternativ måde at motivere eleven på. Det kunne være igennem spil.

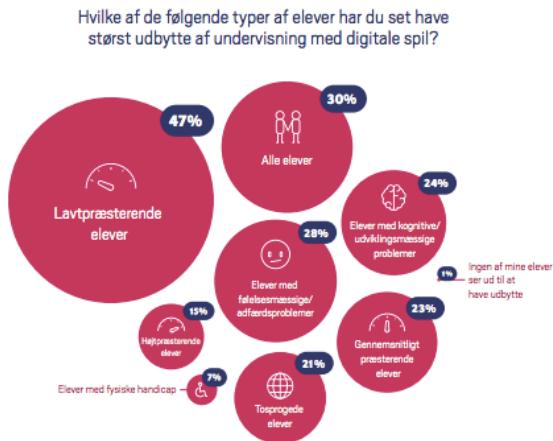
Spil kombinerer læring med noget praktisk, der er håndgribeligt for eleven. Spil har været anvendt gennem tusinder af år til at undervise personer. Tilbage i jernalderen blev skak opfundet som en måde til at stimulere hjernen på. Det gjorde skak fordi det gav en bedre forståelse for feudalsamfundet. Hvis kongen falder, så taber du spillet. Det samme er sandt for feudalsamfundet. Du har bønnerne Forrest, fordi det er sikkerhedsnettet og grundlaget for feudalsamfundet. Biskop Wibold af Cambrai opfandt også et spil, til læring om katolicismens syv dyder. Og så videre igennem historien har spil været en naturlig del af læring. Andreas Lieberoth har 7 pædagogiske pointer, når det kommer til

spil. Blot for at nævne de mest essentielle: Lysten til at spille ligger naturligt i mennesket, og naturlig motivation er nok den mest effektive måde til at motivere eleverne på.

Man husker læring og erfaring fra spil langt bedre når det bruges som et opbrud i den traditionelle undervisning. Eksempelvis ved at hver time afsluttes med 15 minutters spil. Læreren skal trække eleverne ind i spillet og bruge det som værktøj. Det skal ikke overtage undervisningen, men gøre læringen mere interessant og intuitiv.

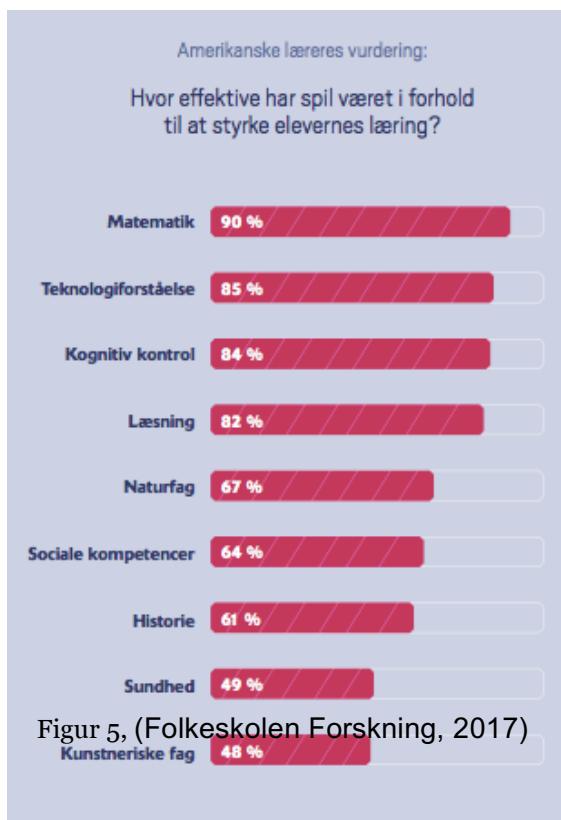
Læreren skal sætte faglige mål og rammer, når det kommer til brugen af spil, så der er et konkret mål ved det. Den nemmeste måde at få gavn af spil, er i gennem det som hedder

”gamification”. Det er hvor man tager noget kedeligt, som addition og subtraktion, og gør det til et kompetitivt spil. Det kan eksempelvis gøres gennem high-scores, eller ved man kan vinde hvis man er god nok.  
(Folkeskolen Forskning, 2017)



Som der kan ses i figur 4, så er det de lavt præsterende elever, som har haft størst gavn

Figur 4, (Folkeskolen Forskning, 2017)



stimulere nysgerrigheden.

(Folkeskolen Forskning, 2017)

af spil. I 47% af tilfældene er det dem, der præsterende elever, der har fået størst udbytte af spil i undervisningen. Altså de elever der ikke tidligere har været motiverede til at lære noget. Samfundet er kun så godt, som de dårligste er. Hvis vi igennem spil kan få de fagligt mest inkompentente til at få et højere niveau, så vil resten af samfundet blive bedre som helhed. Det er særligt effektivt i fag, som matematik, hvor hele 90% af tiden har spil kunne styrke elevernes læring (se figur 5). Psykologen Thomas W. Malone, har igennem nogle eksperimenter konstateret, at den mest effektive måde til at styrke elevernes motivation gennem spil, er gennem udfordrende spil, fantasifulde spil eller ved at

Altså skal spil ikke erstatte den almindelige undervisning, men blot supplere som motiverende faktor.

## Problemformulering

Kan spil bruges som motivation hos elever i folkeskolen til at præstere bedre i de matematiske fag.

## Projektafgrænsning

Jeg vil konstruere og programmere et spil, hvor man kan blive bedre til matematik. Målgruppen som er i fokus, er 3-6 klasses elever i folkeskolen. Mere bestemt er det

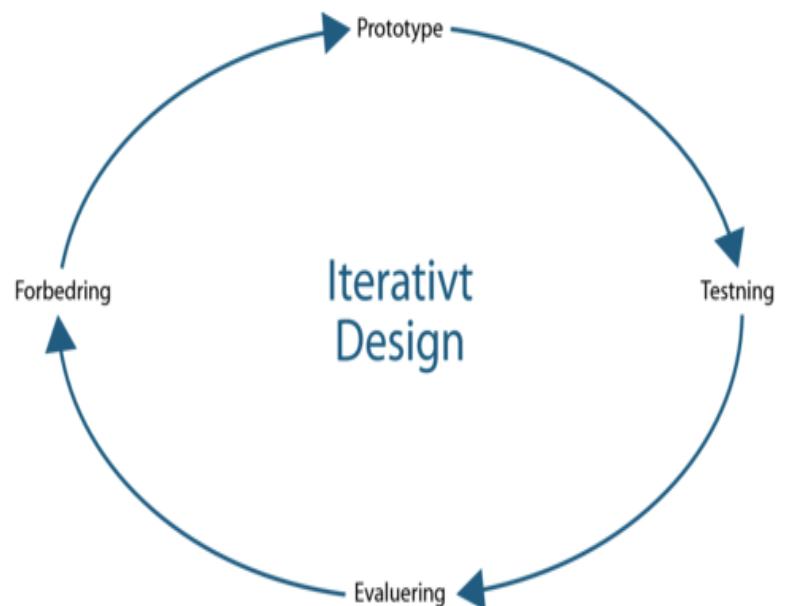
deres kompetencer indenfor tal og algebra som er i fokus. EMUs fagmål inden for emnet ”tal og algebra” i faget matematik fordrer, at efter 3 klasse skal eleverne kunne regne med naturlige tal. Efter 6 klasse skal de kunne benytte rationelle tal i forhold til beregninger og beskrivelser. Det inkludere sådan noget som at konvertere kommatal til procent. Hvis jeg har nok tid til det vil jeg også lave en bane til elever som lige er blevet færdige eller i gang med 9 klasse. Ifølge EMU skal elever efter 9 klasse kunne løse simpelt algebra. (Undervisnings Ministeriet, 2016)

Motivationsfaktoren skal primært være baseret på mestringsmotivation, eftersom spillet skal starte med at være meget nemt, og jo mere du spiller, jo svære bliver det. Desuden skal spillet løbe hurtigere og hurtigere, så man virkelig føler, det bliver sværere og sværere, så man kan blive udfordret. Jeg vil prøve at lave spillet, og derefter efterteste det, så der er en god balance mellem udfordrende og straffende. Spillet skal ikke føles som om man bliver straffet, men bare at det bliver lidt mere udfordrende. Så hver gang du fejler skal have følelsen af at du var tæt på at klare det. (Extra-Credits, 2013)

Jeg vil også skabe noget præstationsmotivation hos spilleren. Det vil jeg gøre igennem et high-score system så når du engang dør/fejler, får du en karakter for hvor god du var. Det vil jeg gøre for at få spillet til at følelse mere jordnært.

Mit MVP (Minimum Viable Product) kommer til at være noget hvor svaret står der, og du så på en eller anden måde skal finde den rigtige ligning som er lig med resultatet, inden for en hvis tidsramme.

Hvis jeg har nok tid vil jeg gerne implementere musik og forskellige sværhedsgader. Men eftersom jeg er helt alene om projektet, så kan det godt være at det ikke er muligt for mig, at skabe mere end MVP'en. Jeg vil benytte Unity til at programmere og designe spillet i. Jeg vil benytte det program fordi det er det eneste program som jeg



har lidt kendskab til, samtidigt med at de har en god asset store. Hvis der er nogle ting jeg ikke selv kan nå at lave på grund af tidspres, så vil jeg prøve at benytte ”Unity Asset Store”. Der kan man installere små præprogrammerede funktioner. Som musik - hvis jeg når til den del - vil jeg prøve at lave et beat i programmet Fruity Loops Studio. Det skal helst være noget musik som holder tempoet oppe, så man hele tiden føler sig engageret i spillet. Som design vil jeg prøve at bruge kontrastfarver. Det vil jeg gøre for at tingene er nemmere at se forskel på, og for at lave et mere simpelt design baseret på princippet om ”Less is more”. Samtidigt kan det også spare mig en del tid i design delen. Hvis jeg på en eller anden måde når alle mine mål, så vil jeg prøve at få spillet konverteret til mobilen så det kan spilles lige meget hvor du er.

Jeg vil under skabelsen af dette produkt benytte mig af en metode som kaldes Iterativ design (se figur 6, til højre). Det er hvor man starter med en prototype, som man hurtigt derefter tester. Med udgangspunkt i testresultaterne lavers en evaluering af feedback. Ved hjælp af feedback forbedres produktet. Det er mit mål at gøre det hver uge så jeg undgår overhovedet at komme til at lave for store fejl, uden at jeg kan nå at rette op på det. (Møller, 2017, s. 142)

*Figur 6, (Møller, 2017, s. 142)*

Efter jeg har lavet spillet vil jeg efterteste spillet i en folkeskole, og igennem en kvalitativ undersøgelse se om spillet er motiverende. Dertil kommer, at jeg til sidst vil foretage en kvantitativ empirisk undersøgelse hvor jeg ser om de er blevet bedre til matematik. Det vil jeg gøre ved at bede dem besvare x-antal matematiske spørgsmål før De spiller, og de samme spørgsmål bagefter. Ud fra det vil jeg konkludere om spillet både er lærerigt og motiverende eller kun det ene, eller måske slet ingen af dem. Jeg vil også undersøge om spillet gavner drenge eller piger mest. Ud fra det vil jeg konkludere, om mit spil er et værktøj som eleverne i folkeskolen reelt kan benytte og få gavn af.

# Spilteori

## Hvad er et spil?

Spil kan deles op i 2 overordnede kategorier: Ludologi og narratologi.

Narratologi er spil som påtager sig en fortælling. Spilleren har kontrol over hvad han/hun gør og hvad der sker, men ikke desto mindre er der en konkret fortælling. I det der er en konkret fortælling, så kan sådan et spil også analyseres og fortolkes.

Ludologi derimod er det almene studie af spil. Det er altså ikke kun digitale computer spil der er tale om, men også brætspil og sociale spil. Ludologi fokuserer mere vedrørende spillets regler og struktur, på hvordan spillene spilles og den kultur der er rundt om spillet. Eksempelvis har Black Jack meget simple regler, men det har en ludomansk kultur rundt om sig. Eller et spil som Minecraft hvor der egentlig ikke er nogen regler, så spillerne selv danner et overordnet regelsæt, fra spillerbase til spillerbase.

Ludologiske spil behøver derfor ikke en konkret fortælling og spillet kan stå alene som produkt. Det er den type spil, jeg primært vil fokusere på.

(Møller, 2017, s. 134)

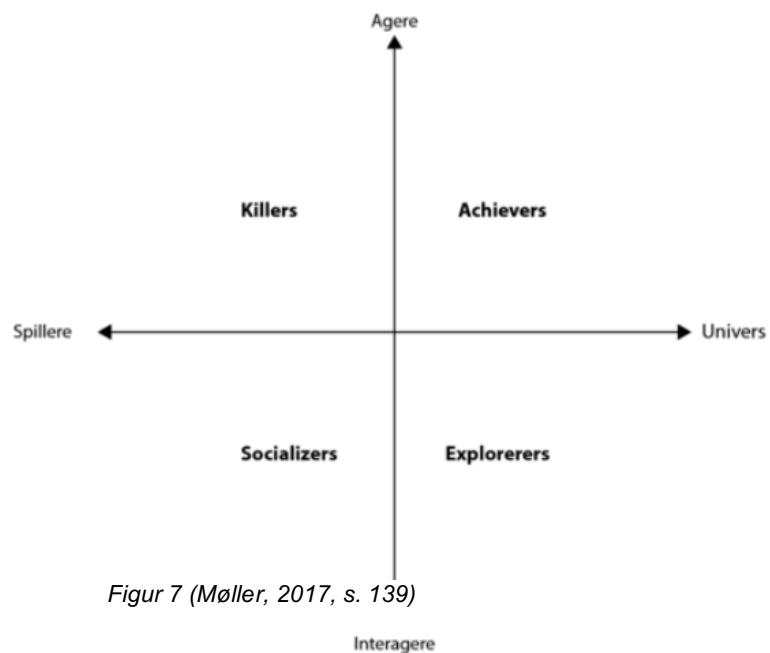
Disse to grundlæggende former for spil kan oversættes og bruges i andre sammenhænge. Eksempelvis kan de bruges i undervisningen til at motionere eller som reklame for et firmas produkter. Dette begreb kaldes ”gamification”. Før gamification kan være brugbart i eksempelvis en undervisningsmæssig sammenhæng, så skal spille mekanikken være i orden så spilleren motiveres til at spille. (Møller, 2017, s. 178)

## Motivationsteori

Når det kommer til motivation inden for spille verdenen, så er det vidt forskellige ting som forskellige individer finder motiverende. Den britiske spilleforsker Richard Bartle inddeler spilleren i fire forskellige kategorier.

De fleste spillere finder elementer fra flere af disse kategorier motiverende, men på samme tid så er man mest tilbøjelig til at hælde imod en af kategorierne.

Bartle inddeler spillerene ind i killers, achievers, socializers og exploreres (se figur 7 til højre).



Figur 7 (Møller, 2017, s. 139)

Killers opsøger konstant kampen. De vil hele tiden eliminere modstanderen. Hvis der ikke eksistere nogen kamp i spillet, så vil de prøve at fremprovokere det, så længe der er en mekanisme der tillader det. Det kunne være i Call of Duty hvor man skyder andre, eller i et byggespil som har en dødsmekanisme. Det er vigtigt for dem at de kæmper mod andre spillere, og ikke bare imod computeren. Ellers er der ikke det samme feedback eller tilfredsstillelse.

Så er der socializers. Socializers vil gerne opsøge det større fællesskab. For dem er spillets mekanismer ligegyldige, så længe der eksistere et godt fællesskab. Det kan være dem som ofte spiller RPG (Role Playing Games), fordi det vigtigste i de spil er selve rollespillet.

Explorers er den næstesidste kategori. Explorers elsker at udforske spillet. De er den digitale verdens opdagelsesrejsende. De bliver motiverede af at opdage nyt land, og finde skjulte områder eller funktioner; såkaldte "Easter Eggs". Deres interesse bliver

primært vækket af ting som ikke er tydelige fra starten, eksempelvis en spille mekanisme der kan udnyttes til at gøre sjove og skøre ting. Eller en sjov reference til noget som ikke har noget med spillet at gøre.

Så er der den sidste kategori: Achievers. Achievers finder deres primære motivation, ved at opnå ting. De kan godt lide at få belønninger i spillet eller at opnå nye highscores. De kan godt lide at blive belønnet, når de gør tingene bedre. De er interesseret i at sammenligne sig selv med andre og konkurrere med andre. De stræber som regel altid efter at være de bedste. At have det bedste udstyr i en MMORPG (Massively Multiplayer Online Role Playing Game), at have den bedste rang osv. Det er fortrinsvis denne kategori som jeg vil fokusere på, når jeg designer mit spil.

(Møller, 2017, s. 139)

Et spil som kan inkorporere elementer fra flest muligt af disse kategorier kan potentielt set have den største brugerbase. Det er det ideelle scenarie: Alle slags spillere, som installerer spillet, bliver motiverede til at blive ved med at spille.

## Interaktionsdesign

Før at det faktisk er et spil, skal du kunne interagere. For hvis spillet har samme handling og alting sker på samme måde altid, og du ingen indflydelse har, så må det per definition kaldes en animationsfilm, altså en lineær fortælling.

Jakob Nielsen, som er en succesrig interaktions designere, siger der er 5 grundlæggende succes riterier:

1. "Learnability". Hvor nemt er det, at udføre simple ting i spillet, første gang man ser det. Hvor intuitivt er spillet at spille. Er det noget som giver mening med det samme, eller skal spilleren forklares konceptet, før spilleren kan finde ud af det? Ideelt set, så kan spilleren finde ud af det næsten øjeblikkeligt.
2. "Efficiency". Hvor nemt er det for spilleren at overskue alle funktionerne, en gang spilleren har lært spille designet at kende?  
Forstår spilleren alle de ting som spilleren kan gøre for at vinde eller komme videre i spillet? Er det nemt at komme igennem menuerne, eller skal spilleren

klikke 5 gange, før spilleren kan justere lyden eller starte en bane. Spillet behøver ikke at være nemt, det skal bare være overskueligt, og simpelt, så spilleren ikke klikker på en million ting, før spilleren finder ud, hvordan spilleren kan starte spillet.

3. "Memorability". Hvor nemt er det, at genlære spillets design, engang man kommer tilbage til det, efter en lang periode uden at have spillet det. Det burde være relativt intuitivt, at komme i gang med spillet igen. Hvis det er for svært, så kan det være spilleren bare giver op, og går væk igen.
4. "Errors". Hvilke fejl kan spilleren komme til at lave, og hvor svært er det at rette op på. Hvis spilleren bliver straffet for hårdt når spilleren dør så gider spilleren ikke længere at spille. Eller hvis slet knappen er ved siden af indlæs knappen, så spilleren kan komme til at slette flere timer, eller dages arbejde, så er det også for meget.
5. "Satisfaction". Hvor behageligt er designet? Er designet æstetisk, eller er det grimt. Er lyden irriterende, eller er den motiverende eller afslappende. Er spillet indviklet, eller er det brugervenligt. Et indviklet design, med kompliceret funktioner, er ikke lige så behageligt, som et minimalistisk design.

(Møller, 2017, s. 165)

## MVP (Minimum Viable Product)

MVP'en (Minimum Viable Product) er det product eller spil, som man minimalt skal have før det er funktionsdygtigt og kan bruges. Det vil sige meget simpel grafik, måske ingen lyd, hvis det ikke er essentielt. Kun de mest grundlæggende funktioner, som gør spillet muligt, skal være til stede. Det gør det nemmere at teste om spilles mest basale mekanik faktisk er god eller ej. Fordi der ikke er så mange andre elementer, som forstyrre vurderingen af spille idéen. Et eksempel på en MVP, kunne være i Super Mario. MVP'en i det tilfælde er at du kan løbe og hoppe. Alle de andre funktioner, som at du har modstandere, og at du kan samle mønster, er ekstra funktioner oven på MVP'en. MVP'en skal altså være færdig hurtigt muligt, så spillet kan testes og ændres, før de indebærer for store konsekvenser. (York, 2019)

Men i stedet for at fylde en masse ting oven på MVP'en, så er der opstået en hel genre hvor du fokuserer på de mest basale funktioner og spillemekanik. Denne genre kaldes Hyper Casual Games

## HCG (Hyper Casual Games)

I mobil verdenen er HCG (Hyper Casual Games) ikke noget usædvanligt. Det er den perfekte cocktail af spille mekanik. Et spil som alle intuitivt vil kunne spille uden så mange instruktioner. Med et simpelt soundtrack som gør at spillet er afslappende, med et behageligt simpelt design. En blanding af Richard Bartles 4 kategorier, for at holde spilleren fanget og motiveret til at blive ved. Næsten alle spil i mobilspille verdenen er til en hvis grad et HCG spil.

HCG spil kræver normalt minimalt til intet input fra spilleren. Derfor kan det spilles og afsluttes overalt uden de større problemer, selv midt i en samtale.

HCG spil består normalt af en eller flere af disse former for mekanik:

1. Timing mekanik. Hvor du skal klikke i det rigtige sekund, eller klikke det rigtige sted, for at få point. Eksempelvis Guitar Hero, hvor du skal klikke på noderne når de kommer, før din person i Guitar Hero, kan spille akkorderne. Hvorved du så får point.
2. Stak mekanik. Hvor du skal stække objekter oven på hinanden, for eksempelvis at danne et stabilt tårn. Dette kan eksempelvis kombineres med timing mekanik.
3. Udholdenheds mekanik. Hvor ting begynder at gå hurtigere og hurtigere, så det bliver svære, og derved mere spændende, jo længere ind i spillet du er.
4. Tyngde mekanik. Det kan eksempelvis, være spil, hvor din karakter enten falder hurtigere og hurtigere ned, eller langsomt stiger længere og længere op. Hvorved forhindringerne er i vejen for dit fald, prøver at stoppe dig i at komme længere op osv.
5. Undgåelses/swipe mekanik. Det er de spil hvor du simpelt og intuitivt swiper til højre eller venstre på din mobil for at undgå kommende fjender eller for at ramme de gode objekter.

6. Vækst mekanik. Hvor din karakter, eksempelvis en slange, bliver større og større, jo mere du æder. Eller hvor man prøver at sammensætte objekter, for at gøre dem større.
7. Puslespils mekanik. Det er spil hvor du ved logisk tænkning prøver at sammensætte geometriske objekter så de passer sammen.
8. "Idle" mekanik. Spil hvor du ikke gør noget som helst ud over at vente. Det kan eksempelvis være Cookie Clicker, hvor du i spillet køber ting, for at du kan producere mere mad hvert sekund. Hvorved du så venter på, at spillet helt automatisk har genereret nok mad, til at du kan sælge maden, og købe flere ting der producere mad.  
(Kinniburgh, N/A)

## Arbejdsspørgsmål

1. Kan spillet motivere elever i matematik?
2. Forbedrer spillet elevernes matematiske færdigheder, som i addition og subtraktion og simpelt algebra?
3. Er spillet intuitivt nok til at spilleren ikke behøver nogen instruktioner om hvordan det fungerer?
4. Er spillet simpelt og ikke forvirrende?
5. Føles spillet kompetitivt?
6. Ser spillet professionelt ud?

## Designkrav

1. Det er sjovt at spille (vægtning 4/5)
2. Designet af grafikken er simpelt og ikke overflødig (vægtning 2/5)
3. Det skaber en motivations følelse til at blive ved. (vægtning 4/5)
4. Det er lærerigt (vægtning 5/5)
5. Spillet er intuitivt (vægtning 3/5)

## Løsningsforslag

For at komme frem til disse løsningsforslag, har jeg kigget på de mest populære spil og brainstormet på hvordan jeg kunne lave disse spil om til et matematik spil. Det har jeg så skitseret mig frem til, for at visualisere og videretænke på spilleideerne.

1. Geometrispil: Et geometri spil, hvor regnestykket står i midten, og svarene kommer imod midten, og du så skal pege munden mod de rigtige svar, for at få point. (se bilag 1-5)
2. Flowspil på tid: Et spil hvor svaret står i midten, og du så skal navigere dig hurtigst muligt gennem de rigtige svar indtil du er ude af cirklen. Hvis du er for langsom, så løber du ud af tid. (se bilag 6-7)
3. 2048spil: Et spil hvor du skal swipe op, ned, højre eller venstre for at rykke alle brikker den vej. Hvis regnestykket og svaret er lig med hinanden, så får du et point, og hver gang du swiper, så kommer der flere brikker. (se bilag 8)
4. Raketspil: Et spil hvor din figur er i bunden, og du kan swipe til højre og venstre, for at navigere ind i de regnestykker som er lig med dit svar. (se bilag 9-10)

## P/V skema

Hvert spil får en score for hver kategori, og den score ganger jeg så med min vægtning fra designkravene. På den måde kan jeg få en mere fair vurdering af hvilken spilleide er bedst.

Hvor intuitivt spillet er vurdere jeg på, hvor få spille funktioner der er involveret i MVP, og hvor meget spillendes funktioner minder om HCG spil. Ud over det, så vurderer jeg det på Jakob Nielsens trin om "Learnability" og "Efficiency".

Hvor lærerigt spillet er, vurderer jeg på, hvor stor en del af spillet handler om læring. Hvor motiverende spillet er, vurderer jeg på Jakob Nielsens trin om "Errors", og hvor godt det opfylder motivationsteori.

Hvor simpelt spillet er vurdere jeg på grundlag af hvor æstetisk spillet er og hvor mange ting der fylder skærmen generelt. Er der mange overflødige ting, eller ting du skal holde øje med i selve MVP'en, så er spillet ikke simpelt.

Hvor sjovt spillet er, har jeg vurderet ud fra en spørgeundersøgelse af mine forskellige ideer. 7 sagde raketspillet så sjovest ud, 5 sagde geometrispillet var sjovest, 4 sagde flowspillet var sjovest, og kun en sagde 2048spillet var sjovest.

P/V Skema	Sjovt vægt = 4	Simpelt vægt = 2	Motiverende vægt = 4	Lærerigt vægt = 5	Intuitivt vægt = 3	Total:
Geometrispil	4/5 $4*4 = 16$	4/5 $4*2 = 8$	4/5 $4 * 4 = 16$	4/5 $4*5 = 20$	3/5 $3*3 = 9$	69
Flowspil	3/5 $3*4 = 12$	4/5 $4*2 = 8$	4/5 $4 * 4 = 16$	5/5 $5*5 = 25$	4/5 $4*3 = 12$	73
2048spil	1/5 $1*4 = 4$	4/5 $4*2 = 8$	1/5 $1 * 4 = 4$	1,5/5 $1,5*5 = 7,5$	5/5 $5*3 = 15$	38,5
Raketspil	5/5 $5*4 = 20$	5/5 $5*2 = 10$	5/5 $5*4 = 20$	3,5/5 $3,5*5=17,5$	4/5 $4*3 = 12$	79,5

Altså er det raketspillet der har fået flest point og derfor det spil, jeg vil lave.

## Implementeringsfase

Her er en liste af de funktioner jeg gerne skulle have med i det færdige produkt. Det er ikke sikkert jeg når det hele, hvorfor jeg har lavet en rækkefølge fra de vigtigste til de mindst vigtige funktioner i spillet. Det vigtigste er, at jeg få lavet MVP'en færdig:

1. Spilleren og bevægelse (MVP)
2. Regnestykkerne, som kommer i mod spilleren (MVP)
3. Ændring i hvilke regnestykker der kommer (MVP)
4. Point tæller
5. Højde tæller
6. High score gemt
7. Hovedmenu
8. Grafik og animation
9. Baggrund
10. Lyd effekter
11. Musik
12. Flere niveau og baner
13. Indstillinger

14. Udgive det på appstore
15. Forskellige figurer, du kan låse op for

## Første iteration

Jeg laver spillet i programmet Unity. Det vil sige at al kode bliver lavet i sproget C#. Det første jeg startede med, var at hente en model fra Unity assets store, som skulle forstille at være min karakter i spillet (se bilag 11). Selve asset pakken med mit rumskib hedder: "Moduler Space Raider".

Derefter begyndte jeg at programmere spillerens bevægelse, så min karakter kunne bevæge sig til højere og

venstre. Jeg satte

bevægelseshastigheden til 0.2f.

Så implementerede jeg en

funktion, der gjorde at man

ikke kunne bevæge sig ud af skærmen, ved at sætte den

maksimale x værdi til at være mellem -3.5 og 3.5 (linje 15-21

på figur 8, til højre).

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Controls : MonoBehaviour {
6
7      // Use this for initialization
8      void Start () {
9
10     }
11
12     // Update is called once per frame
13     void Update()
14     {
15         if (Input.GetKey("right") && this.transform.position.x < 3.5)
16         {
17             this.transform.Translate(0.2f, 0, 0);
18         }
19         else if (Input.GetKey("left") && this.transform.position.x > -3.5)
20         {
21             this.transform.Translate(-0.2f, 0, 0);
22         }
23     }
24 }
25 }
```

Figur 8

Derpå tegnede jeg baggrunden i programmet paint.net, og lagde det ind i spillet (se bilag

12). Derefter programmerede jeg en scrolling funktion, så baggrunden blev ved med stille og roligt at scrolle fra top til bund, så det gav en effekt af at spilleren bevæger sig fremad, uden at spilleren faktisk gør det (se figur 9, til højre). Det gøres ved funktionen vector2 bliver ændret til at ændre x og y aksen på billedet med en hvis hastighed.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BackgroundScrolling : MonoBehaviour {
6      public Vector2 uvSpeed = new Vector2(0.0f, 0.2f);
7      private Vector2 uvOffset = Vector2.zero;
8
9      // Use this for initialization
10     void LateUpdate ()
11     {
12         uvOffset += (uvSpeed * Time.deltaTime);
13         this.GetComponent<Renderer>().materials[0].
14             SetTextureOffset("_MainTex", uvOffset);
15     }
16 }
17
18 }
19 }
```

Figur 9

Derefter tegnede jeg en ny baggrund, men gjorde denne baggrund en smule transparent ved hjælp af Unitys ”Material alpha transparency”. Jeg lagde dette billede lidt oven over det første billede, så det var første prioritet i synlighed. Derefter satte jeg hastigheden på hvor hurtigt det billede ville scroll ned, så det gav en effekt af, at spillets plan var mere tredimensionelt (se bilag 13).

Derefter tilføjede jeg en box colider til spilleren, så hvis nogen objekter ramte den, så ville jeg kunne registrerer det, i stedet for, objekter i spillet ikke har nogen effekt på spilleren (se bilag 14).

Så lavede jeg en masse bokse som var  $2 \times 2 \times 1$  i størrelsen (se bilag 15). Disse bokse skal være dem, hvor regnestykkerne står på. Så tilføjede jeg et tekstobjekt til hver boks, så man kan se regnestykkerne. Så tilføjede jeg en rigidbody til selve boksene, så de kan blive påvirket af tyngdekraft. I selve projekt indstillings menuen, så satte jeg tyngdekraften til at være hen af z aksen, og satte hastigheden til -0.2. Dette betyder at boksene går mod spilleren ved hjælp af C# tyngdekrafts mekanik (se bilag 16).

Nu har jeg vores spilleobjekt som indehaver svarende. Så nu ville jeg lave et objekt, hvorfra vores kasser kan spawne fra.

Jeg lavede en ny kasse uden for den synlige del af banen, og lavede et script, som spawnede dem ved tilfældige intervaller, ved hjælp af funktionen void Update og Random.Range” (se figur 10, til højre). Ud over det, så bliver boksene spawnet tilfældigt på x aksen, mellem 0.0 og 1.8, for at tallene bliver mere tilfældige.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class spawn : MonoBehaviour {
6      //første bogstav er lig med hvad resultatet er (a=1 b=2), andet og tredje bogstav er lig m
7      public GameObject mat1;
8
9      // Use this for initialization
10     void Start () {
11
12     }
13
14     // Update is called once per frame
15     void Update () {
16         if (Random.Range(0, 1000) < 1)
17         {
18             // Vector3 pos = new Vector3(this.transform.position.x * Random.Range(-1.2f,0.0f),
19             Vector3 pos = new Vector3(this.transform.position.x * Random.Range(0.0f,1.8f),
20                                         this.transform.position.y,
21                                         this.transform.position.z);
22             Instantiate(mat1, pos, mat1.transform.rotation);
23         }
24     }
25 }
26
27 }
```

Figur 10

Så skrev jeg et script til at tilfældiggøre regnestykkerne som hver kasse viste. Det var manuelt arbejde, med at skrive alle de mulige regnestykker ned (se bilag 17). Og så valgte scriptet et tilfældigt nedskrevet regnestyk, som den ville vise. Så alle regnestykker var i hensyn til at svaret ikke skulle være lig med 1. På nær boksen med regnestykket som gav 1. For at give noget player feedback, så skrev jeg et script som gør at når rumskibet rammer det rigtige resultat, så lyser boksen grønt, og når rumskibet rammer den forkerte farve, så lyser boksen rødt (se bilag 18-20). Jeg tilføjede farven rød til scriptet til de bokse med det forkerte regnestykke, og farven grøn til boksen med det rigtige regnestykke. Det virker ved at boksene skifter farve, når boksen rammer en box collider, såsom den jeg tilføjede til rumskibet.

Nu er første og den mest tidlige iteration af mit spil færdigt, så jeg spurgt 10 personer, om deres kvalitative feedback:

1. Der mangler en straf når man svarer forkert
2. Man må gerne have 3 liv
3. Global Ranking og highscores
4. Opladning af liv, eksempelvis en partikel som gav et liv mere
5. Bonus point for flere rigtige i træk
6. Spillet går hurtigere jo længere ind i spillet det er
7. Forskellige spillemodeller
8. Baggrundsmusik
9. Lyd effekter
10. Tilfældige svar, så altting ikke kun skal give 1
11. Kedelig baggrund, lav en mere moderne en
12. En start menu
13. Pause menu
14. Lav andre objekter, såsom et der giver dig et "boost" når du rammer den

## Anden iteration

Jeg startede med den del som jeg vurderede til at være den mest væsentlige del af hovedmekanikken i spillet, da det var en del af min MVP. Jeg ville ud fra min kvalitative feedback forbedre matematik delen af spillet, så den var fuldstændig tilfældig.

Først lavede jeg en liste ”i”, hvor der kunne være tilfældige nummer fra 1-9 (som selvfølgelig kunne ændres). Derefter lavede jeg en liste ”j” med præcis det samme. Disse to variable kan så enten ganges, divideres, pluses eller minuses til at give et bestemt tal (se figur 11, til højre). Dette tal skulle så være lig med min ”random.range” variable. Jeg lavede derefter nogle public spilleobjekter, hvor kun en af dem ville visse et regnestyk som var lig med min ”random.range”, mens de resterende var lig med alt andet som ikke var lig med min ”random.range”. Nu er alle disse variabler og spilleobjekters information bibeholdt i en array, som jeg så skal få til at blive vist på tekst stykket tilkoblet de forskellige spilleobjekter inde i spillet. Det gjorde jeg ved lave et skript som fandt komponenterne der var sat på de forskellige spilleobjekter. Derefter brugte jeg en ”SetData” funktion, der ændrede teksten i spillet til de tilfældigt genereret regnestykker (se bilag 27).

Jeg printede også selve svarene på regnestykkerne i spillet, så man var klar over, hvilket regnestyk man skulle ramme (se bilag 28 og 29). Derefter ændrede jeg scriptet som

```

47 // Take all combinations with correct sum
48 List<List<int>> OK = number_list.FindAll(i => (i[0]*i[1] == diff));
49
50 // Take all combinations with wrong sum
51 List<List<int>> not_OK = number_list.FindAll(i => (i[0]*i[1] != diff));
52
53 // Randomize lists
54 System.Random rnd = new System.Random();
55 var rnd_OK = OK.OrderBy(item => rnd.Next()).Take(1).ToList();
56 var rnd_not_OK = not_OK.OrderBy(item => rnd.Next()).Take(num_wrong).ToList();
57
58 // Print numbers with correct sum
59 // bloack er navnet på cuben
60 GameObject blockCorrect = Instantiate(blockPrefabCorrect, spawnPoints[4]);
61 GameObject block2 = Instantiate(blockPrefab2, spawnPoints[0]);
62 GameObject block3 = Instantiate(blockPrefab3, spawnPoints[1]);
63 GameObject block4 = Instantiate(blockPrefab4, spawnPoints[2]);
64 GameObject block5 = Instantiate(blockPrefab5, spawnPoints[3]);
65 GameObject block6 = Instantiate(blockPrefab6, spawnPoints[5]);
66 GameObject block7 = Instantiate(blockPrefab7, spawnPoints[6]);
67 GameObject block8 = Instantiate(blockPrefab8, spawnPoints[7]);
68 GameObject block9 = Instantiate(blockPrefab9, spawnPoints[8]);
69
70 BlockData bdataCorrect = blockCorrect.GetComponent<BlockData>();
71 BlockData bdata2 = block2.GetComponent<BlockData>();
72 BlockData bdata3 = block3.GetComponent<BlockData>();
73 BlockData bdata4 = block4.GetComponent<BlockData>();
74 BlockData bdata5 = block5.GetComponent<BlockData>();
75 BlockData bdata6 = block6.GetComponent<BlockData>();
76 BlockData bdata7 = block7.GetComponent<BlockData>();
77 BlockData bdata8 = block8.GetComponent<BlockData>();
78 BlockData bdata9 = block9.GetComponent<BlockData>();
79
80 bdataCorrect.SetData(rnd.OK[0][0] + "*" + rnd.OK[0][1] + "\n\n", true);
81 bdata2.SetData(rnd.not_OK[0][0] + "*" + rnd.not_OK[0][1] + "\n\n", false);
82 bdata3.SetData(rnd.not_OK[1][0] + "*" + rnd.not_OK[1][1] + "\n\n", false);
83 bdata4.SetData(rnd.not_OK[2][0] + "*" + rnd.not_OK[2][1] + "\n\n", false);
84 bdata5.SetData(rnd.not_OK[3][0] + "*" + rnd.not_OK[3][1] + "\n\n", false);
85 bdata6.SetData(rnd.not_OK[4][0] + "*" + rnd.not_OK[4][1] + "\n\n", false);
86 bdata7.SetData(rnd.not_OK[5][0] + "*" + rnd.not_OK[5][1] + "\n\n", false);
87 bdata8.SetData(rnd.not_OK[6][0] + "*" + rnd.not_OK[6][1] + "\n\n", false);
88 bdata9.SetData(rnd.not_OK[7][0] + "*" + rnd.not_OK[7][1] + "\n\n", false);

```

Figur 11

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5 using System.Text;
6 using System;
7 using System.Linq;
8
9 public class RandomMath2 : MonoBehaviour
10 {
11     public Transform[] spawnPoints;
12     public GameObject blockPrefabCorrect;
13     public GameObject blockPrefab2;
14     public GameObject blockPrefab3;
15     public GameObject blockPrefab4;
16     public GameObject blockPrefab5;
17     public GameObject blockPrefab6;
18     public GameObject blockPrefab7;
19     public GameObject blockPrefab8;
20     public GameObject blockPrefab9;
21     public int EqualsToo;
22     public int diff; //delete
23
24     //public static void Main(string[] args)
25     //{
26     public void Start()
27     {
28
29         // Generate all combinations of i = 1..9 and j = 1..
30         List<List<int>> number_list = new List<List<int>>();
31         for (int i = 1; i <= 9; i++)
32         {
33             for (int j = i; j <= 9; j++)
34             {
35                 List<int> numbers = new List<int>();
36                 numbers.Add(j);
37                 numbers.Add(i);
38                 number_list.Add(numbers);
39             }
40         }
41         // The sum should be this number
42         //int diff = EqualsToo;
43         diff = UnityEngine.Random.Range(1, 10); //change this
44
45         // Number of wrong answers
46         int num_wrong = 8;

```

Figur 11

spawnede mine regnestykker, så stykkerne ikke blev spawnet tilfældigt, men ved et bestemt interval. Dette gjorde jeg ved en "IEnumerator" og en "Yield" funktion (se bilag 30).

Jeg valgte derefter at opdatere hovedfiguren til en raket, jeg selv havde designet i photoshop (se bilag 31). Derefter opdaterede jeg baggrunden til en mere moderne og simpel baggrund uden lige så meget støj (se bilag 32). For at få det til at se mere moderne ud, så kiggede på de mest populære spil, for at se hvordan de havde gjort det, og de havde benyttet simple geometriske figure med mange farver når de designede deres spil (se bilag 33).

Så lavede jeg hovedmenuen hvor alle funktionerne skulle være. Ud fra principippet om at min baggrund i spillet var kedelig, valgte jeg at designe en mere interessant og moderne baggrund til hovedmenuen (se bilag 21-23). Knapperne fungerede på den måde, at når du trykkede på dem, så skjulte den en menu, og viste en usynlig menu frem (se bilag 24).

Derefter lavede jeg et script, som skulle ændre hele scenen, fra min hovedmenu scene, til selve min spilscene. Scriptet fungerede ved, at den nuværende scene plus 1, skulle blive den nye scene (se bilag 25). Scener i Unity, kan sættes i en bestem rækkefølge, og på den måde så bliver scenen ændret (se bilag 26).

Alle spilleobjekter lige nu fortsætter med at eksistere selv efter de er gået ud af skærmen. Dette gør at efter noget tid så begynder spillet at lagge (se bilag 34). For at dette undgås lavede jeg en kæmpe væg bag min spiller, som alle objekter ville ramme på et eller andet tidspunkt (se bilag 35). På denne væg lavede jeg et simpelt skript, som ødelagde alle spilleobjekter som ramte den (se bilag 36).

Derefter lavede jeg en score script, hvor scoren blev større, ud fra hvor lang tid spillet havde været i gang. (se figur 12, til højre). Det gjorde jeg igennem funktionen time.deltaTime (se bilag 38).



Figur 12

Derefter lavede jeg en pausemenu. Det fungerede ved at jeg lavede et script, hvor når man trykkede på en knap, så ændrede den



Figur 13

værdien af Time.Timescale til at være 0, så spillet ikke længere kørte. På samme måde lavede jeg en knap til at starte spillet igen, ved at den satte Time.Timescale til at være 1 (se bilag 39). Derefter lavede jeg en hjem knap som skulle tage spilleren over til hovedmenuen (som der kan ses på figur 13 til venstre). Dette fungerede ved, at jeg lavede en string, som kunne ændres til en bestem scenes navn. Når man så aktiverede scriptet igennem en knap så ændrede man derved scene til det navn man havde angivet (se bilag 40).

Så begyndte jeg at lave en livbar, så man ville kunne få og miste liv (se bilag 41). Dette gjorde jeg ved at når spilleren ramte et forkert svar, så ville man miste x antal liv, og når man ramte det rigtige svar, så ville

man få x antal liv. Hvis ens liv så ramte 0, så ville en if statement aktivere, som gjorde at spillet stoppet og spilleren døde (se bilag 42). Derefter benyttede jeg funktionen SetActive(false); til at fjerne den normale UI, og aktivere gameover menuen. Ud over det så gemte jeg scoren i en variable, så man vil kunne se ens score efter man er død (se bilag 43).

Nu er anden iteration af mit spil færdigt, så jeg spurgte 13 personer, om deres kvalitative feedback:

1. Knapperne på hovedmenuen skal justeres så de virker bedre
2. Restart knap så man hurtigt kan prøve igen
3. I lig med tallene som kommer før regnestykkerne skal være en skrigende farve, eventuelt farveblind vendelig.
4. Svære regnestykker
5. Flere figurer og variation i din spiller
6. Musik og lydeffekter
7. Kasserne skal være større
8. At det ikke er muligt at få rigtig og forkert på sammentid
9. Gemt highscore
10. Spillet følger et mere bestemt tema, eksempelvis ser det mærkeligt ud med tredimensionelle bokse
11. Regnestykkerne er mere tydelig

## Tredje iteration

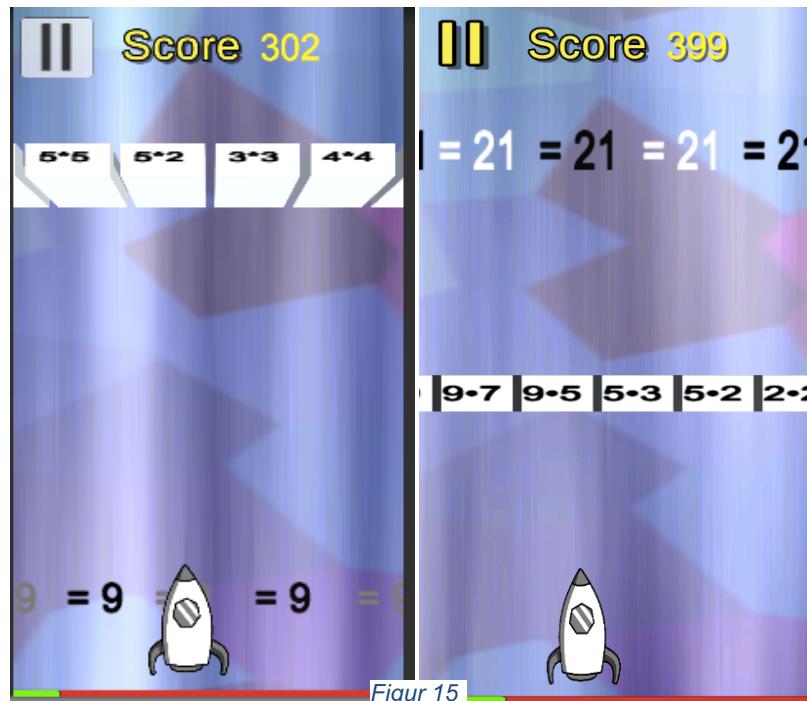
For at gøre spillet sværere skal gangestykkerne give en højere værdi. Her opstod der et problem, hvor to naturlige tal ikke kan give et primtal. Derved stoppede spillet med at virke. Så for at lave svære gange stykker, blev jeg nødt til at lave om på mit script, så den ignorerede primtal over 10. Det gjorde jeg ved en if statement, inden i et for loop, som ville blive ved, hvis tallet som var genereret var et primtal over 10 (se figur 14 nedenunder). For at spillet vidste hvad et primtal var, så lavede jeg en primtals detektor i scriptet (se bilag 37). Derefter kom der et nyt problem, hvor spillet igen holdt op med at virke, når den ramte et tal som ikke var muligt at få ved at gange 2 tal under 10 med hinanden, så som 60. For at fikse dette, så lavede jeg en Int som tog 2 af mine tal fra mit

array og ganget dem sammen til at få et resultat som var muligt.

```
for ( ; ; )
{
    int rnd_num = UnityEngine.Random.Range(1, number_list.Count()) / 2; //change this
    diff = number_list[rnd_num][0] * number_list[rnd_num][1];
    if (diff > 10 && IsPrime(diff))
    {
        continue;
    }
    break;
}
```

Figur 14

Derefter for at få tallene til at blive mere tydelige og selve spillet til at følge et mere konkret tema, så satte jeg skygge på scoren, ændrede pause knappen, ændrede regnestykkerne så de ikke længere er tredimensionelle og gjort så talenes proportioner er bedre, så det ikke ligner at tallene er sammenpresset mere. Ud over dette, så har jeg sat farvetemaet til at være hvid og sort i stedet for grå og sort når det kommer til regnestykkerne og resultatet. Så har jeg sat en sort boks rundt om regnestykkerne, så det ligner at de er sådan en port man forbi passerer i stedet for bare nogen objekter man skal ramme. Alt dette kan ses på figur 15 til højre. Venstreside er hvordan det så ud før, og højre er hvordan det ser ud nu.



Figur 15

For at forbedre interaktionsdesignets "error" element, så lavede jeg rakettens "box collider" til en tynd pind, så den kun kan ramme og aktivere regnestykket, som spidsen af raketten rammer.

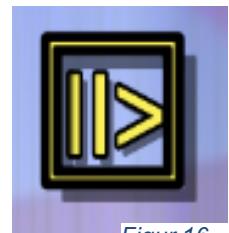
For at der var noget mere konkret feedback, så lavede jeg 3 lyd effekter i programmet FL studio. Den ene korte lyse lyd til når du fik et svar rigtigt. Så en dyb lang lyd til når man

fik et svar forkert, og så en neutral lyd til hver gang man trykker på en knap (se bilag 44). For at lydene blev spillet på de rigtige tidspunkt, så lavede jeg et hurtigt script, som gjorde at når spilleren ramte ind i et objekt med tagget ”rigtig” så blev den ene lyd afspillet, og når spilleren ramte ind i et objekt med tagget ”forkert” så blev den anden lyd afspillet, osv (se bilag 45).

Derefter hoppede jeg ind i FL studio igen, for at lave en baggrunds melodi. Det gjorde jeg ved at lave en klaver melodi kombineret med noget stille og rolig bas (se bilag 47).

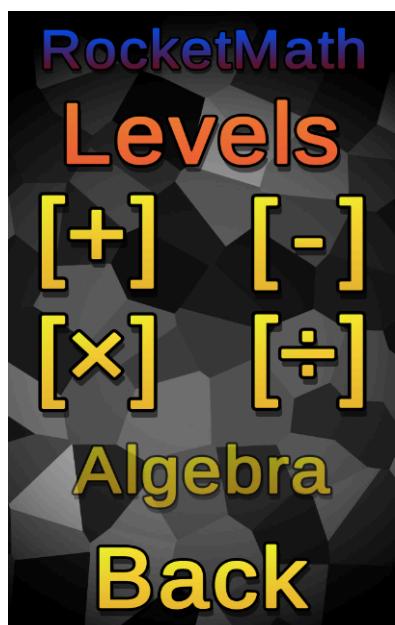
Derefter benyttede jeg samme script som jeg brugte til at loade banen, til at lave en knap på gameover menuen, som gjorde du kunne genindlæse banen. Så du ikke behøver gå helt ud af spillet for at spille samme bane igen.

Dernæst begyndte jeg at lave et objekt som ville gør spillet langsommere i et par sekunder hvis man ramte den. Først designede jeg et objekt (se figur 16 til højre), og derefter fik jeg den til at spawne ved et tilfældigt interval mellem min minimum og maksimum værdi. Derefter så brugte jeg en funktion der halveret spillets hastighed, og ødelagde objektet, så det forsvandt. Jeg brugte en ”Yield Return New timer”, så den efter 3 sekunder ville sætte hastigheden op til normal spillehastighed (se bilag 46).



Figur 16

Jeg begyndte at lave et highscore system. Dette lykkedes ved hjælp af C#s PlayerPrefs funktion, som gjorde det muligt at gemme en værdi. Jeg lavede så et kort script, der



Figur 17

sagde at den højeste score som blev lavet, skulle gemmes, og printes på et tekst objekt i spillet (se bilag 48-50).

Dernæst kopierede jeg min scene med spillet 4 gange, så der var en scene til hver af regnearterne. Så ændrede jeg en smule af koden til hver scene, så regnestykkerne passede til deres respektive scene. Dernæst lavede jeg hovedmenuen lidt om, så man kunne vælge hvilken af regnearterne man ville øve (se figur 17 til venstre).

For at gøre scenerne mere interessante, så skrev jeg en lille kode, som gjorde at min rumrakket havde forskellig farve fra scene til scene. (se bilag 51)

For at gøre spillet klar til at teste det på min målgruppe, så ville jeg implementere touch funktion til spillet, så spillets styring forgår ved hjælp af skærmklik. Dette gjorde jeg ved at benytte Unitys touchInput funktion, som kan registrere når spilleren trykker på skærmen. Derefter sagde jeg at hvis spilleren trykkede på venstre side af skærmen, så ville spilleren rykke sig til venstre, med mindre x værdien var højere end hvor bredt spillet var. Det samme gjorde jeg med højre side (se bilag 52).

Så var jeg klar til at eksportere det til app store, men opdagede lige pludselig at der er en utrolig lang venteperiode før spil kan blive accepteret. Ud over dette, så koster det åbenbart 1000kr om året, at have spillet på app store. Derfor benyttede jeg Unitys "remote control" som gjorde det muligt at teste spillet direkte på telefonen, uden at det skal blive accepteret på app store.

Her er en video af det endelig spil: [https://youtu.be/qJ\\_SoRqMAWI](https://youtu.be/qJ_SoRqMAWI)

## Postremum testfase

I den sidste fase af spillet, så fik jeg adspurgt en folkeskole vedrørende om jeg måtte låne et par elever til at teste spillet. Testen forgår på den måde at jeg først giver eleverne et øve ark, hvor de får et minut til at svare rigtig på så mange som muligt (se bilag 52).

Derefter skal de spille mit spil i fem minutter, og derefter skal de prøve at lave øve arket igen. Ud fra om de er i stand til at svare rigtig på flere regnestykker, kan jeg se om de er blevet bedre eller ej. For at sikre mig at grunden til de er blevet bedre ikke er bare fordi de lige havde lavet et øvelses ark, så fik jeg lavet en kontrolgruppe. Kontrolgruppen skal lave øvelses arket, og derefter vente fem minutter, hvorefter de skal lave øvelses arket igen. Denne værdi som jeg får fra kontrolgruppen, kan jeg så fjerne fra forbedringerne (hvis forbedring eksisterer) hos dem der fik spillet mig spil. Dette kvantitative eksperiment skal svare på mit arbejdsspørgsmål vedrørende om det gør eleverne bedre.

Når de spiller mit spil, så vil jeg undervejs observere om de selv kan finde ud af at navigere spillet, eller om jeg skal forklare hvordan det fungerer. De får 30 sekunder til selv at finde ud af spillet, ellers forklarer jeg hvordan det fungerer. For hver gang jeg skal forklare det, så noterer jeg det, så jeg vil kunne udregne hvor intuitivt spillet er i procent. På denne måde kan jeg teste interaktionsdesignet når det kommer til "learnability", og "efficiency".

Engang de er færdige med at spille, så skal eleverne vurdere på en skala fra 1-5, hvor sjovt spillet er, hvor 1 er lige så sjovt som at lave en masse øvelses ark, og 5 er et spil de selv ville kunne finde på at spille i deres fritid.

Ud over dette skal de vurdere på en skala fra 1-5 hvor professionelt spillet ser ud, hvor 1 ligner noget vuggestue børn tegner, og hvor 5 er lige så godt som de andre spil de spiller. På den her måde kan jeg vurdere interaktionsdesignets "satisfaction" faktor.

## Resultater og data behandling

Kontrol gruppe	Antal rigtige	Antal rigtige andet forsøg	% stigning
Pers. 1	16	18	12,5
Pers. 2	14	15	7,14285714
Pers. 3	13	14	7,69230769
Pers. 4	19	19	0
Pers. 5	16	17	6,25
Pers. 6	15	17	13,33333333
Gennemsnit	15,5	16,67	7,53

Kontrolgruppen nåede i gennemsnit at svare rigtig på 15,5 spørgsmål på et minut første gang de lavede øvelses arket. Efter de havde ventet 5 minutter, så kunne de i gennemsnit svare rigtig på 16,67 af regnestykkerne. Dette var en stigning på 7,53 procent ved bare at lave øvelses arket en gang til.

Forsøg	Antal rigtige	Antal rigtige andet forsøg	% stigning	Forstod de spillet? Ja/nej	Professionalisme 1-5	Sjovhed 1-5
Pers. 1	16	18	12,5	ja	4	5
Pers. 2	14	15	7,14285714	ja	3	4
Pers. 3	14	14	0	nej	4	2
Pers. 4	14	19	35,7142857	ja	4	5
Pers. 5	13	17	30,7692308	ja	3	5
Pers. 6	20	21	5	ja	4	3
Pers. 7	19	18	-5,2631579	ja	5	2
Pers. 8	18	20	11,1111111	ja	4	4
Pers. 9	15	18	20	ja	4	4
Pers. 10	15	18	20	ja	4	4
Pers. 11	9	14	55,5555556	ja	4	4
Pers. 12	12	15	25	ja	3	4
Pers. 13	15	16	6,66666667	nej	3	4
Pers. 14	16	19	18,75	ja	4	4
Pers. 15	17	20	17,6470588	ja	5	5
Pers. 16	20	24	20	ja	4	4
Pers. 17	19	22	15,7894737	ja	4	4
Gennemsnit	15,65	18,12	15,79	88% forstod det	3,88	3,94

Hos de personer som skulle prøve spillet, så svarede de i gennemsnit rigtig på 15,65 af regnestykkerne før de prøvede mit spil. Efter de havde spillet mit spil i et par minutter så svarede de i gennemsnit rigtig på 18,12 af spørgsmålene. Dette var en procentvis stigning på 15,79%. Hvis vi fjerner elementet fra kontrolgruppen om, at de bare blev bedre til at udregne regnestykkerne fordi de lavede øvelsesarket igen, så er det en procentvis stigning på  $15,79\% - 7,53\% = 8,26\%$ . Med andre ord er der ikke nogen tvivl om, at spillede gjorde eleverne bedre til basal matematik. 88% af eleverne forstod spillereglerne inden for 30 sekunder, mens jeg blev nødt til at forklare hvordan spillet

fungerede til de resterende 12%. I snit så gav de mit spil 3,88/5 når det kom til hvor professionelt og flot spillet så ud. De syntes det var væsentligt sjovere at lære matematik på den her måde, end ved at lave øvelses opgaver igen og igen. Det kan ses i det de gav mit spil 3,94/5 i sjovhedsgrad. Der var faktisk et par af eleverne som ikke kunne lad hver med at prøve igen og igen. Da de ville slå deres egen highscore. Mange af eleverne spurgte mig endda om de kunne downloade spillet selv.

## Diskussion

For så vidt angår mine test resultater, så kan det ikke siges om testpersoner faktisk er blevet permanent klogere, eller om det bare er korttidslære. For at sikre og efterteste at eleverne faktisk lærer noget, så bør testen anvendes på de samme testpersoner over en længere periode så resultaterne kan sammenlignes fra dag til dag.

Et andet problem er, at spillet optager så utroligt meget af computerens hukommelse (ram), at det lukker ned nogle gange. En anden gang kunne spillets kode godt optimeres, og selve spillet gøres mere effektivt, så den ikke skal tænke på så mange ting på samme tid.

Spillet virker altså ikke helt perfekt altid, og kræver en del fra computeren. Hertil kommer, at spillet ikke nødvendigvis er det kønneste, for eksempel ser regnestykkerne en smule sjusket ud.

Spillet benytter flittigt motivationsteoriens overvejelser om præstationsmotivation, da high score systemet så ud til at fungere motiverende på eleverne. Til gengæld var der ikke megen mestringsmotivation i spillet, eftersom det ikke tager lang tid at blive rigtig godt til en af de fire regnearter. I fremtiden kunne der med fordel implementeres en form for achievement system, hvor der er nogle bestemte mål man skal opnå. Tillige kunne der muligvis laves et system, hvor spillet bliver sværere og sværere.

## Konklusion

Jeg har skabt et produkt som kan være med til at hjælpe på skoleelevernes manglende motivation i folkeskolen. Jeg har ikke genopfundet den dybe tallerken, men i stedet fundet på en lille forbedring til hvordan et spil kan være sjovt og lærerigt. For at det skulle virke mest muligt motiverende, har jeg fokuseret på at vække den kompetitive spille person inden i os, også kaldet "Achievers". Det virkede udemærket, da der var op til flere af de elever, som jeg testede til sidst, der igen og igen prøvede at slå deres egen rekord. Mine test resultater har også vist, at selve spillet er lærerigt, da eleverne er blevet i stand til at udregne basale matematiske regnestykker en del hurtigere efter de havde spillet mit spil. For eleverne var spillet også relativt intuitivt, da de meget hurtigt var i stand til at forstå essensen af spillet og gav endda spillet 3,94 ud af 5 mulige point hvad angik spillets sjovhedsgrad. Der, hvor der primært er plads til forbedringer, er spillets udseende som skal se mere professionelt ud, og spillets mekanik, der skal optimeres. Som beskrevet, gør spillet også kraftigt brug af motivationsteori og intuitivt design. Således kan jeg konkludere, at jeg har svaret på alle mine arbejdsspørgsmål og at jeg har opnået det mål, som jeg satte mig for i problemformuleringen. Spillet fungerer motiverende hos eleverne og derved kan spillet være med til at gøre de matematiske fag mindre kedelige. Derved kan dette spil være med som et lille skridt imod at flere personer finder de matematiske videnskaber interessant og derved måske uddanner sig inden for de matematiske videnskaber, så vi ikke falder for langt bag ud i forhold til den internationale konkurrence, og øge velfærden på en national skala.

## Perspektivering

Jeg havde håbet på, at jeg ville kunne nå igennem flere iterationer af spillet, og at jeg ville nå at implementere flere funktioner end jeg nåede. Problemet var bare, at for hver lille funktion i spillet, så skulle jeg nå at sætte mig ind i nyt materiale, og slås med en masse fejl som spillet spyttede i hovedet på mig. Det gjorde at hver enkelt funktion kunne tage mange timer, de store funktioner endda flere dage, før jeg fandt en løsning. Til gengæld har jeg nået at lære et helt nyt fag fra bunden af (c# programmering), og

fået færdiggjort et produkt bestående af mere end 1000 linjer. Jeg føler, at jeg forstår den programmerings videnskabelige verden så meget bedre end da jeg startede.  
At programmere er meget mere tidskrævende end man lige umildbart skulle tro.

Andre løsninger til at gøre skoleelverne faglige kompetencer bedre, kunne være at omstrukturere undervisningen, så undervisningen kan forgå på anden form, end tavleundervisning. Især for de unge elevers vedkommende, så har de svært ved at koncentrere sig om en ting i særligt lang tid adgangen. Hvis hele undervisnings systemet blev lavet om, så kunne problemet måske blive løst, i forhold til mit spille produkt som næsten fungere som et plaster på såret.

## Tidsplan

### TEKNIK A PROJEKT



## Litteraturliste

- Aisinger, P. (21. 2 2019). *Motivation af drengene: Mindre snak og mere action i skolen*. Hentet 9. 3 2019 fra Folkeskolen: <https://www.folkeskolen.dk/653756/motivation-af-drengene-mindre-snak-og-mere-action-i-skolen>
- Barse, M. (10. 8 2015). *Sådan bliver elever motiveret i skolen*. Hentet 7. 3 2019 fra Videnskab: <https://videnskab.dk/kultur-samfund/sadan-bliver-elever-motiveret-i-skolen>
- Bingham, J. (12. 7 2012). *Study makes you happier, official figures suggest*. Hentet fra Telegraph: <https://www.telegraph.co.uk/education/educationnews/9379054/Study-makes-you-happier-official-figures-suggest.html>
- Brainstats. (16. 3 2019). *World ranking of countries by their average IQ*. Hentet 16. 3 2019 fra Brainstats: <https://brainstats.com/average-iq-by-country.html>
- Dickerson, R. E. (2005). *Exponential correlation of IQ and the wealth of nations*. Molecular Biology Institute. Los Angeles: Elsevier. Hentet fra <https://www.gwern.net/docs/iq/2006-dickerson.pdf>
- Extra-Credits. (4. 7 2013). *When Difficult Is Fun - Challenging vs. Punishing Games - Extra Credits*. Hentet fra Youtube: <https://www.youtube.com/watch?v=ea6UuRTjkKs>
- Folkeskolen Forskning. (2017). *Spil Styrker Motivationen*. Hentet 1. 3 2019 fra Folkeskolen: <https://backend.folkeskolen.dk/~3/2/fs1617folkeskolenforskningfinal.pdf>
- Jacobsen, S. B. (13. 5 2013). *Danske Elever Mangler motivation*. Hentet 11. 3 2019 fra Berlingske: <https://www.berlingske.dk/samfund/danske-elever-mangler-motivation>
- Kinniburgh, T. (N/A). *Top 10 Game Mechanics for Hyper Casual Games*. Hentet 12. 3 2019 fra Mobilefreetoplay: <https://mobilefreetoplay.com/top-10-game-mechanics-for-hyper-casual-games/>
- Møller, J. H. (2017). *Computerspil*. Aarhus: Systime.
- Nygaard, S. (18. 9 2018). *Disse 20 djøn-uddannelser giver højst løn*. Hentet 4. 3 2019 fra Djøf Bladet: <https://www.djoefbladet.dk/artikler/2018/9/disse-20-dj-oe-f-uddannelser-giver-h-oe-jest-l-oe-n.aspx>
- Pi day. (16. 8 2018). *10 reasons why math is important in life*. Hentet 6. 3 2019 fra Pi day: <https://www.piday.org/2018/10-reasons-why-math-is-important-in-life/>
- PISA. (2016). *PISA worldwide ranking average score of math, science and reading*. Hentet 2. 3 2019 fra Factsmaps: <http://factsmaps.com/pisa-worldwide-ranking-average-score-of-math-science-reading/>
- Stensdal, K. (13. 12 2017). *Opgørelse over de mest eftertragtede nye it-profiler: Disse it-folk kan vælge og vrage* Read more at <https://www.computerworld.dk/art/241975/opgoerelse-over-de-mest-eftertragtede-nye-it-profiler-disse-it-folk-kan-vaelge-og-vrage#jckql4GOEjY7zofu.99>. Hentet 14. 3 2019 fra computerworld: <https://www.computerworld.dk/art/241975/opgoerelse-over-de-mest-eftertragtede-nye-it-profiler-disse-it-folk-kan-vaelge-og-vrage>
- TheTopTens. (11. 3 2019). *Important school subjects*. Hentet 11. 3 2019 fra TheTopTens: <https://www.thetoptens.com/important-school-subjects/>
- Undervisnings Ministeriet. (1 2016). *Fagmål for faget matematik*. Hentet fra EMU: <https://arkiv.emu.dk/sites/default/files/Matematik%20-%20januar%202016.pdf>

Eksamensprojekt i Teknik A, 2019  
Rapport udformet af Hans Heje, 3.z, Albertslund Gymnasium.

Undervisnings Ministeriet. (24. 2 2019). *Adgangskrav til de gymnasiale uddannelser*. Hentet 11.

3 2019 fra Uddannelsesguiden:

<https://www.ug.dk/uddannelser/artikleromuddannelser/omgymnasialeuddannelser/adgangskrav-til-de-gymnasiale-uddannelser>

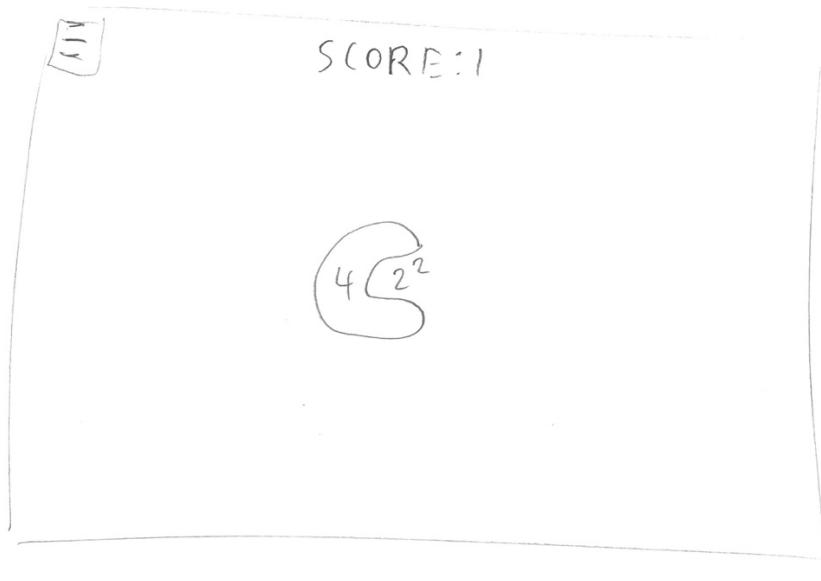
York, T. (2019). *Making Lean Startup Tactics Work for Games*. Hentet 18. 3 2019 fra

Gamasutra:

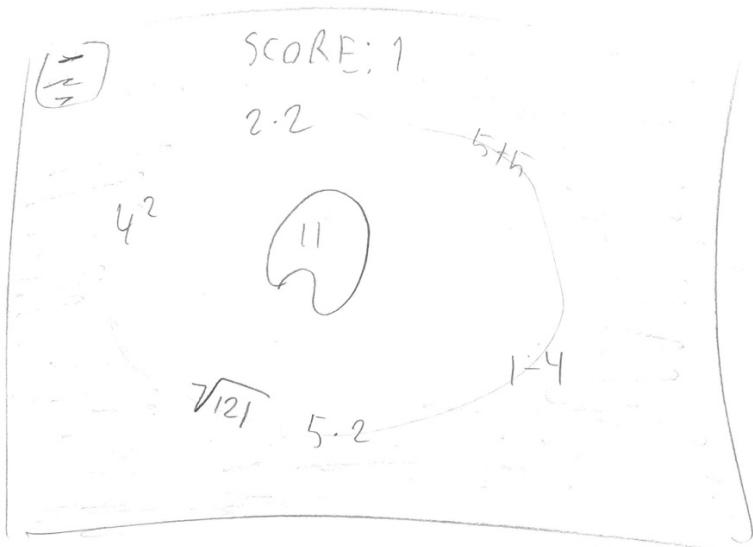
[https://www.gamasutra.com/view/feature/168647/making\\_lean\\_startup\\_tactics\\_work\\_.php?print=1](https://www.gamasutra.com/view/feature/168647/making_lean_startup_tactics_work_.php?print=1)

## Bilag

### Bilag 1



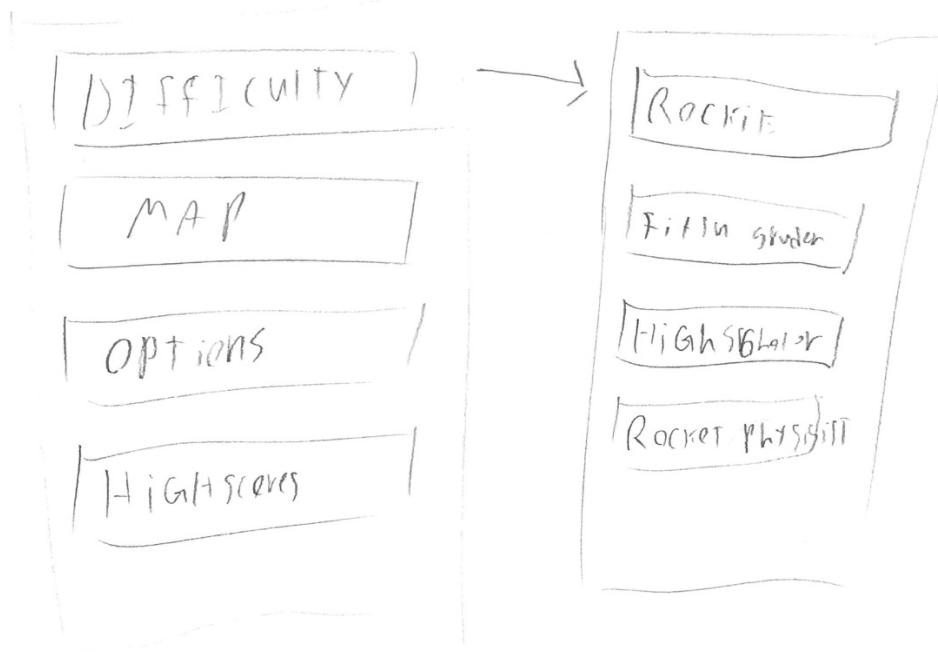
### bilag 2



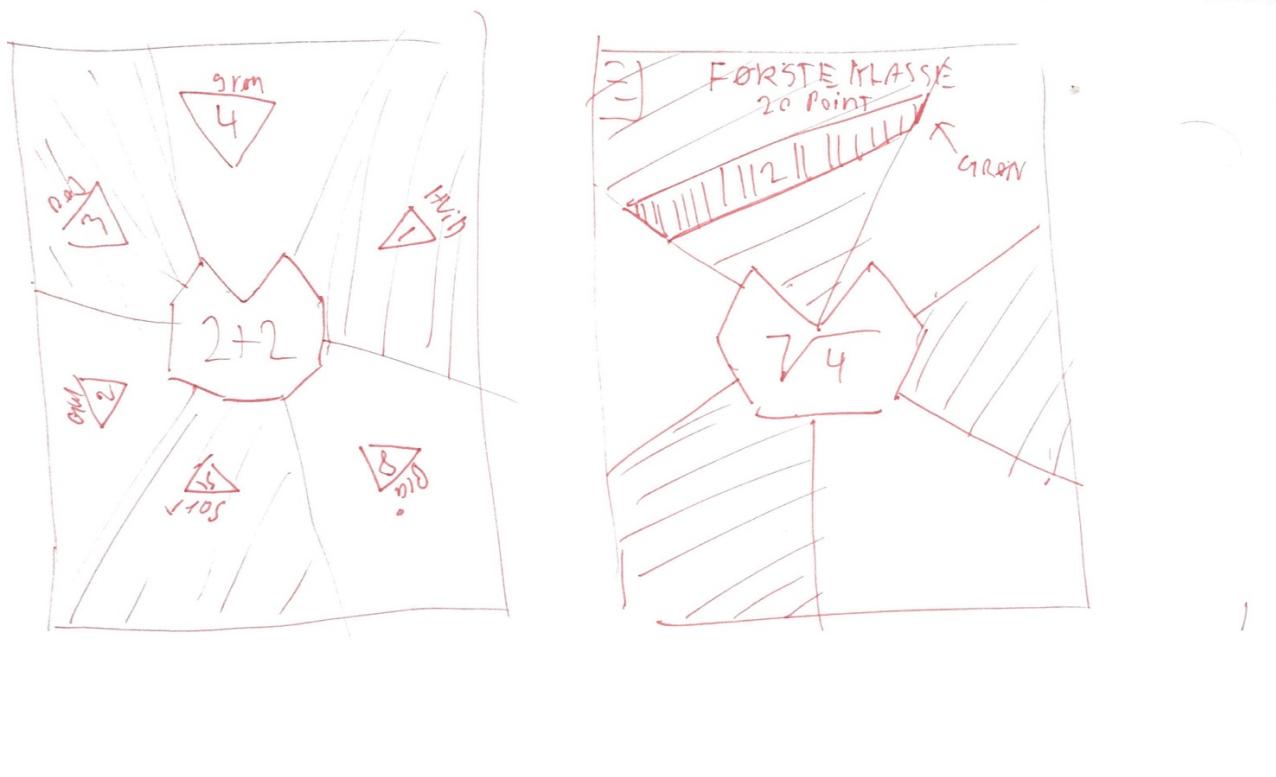
Bilag 3



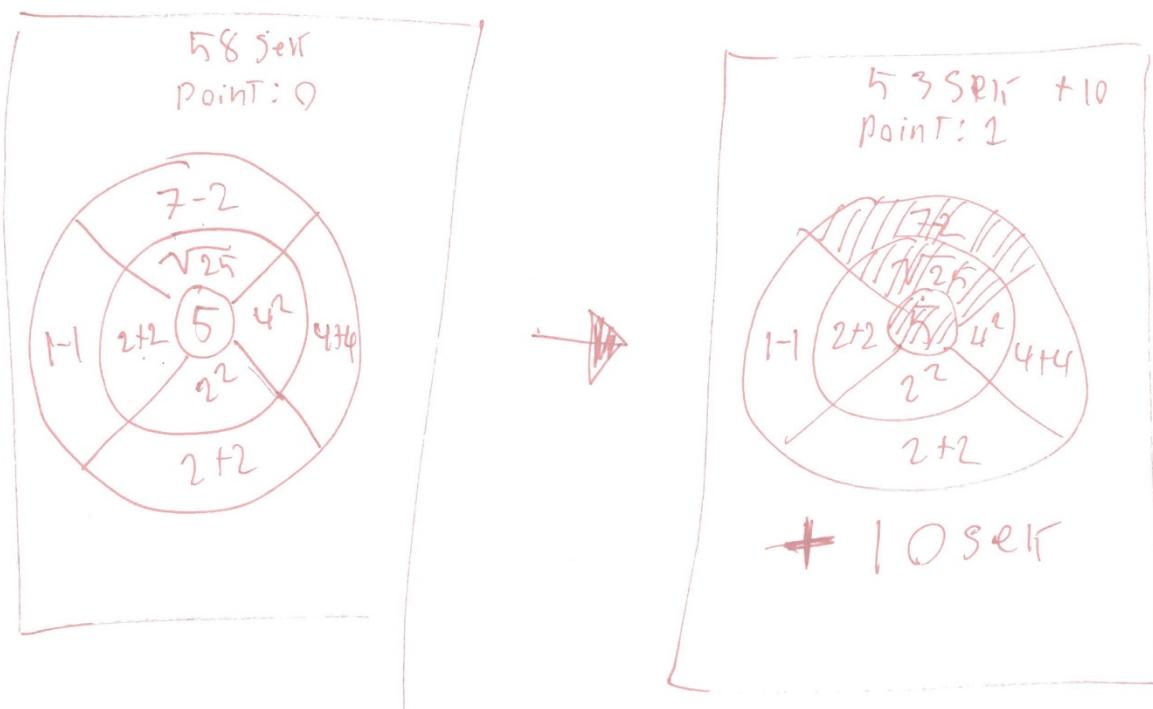
Bilag 4



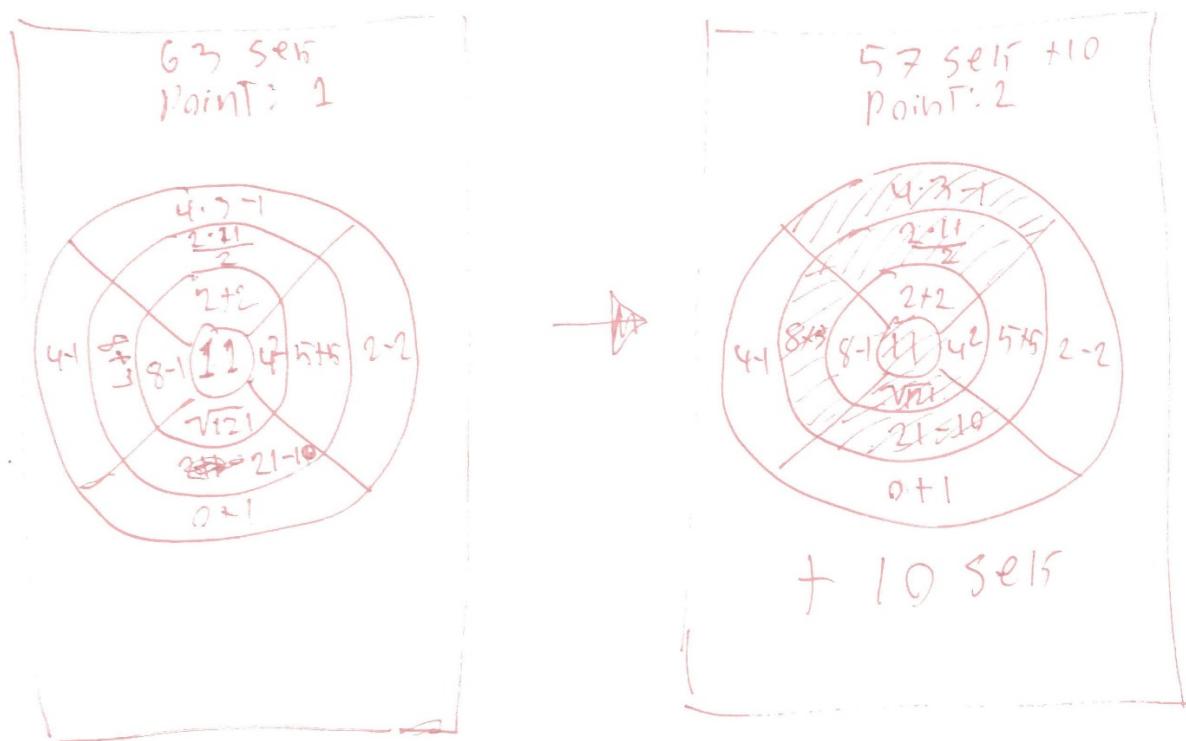
Bilag 5



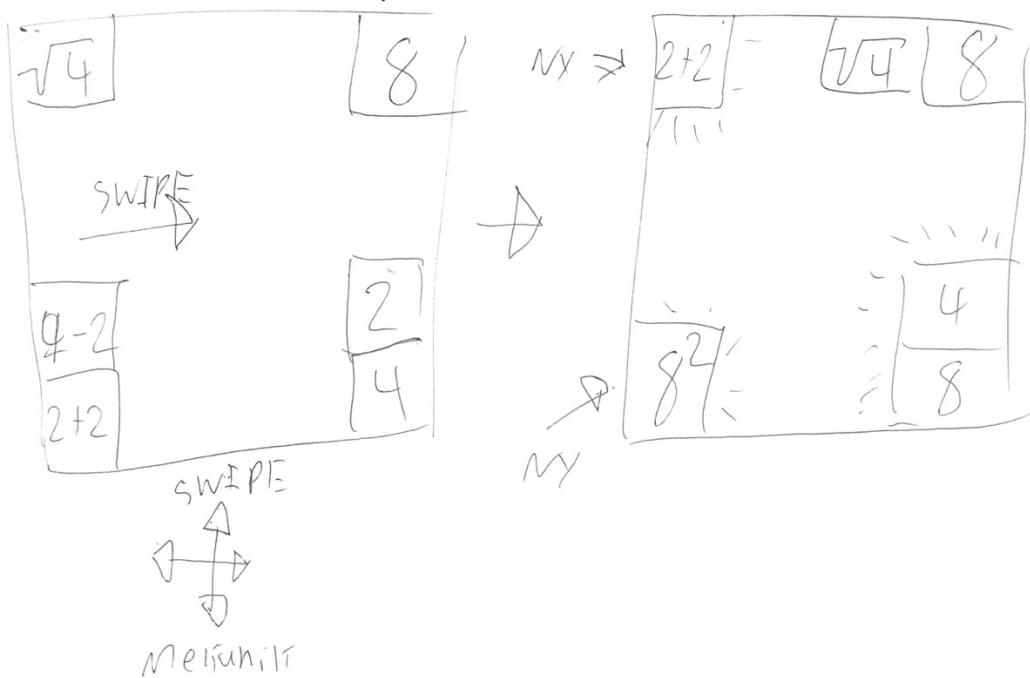
Bilag 6



Bilag 7



bilag 8



bilag 9

$$\lambda = 2 \mid \lambda = 1 \mid \lambda = 5 \mid \lambda + 4 = 6$$
$$x^2 = 4$$
$$x = 3 \mid x = 5 \mid x = 2 \mid x = 0$$

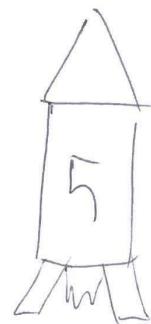
$3x = 5$

← SWIPE →

bilag 10

9

$$3+2 \mid 5-5 \mid 2^2 \mid \sqrt{25} \mid 2 \cdot 2$$

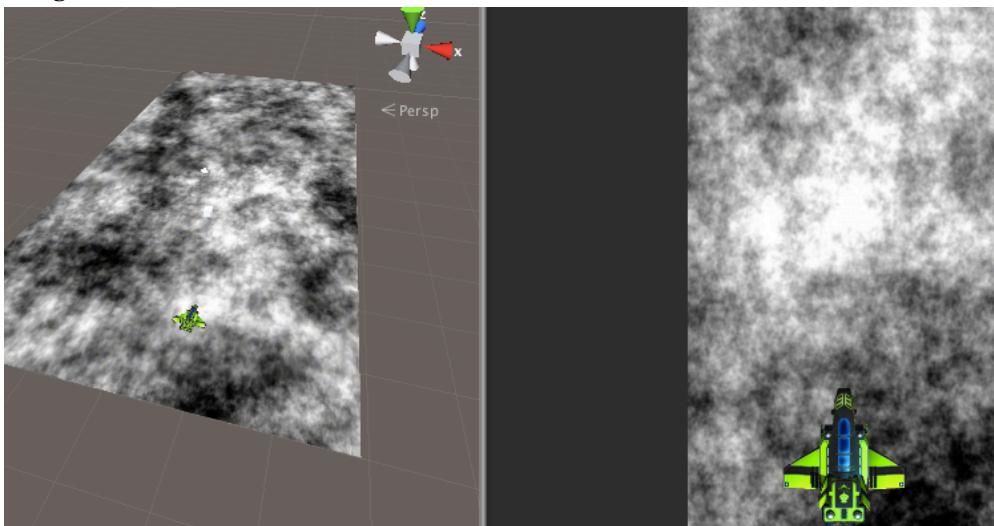


← SWIPE →

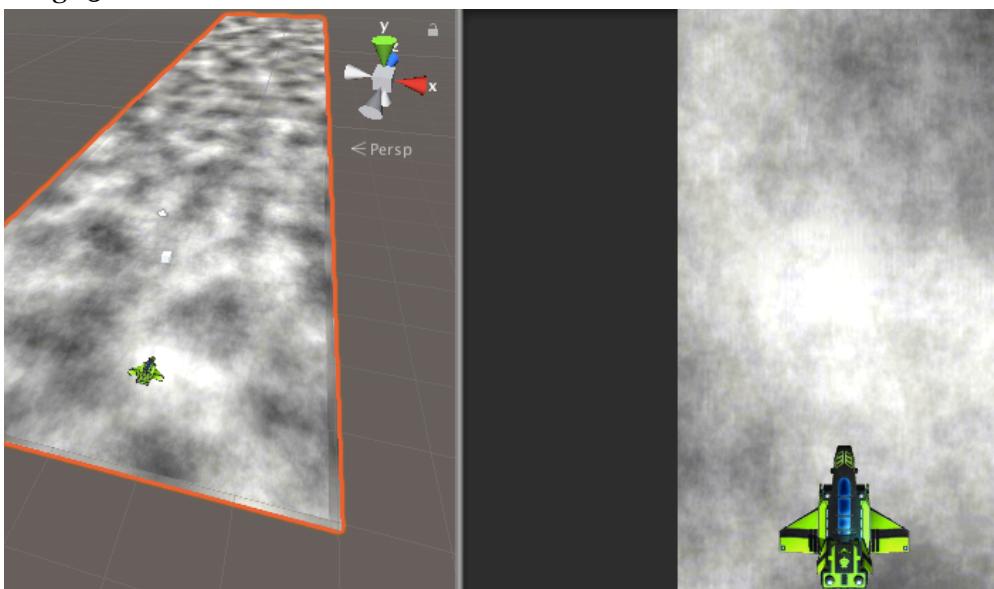
Bilag 11



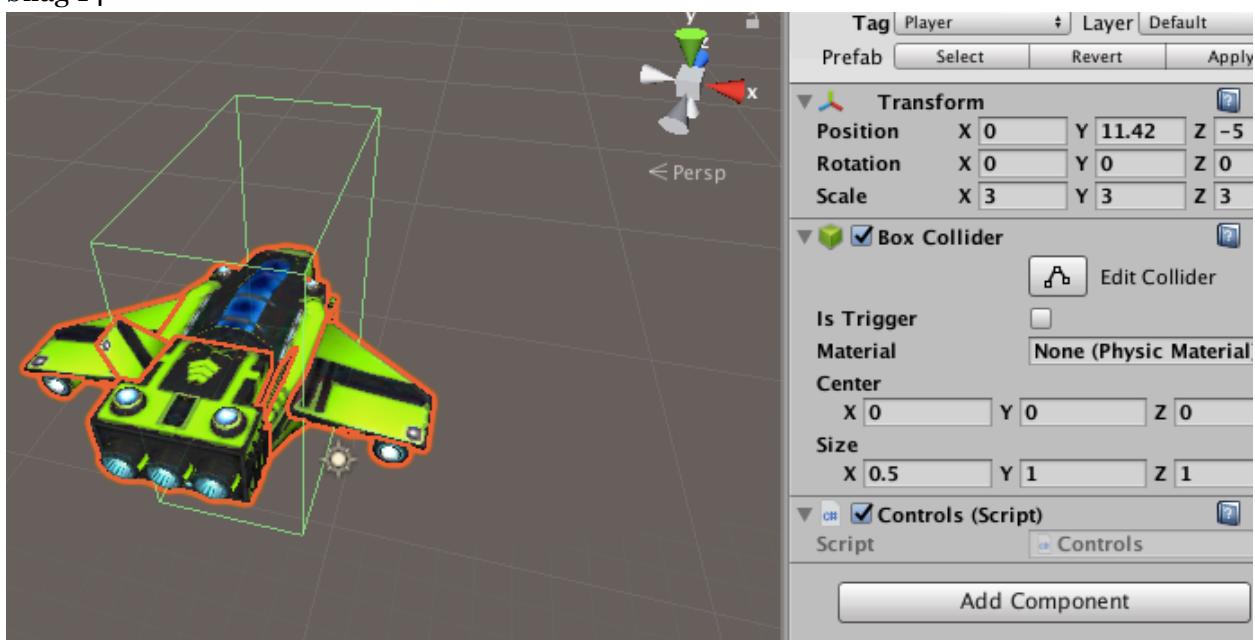
bilag 12



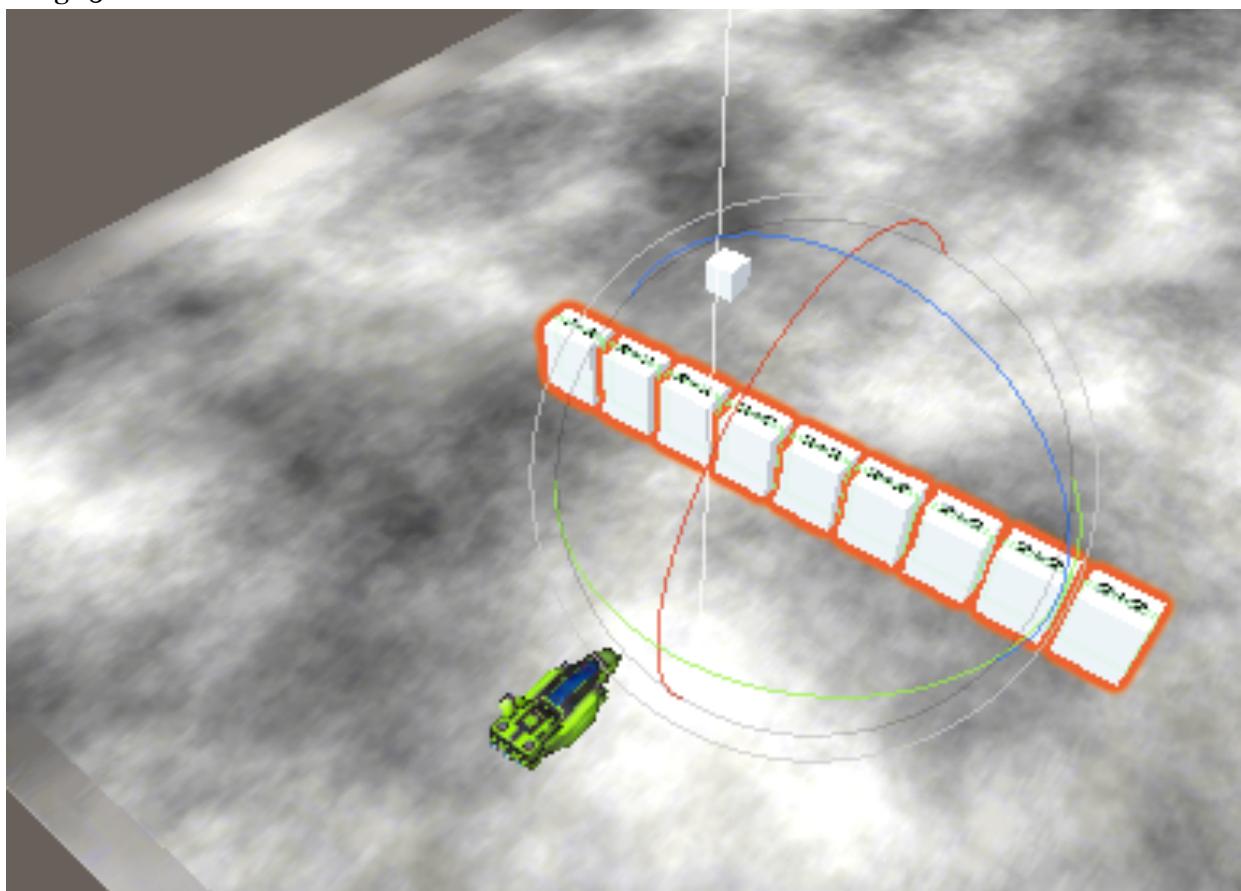
bilag 13



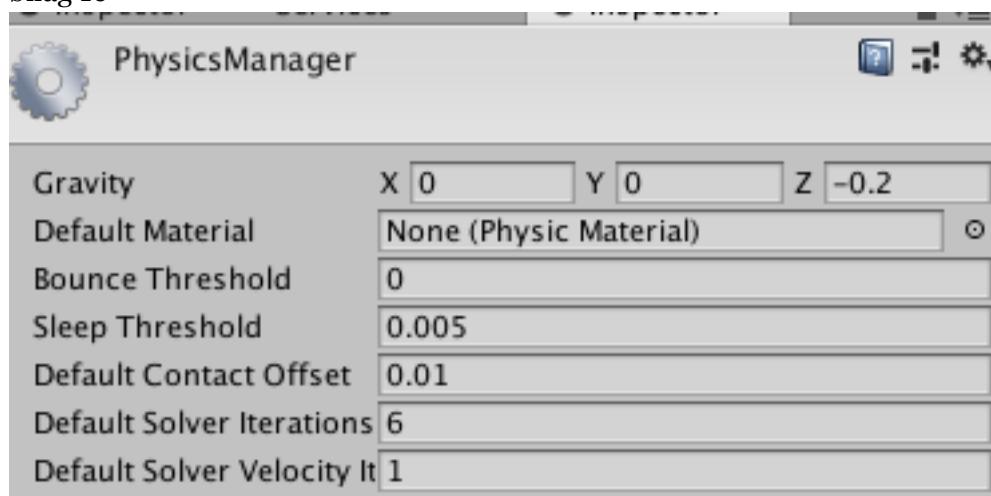
bilag 14



bilag 15



bilag 16



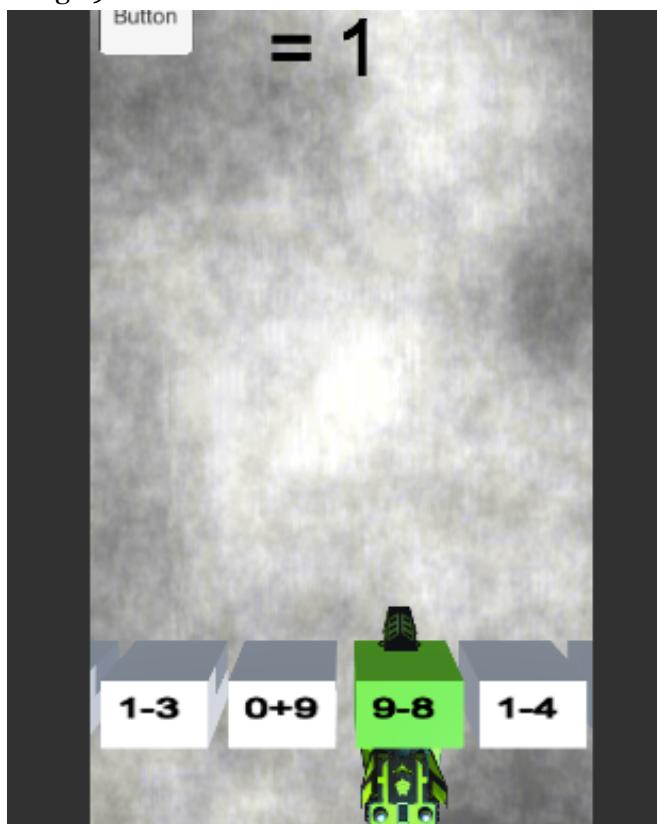
Bilag 17

```
180     "6-9",
181     "7-0",
182     "7-1",
183     "7-2",
184     "7-3",
185     "7-4",
186     "7-5",
187     // "7-6", //lig med 1
188     "7-7",
189     "7-8",
190     "7-9",
191     "8-0",
192     "8-1",
193     "8-2",
194     "8-3",
195     "8-4",
196     "8-5",
197     "8-6",
198     // "8-7", //lig med 1
199     "8-8",
200     "8-9",
201     "9-0",
202     "9-1",
203     "9-2",
204     "9-3",
205     "9-4",
206     "9-5",
207     "9-6",
208     "9-7",
209     // "9-8", //lig med 1
210     "9-9",
211 };
212
213 void Start()
214 {
215     myText = GameObject.FindGameObjectWithTag("Score").GetComponent<Text>();
216     string myString = animalDescriptions [Random.Range (0, animalDescriptions.Length-1)];
217     myText.text = myString;
218     GetComponent<TextMesh>().text = myString;
219 }
220 }
```

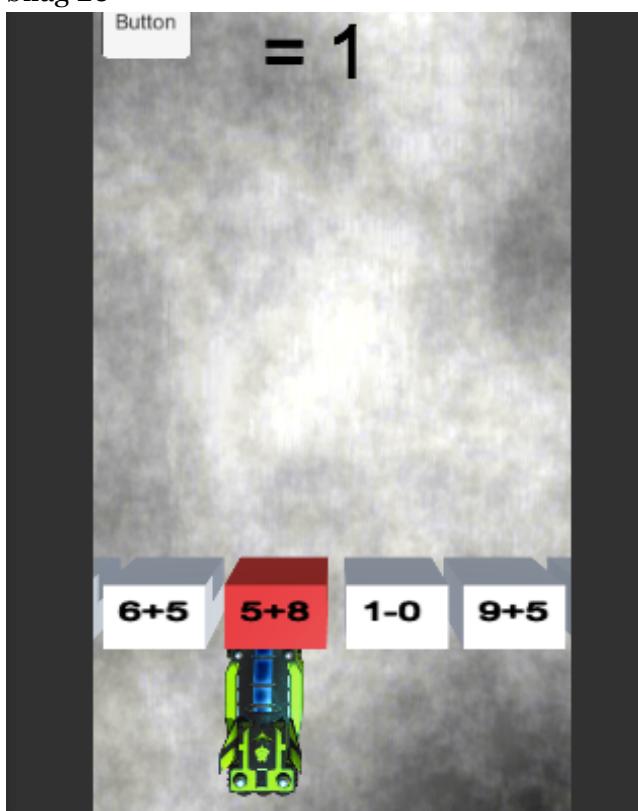
Bilag 18

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Correct : MonoBehaviour
5  {
6
7      public Material[] material;
8      Renderer rend;
9
10     void Start()
11     {
12         rend = GetComponent<Renderer>();
13         rend.enabled = true;
14         rend.sharedMaterial = material[0];
15     }
16     void OnTriggerEnter(Collider col)
17     {
18
19         if (col.gameObject.tag == "Player")
20         {
21             rend.sharedMaterial = material[1];
22         }
23         /*else
24         {
25             rend.sharedMaterial = material[2];
26         }*/
27     }
28 }
```

bilag 19



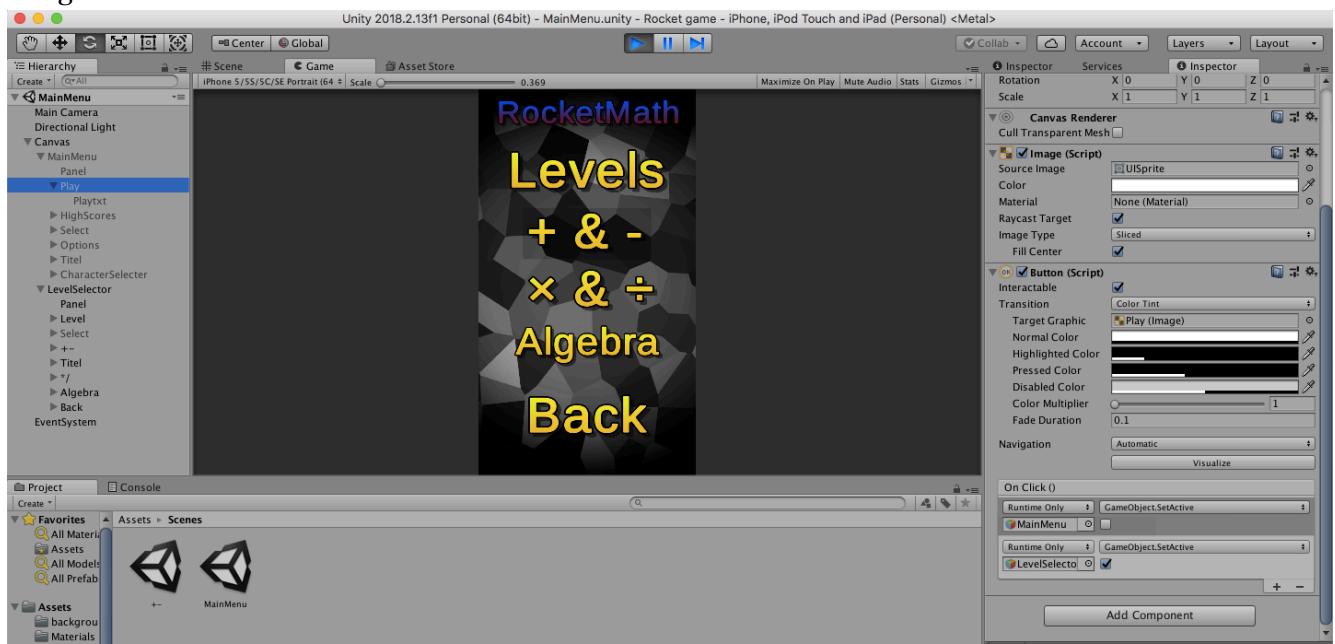
bilag 20



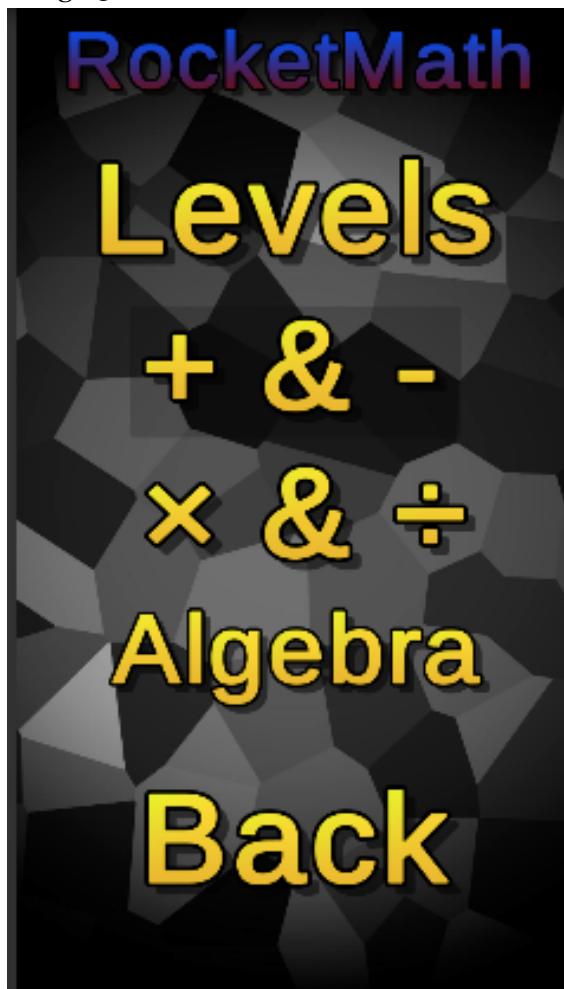
bilag 21



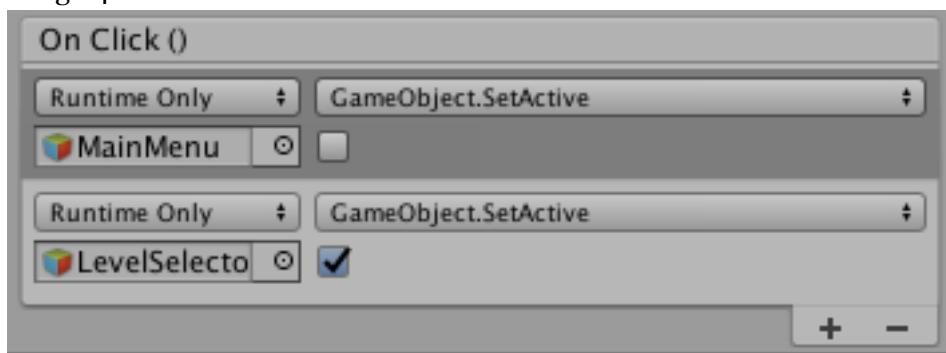
Bilag 22



Bilag 23



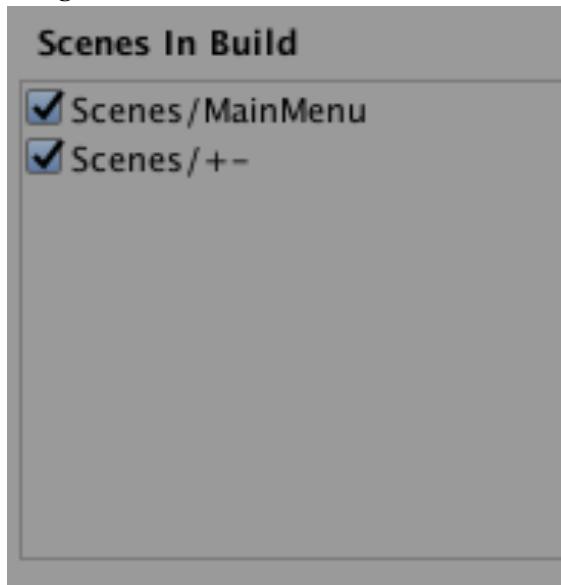
bilag 24



bilag 25

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class MainMenu : MonoBehaviour {
7      public void PlayGame()
8      {
9          SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
10     }
11
12 }
13 }
14 }
```

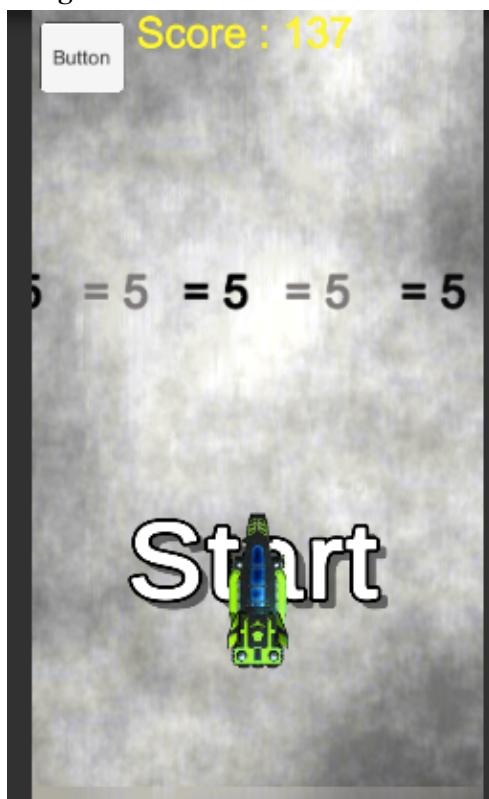
Bilag 26



Bilag 27

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BlockData : MonoBehaviour {
6      public string mathText;
7      public bool rightAnswer;
8      public TextMesh text;
9
10     // Use this for initialization
11     void Start () {
12         rightAnswer = false;
13         text = GetComponentInChildren<TextMesh>();
14     }
15
16     // Update is called once per frame
17     public void SetData (string _mathText, bool _rightAnswer) {
18         mathText = _mathText;
19         rightAnswer = _rightAnswer;
20
21         text = GetComponentInChildren<TextMesh>();
22
23         text.text = mathText;
24     }
25
26 }
27
28 }
```

Bilag 28



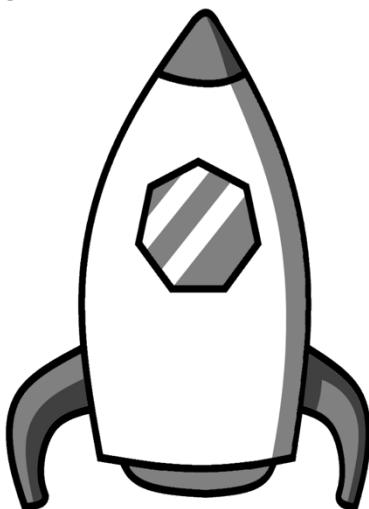
bilag 29

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class AnswerGUI : MonoBehaviour {
6      public TextMesh AnswerText;
7      public TextMesh AnswerText3;
8      public TextMesh AnswerText4;
9      public TextMesh AnswerText5;
10     public TextMesh AnswerText6;
11     public TextMesh AnswerText7;
12     public TextMesh AnswerText8;
13     public TextMesh AnswerText9;
14     private RandomMath2 RandomEqualsToScript2;
15     //public TextMesh AnswerText;
16
17     // Use this for initialization
18     void Start() {
19         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
20         AnswerText.text = " = " + RandomEqualsToScript2.diff.ToString();
21         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
22
23         AnswerText3.text = " = " + RandomEqualsToScript2.diff.ToString();
24         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
25
26         AnswerText4.text = " = " + RandomEqualsToScript2.diff.ToString();
27         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
28
29         AnswerText5.text = " = " + RandomEqualsToScript2.diff.ToString();
30         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
31
32         AnswerText6.text = " = " + RandomEqualsToScript2.diff.ToString();
33         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
34
35         AnswerText7.text = " = " + RandomEqualsToScript2.diff.ToString();
36         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
37
38         AnswerText8.text = " = " + RandomEqualsToScript2.diff.ToString();
39         RandomEqualsToScript2 = gameObject.GetComponent<RandomMath2>();
40
41         AnswerText9.text = " = " + RandomEqualsToScript2.diff.ToString();
42     }
43 }
44 }
```

bilag 30

```
5  public class Spawn : MonoBehaviour {
6      //første bogstav er lig med hvad resultatet er (a=1 b=2), andet og tredje bogstav er lig med :
7      public GameObject mat1;
8
9      // Use this for initialization
10     void Start () {
11         StartCoroutine(WaitTime());
12     }
13
14     // Update is called once per frame
15     void Update () {
16         //
17     }
18
19     IEnumerator WaitTime()
20     {
21         for (; ; )
22         {
23             float tmp = this.transform.position.x;
24             Vector3 pos = new Vector3(this.transform.position.x + Random.Range(-0.1f, 4.5f),
25                                         this.transform.position.y,
26                                         this.transform.position.z);
27
28             // Fang scriptet på gameobjektet
29             GameObject clone = (GameObject)Instantiate(mat1, pos, mat1.transform.rotation);
30             RandomMath2 cloneScript = clone.GetComponentInChildren<RandomMath2>();
31
32             if(clone.tag == "AnswerText")
33             /*
34                 print("Hej");
35                 Vector3 pos2 = new Vector3(0,
36                                         this.transform.position.y,
37                                         this.transform.position.z);
38                 clone.transform.position = pos2;
39             */
34
35             cloneScript.EqualsToo = Random.Range(1, 10);
36             // Ret EqualsTo til random tal
37
38             yield return new WaitForSeconds(5); //lav en public int til at styre waitforseconds t
39         }
40     }
41 }
```

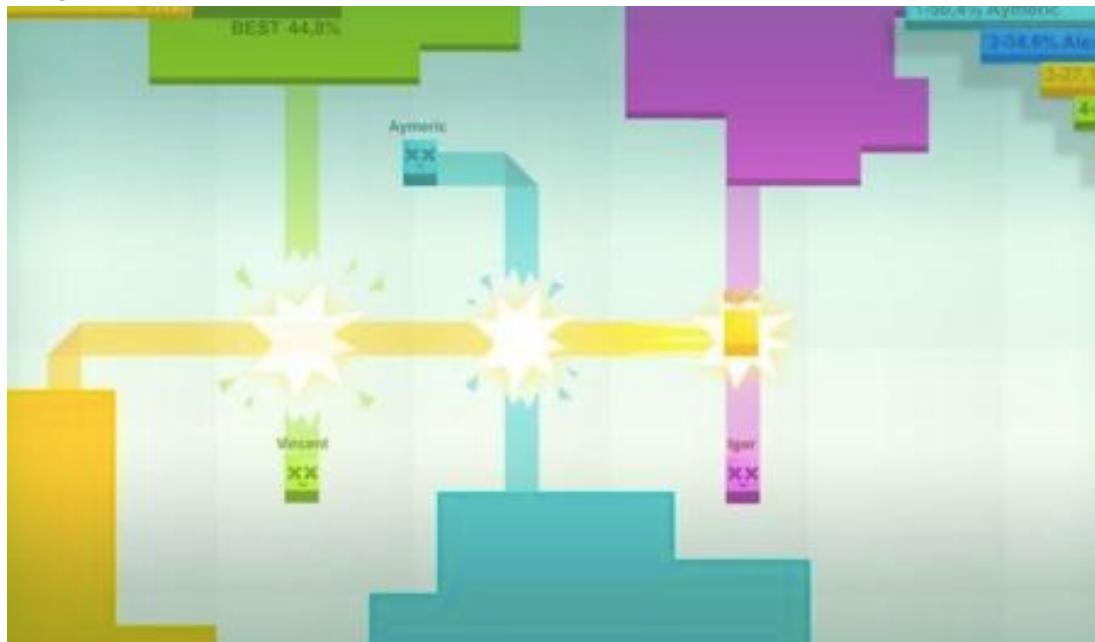
bilag 31



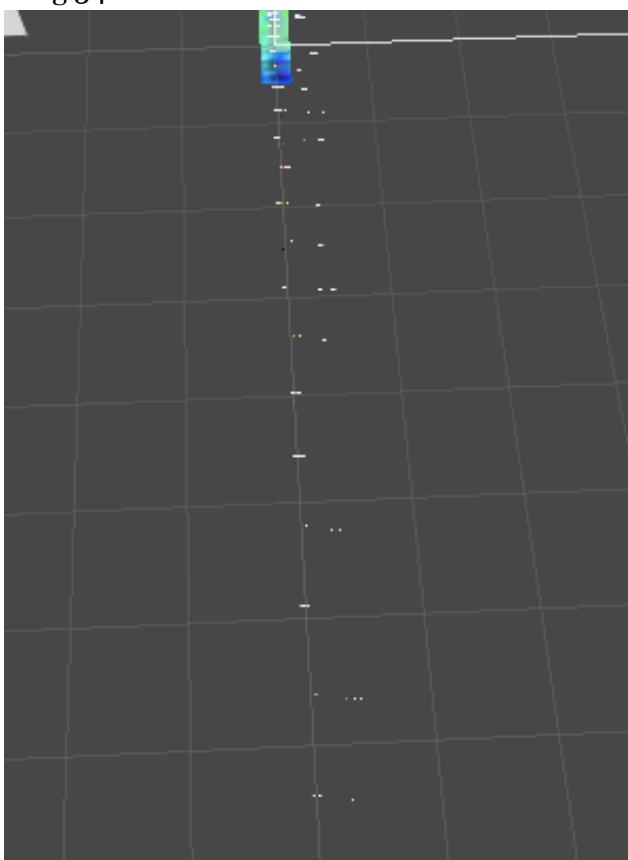
bilag 32



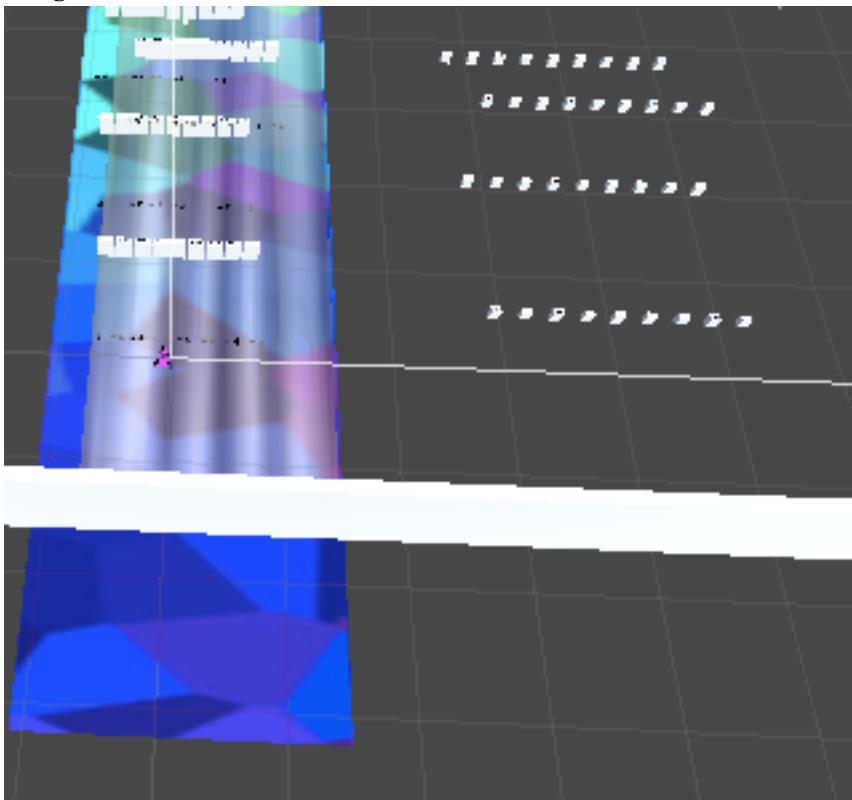
bilag 33



bilag 34



bilag 35



bilag 36

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class DeleteOnColision : MonoBehaviour {
6
7      void OnTriggerEnter(Collider other)
8      {
9          Destroy(other.gameObject);
10     }
11 }
```

bilag 37

```
static bool IsPrime(int n)
{
    if (n > 1)
    {
        return Enumerable.Range(1, n).Where(x => n % x == 0)
                           .SequenceEqual(new[] { 1, n });
    }

    return false;
}
```

bilag 38

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class GameMenu : MonoBehaviour {
7      // amount of text components
8      public Text[] numbersText = new Text[5];
9      public int[] numbers = new int[5];
10
11     public float time;
12     public float roundLength;
13
14     public Text timeText;
15
16
17
18     // Use this for initialization
19     void Start () {
20         time = roundLength;
21
22     }
23
24
25     // Update is called once per frame
26     void Update () {
27         time += 15*Time.deltaTime;
28         timeText.text = " " + time.ToString("0");
29
30         if(time <=0)
31         {
32             time = roundLength;
33             randomNums();
34
35         }
36
37     }
38     void randomNums ()
39     {
40         for (int i = 0; i < numbers.Length; i++)
41         {
42             numbers[i] = Random.Range(0, 10);
43             numbersText[i].text = numbers[i].ToString();
44         }
45     }
46 }
47
```

bilag 39

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class pause : MonoBehaviour {
6      //public int timevalue;
7
8      public void SetTime(int _timeValue){
9
10         Time.timeScale = _timeValue;
11
12     }
13 }
```

bilag 40

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class GoHome : MonoBehaviour {
7
8      public void GoToMenu(string target)
9      {
10          SceneManager.LoadScene(target);
11      }
12
13 }
```

bilag 41



bilag 42

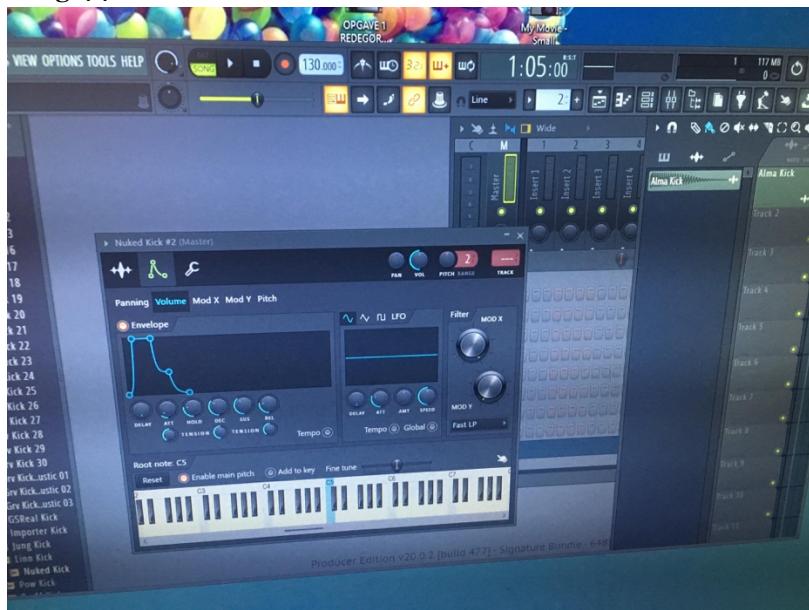
---

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class WrongAnswerCollider : MonoBehaviour
7  {
8      public Slider healthbar;
9      public GameObject GameOver;
10     public Text score;
11     public Text gameoverscore;
12     public GameObject CloseScoreUI;
13
14     public void Start()
15     {
16         GameOver = GameObject.FindGameObjectWithTag("gameover");
17         score = GameObject.Find("Score").GetComponent<Text>();
18         gameoverscore = GameObject.Find("ScoreGameOver").GetComponent<Text>();
19
20         GameOver.SetActive(false);
21
22
23         CloseScoreUI = GameObject.FindGameObjectWithTag("Score");
24         CloseScoreUI.SetActive(true);
25     }
26
27     public void OnTriggerEnter(Collider col)
28     {
29         if (col.gameObject.tag == "wrong")
30         {
31             healthbar.value -= 50;
32
33         }
34         else if (col.gameObject.tag == "right")
35         {
36             healthbar.value += 10;
37
38         }
39         if (healthbar.value <= 1)
40         {
41             GameOver.SetActive(true);
42             gameoverscore.text = score.text;
43             CloseScoreUI.SetActive(false);
44             Destroy(gameObject);
45             Time.timeScale = 0;
46         }
47     }
48
49 }
```

Bilag 43



bilag 44



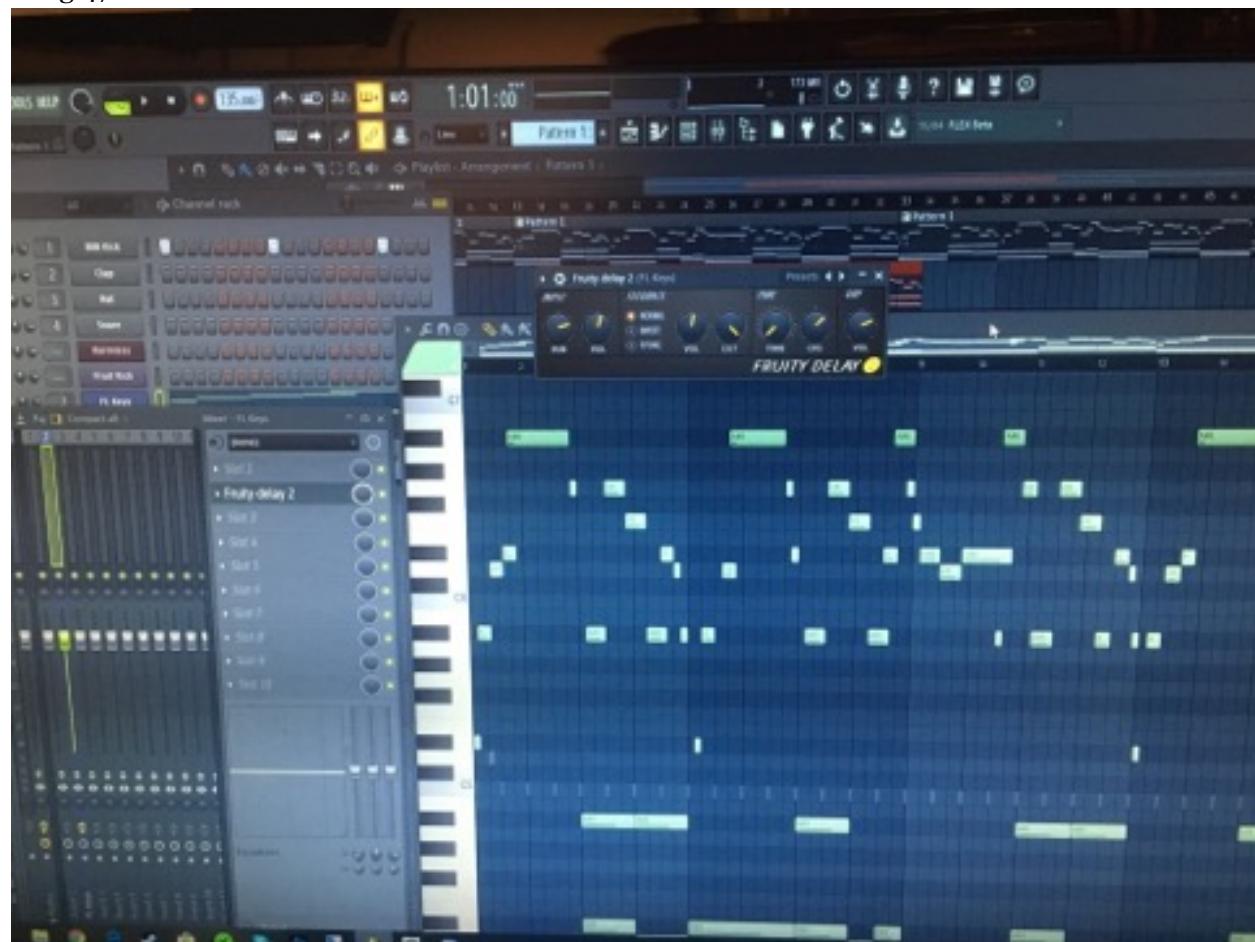
### Bilag 45

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Audio : MonoBehaviour {
6      public AudioClip WrongSound; //file
7      public AudioSource WrongSource; //object
8      public AudioClip CorrectSound;
9      public AudioSource CorrectSource;
10
11     // Use this for initialization
12     // Update is called once per frame
13     public void OnTriggerEnter(Collider col)
14     {
15         if (col.gameObject.tag == "wrong")
16         {
17             WrongSource.clip = WrongSound;
18             WrongSource.Play();
19         }
20         else if(col.gameObject.tag == "right")
21         {
22             CorrectSource.clip = CorrectSound;
23             CorrectSource.Play();
24         }
25     }
26 }
27
28 }
29
```

### Bilag 46

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Threading;
4  using UnityEngine;
5
6  public class SpawnSlowdownPowerup : MonoBehaviour
7  {
8      //første bogstav er lig med hvad resultatet er (a=1 b=2), andet og tredje bogstav er lig med nummer
9      public GameObject _Object;
10     public int min; //how often it spawns
11     public int max;
12     public int SlowDownValue;
13     public int Duration;
14
15
16     // Use this for initialization
17     void Start()
18     {
19         StartCoroutine(WaitTime());
20     }
21
22
23     // Update is called once per frame
24     IEnumerator OnTriggerEnter(Collider col)
25     {
26
27         if (col.gameObject.tag == "SlowDown")
28         {
29             Time.timeScale = SlowDownValue / 100f;
30             Destroy(col.gameObject);
31             yield return new WaitForSeconds(Duration);
32             Time.timeScale = 1;
33         }
34     }
35
36     IEnumerator WaitTime()
37     {
38         for ( ; ; )
39         {
40             yield return new WaitForSeconds(Random.Range(1, 10));
41             float tmp = this.transform.position.x;
42             Vector3 pos = new Vector3(this.transform.position.x + Random.Range(-1.9f, 2.5f),
43                                     this.transform.position.y,
44                                     this.transform.position.z + 20);
45
46             // Fang scriptet på gameobjektet
47             GameObject clone = (GameObject)Instantiate(_Object, pos, _Object.transform.rotation);
48             // Ret EqualsTo til random tal
49             yield return new WaitForSeconds(Random.Range(min, max));
50         }
51     }
52 }
```

bilag 47



bilag 48



bilag 49

```
18     private void Start()
19     {
20         highscore.text = PlayerPrefs.GetFloat("highscore", 0).ToString();
21     }
```

bilag 50

```
34 }
35 if(time > PlayerPrefs.GetFloat("highscore", 0)){
36     PlayerPrefs.SetFloat("highscore", time);
37     highscore.text = " " + time.ToString("0");
38 }
39
```

bilag 51

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PlayerColor : MonoBehaviour {
6      private Renderer rend;
7
8      [SerializeField]
9      public Color colorToTurnTo = Color.white;
10
11     // Use this for initialization
12     private void Start () {
13         rend = GetComponent<Renderer>();
14         rend.material.color = colorToTurnTo;
15     }
16
17 }
18
```

bilag 52

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class TouchControls : MonoBehaviour {
6      int i = 0;
7      private float screenWidth;
8
9      private void Start()
10     {
11         screenWidth = Screen.width;
12     }
13     void Update()
14     {
15         if (Input.touchCount > 0 && Input.GetTouch(i).position.x > screenWidth / 2 && this.transform.position.x < 2.5)
16         {
17             this.transform.Translate(0.125f, 0, 0);
18         }
19         else if (Input.touchCount > 0 && Input.GetTouch(i).position.x < screenWidth / 2 && this.transform.position.x > -1.9)
20         {
21             this.transform.Translate(-0.125f, 0, 0);
22         }
23     }
}
```

bilag 52

$7+6 =$ ____	$5 \cdot 3 =$ ____	$1+2 =$ ____	$9+4 =$ ____
$6 \cdot 1 =$ ____	$7-1 =$ ____	$6 \cdot 6 =$ ____	$7-6 =$ ____
$3-1 =$ ____	$9 \cdot 5 =$ ____	$8+6 =$ ____	$9 \cdot 2 =$ ____
$2+1 =$ ____	$7 \cdot 5 =$ ____	$4 \cdot 1 =$ ____	$4-3 =$ ____
$6-0 =$ ____	$4 \cdot 7 =$ ____	$0 \cdot 2 =$ ____	$6 \cdot 9 =$ ____
$4+2 =$ ____	$5 \cdot 5 =$ ____	$0 \cdot 4 =$ ____	$6 \cdot 5 =$ ____
$3-2 =$ ____	$5-1 =$ ____	$6-4 =$ ____	$3-0 =$ ____
$2 \cdot 4 =$ ____	$2+2 =$ ____	$3+6 =$ ____	$8-5 =$ ____
$3+5 =$ ____	$9 \cdot 7 =$ ____	$7-1 =$ ____	$4+8 =$ ____
$1+1 =$ ____	$9+6 =$ ____	$0 \cdot 1 =$ ____	$5 \cdot 6 =$ ____
$9 \cdot 0 =$ ____	$6-3 =$ ____	$4 \cdot 9 =$ ____	$9 \cdot 9 =$ ____
$2 \cdot 8 =$ ____	$2+6 =$ ____	$3-2 =$ ____	$9-5 =$ ____
$4 \cdot 5 =$ ____	$0+9 =$ ____	$3-0 =$ ____	$9-3 =$ ____
$7+4 =$ ____	$0 \cdot 5 =$ ____	$7-7 =$ ____	$5+4 =$ ____
$5+0 =$ ____	$3+3 =$ ____	$3+8 =$ ____	$7 \cdot 0 =$ ____
$3-1 =$ ____	$8+7 =$ ____	$4-1 =$ ____	$3 \cdot 4 =$ ____
$5 \cdot 1 =$ ____	$3+7 =$ ____	$2+9 =$ ____	$9 \cdot 1 =$ ____
$5-2 =$ ____	$8+9 =$ ____	$7-5 =$ ____	$2 \cdot 7 =$ ____
$7+3 =$ ____	$8 \cdot 3 =$ ____	$6-4 =$ ____	$8-5 =$ ____
$8-0 =$ ____	$8 \cdot 1 =$ ____	$6+8 =$ ____	$2-0 =$ ____
$7-2 =$ ____	$8 \cdot 2 =$ ____	$4 \cdot 0 =$ ____	$5-2 =$ ____
$9 \cdot 3 =$ ____	$0-0 =$ ____	$9-1 =$ ____	$8-4 =$ ____
$1 \cdot 8 =$ ____	$9-7 =$ ____	$1+6 =$ ____	$9 \cdot 8 =$ ____
$8+0 =$ ____	$1-0 =$ ____	$4 \cdot 4 =$ ____	$0 \cdot 7 =$ ____
$8 \cdot 8 =$ ____	$6 \cdot 2 =$ ____	$6-0 =$ ____	$7+8 =$ ____