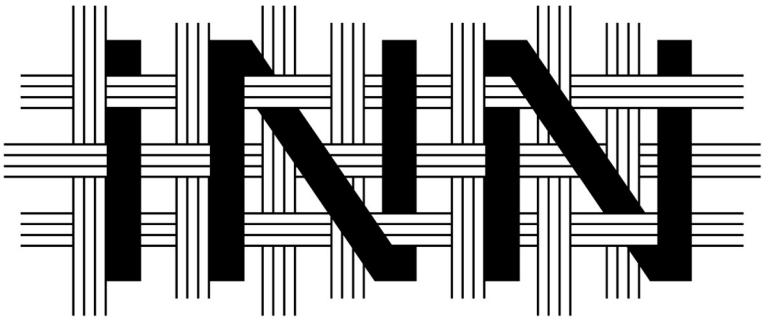



Høgskolen i Innlandet

Fakultet for Lærerutdanning og Pedagogikk

**Thomas Grønning**


## Entreprenøriell Masteroppgave

# Undervisningsopplegg i Sannsynlighet med Blokkprogrammering (USB)

Teaching plan for Probability using Block Programming (TPB)

Master i Realfagenes Didaktikk

2MROPPG2D

**2024**

**Forord**

Jeg retter en stor takk går til alle som har bidratt til å gjøre denne masteroppgaven mulig: Ansatte ved Høgskolen i Innlandet, gode kolleger, støttende ledere, tålmodige med- og motstudenter og ikke minst min nærmeste familie som har hjulpet meg over alle hindre. Til slutt en spesiell takk for gode støttestrukturer og ukuelig optimisme til min veileder Jonas Oskarsson.

**Sammendrag:**

Denne oppgaven presenterer et didaktisk produkt. Produktet mitt er et undervisningsopplegg der elevene gjennom fire programmer får erfaring med eksperimentell sannsynlighet og store talls lov. I programmene vil de utføre simuleringer, bygge et blokkbasert program og eksperimentere med ukjent sannsynlighet. Undervisningsopplegget er tilpasset metodikken rundt tenkende klasserom (Liljedahl, 2020), men kan også gjennomføres på en tradisjonell måte. Opplegget er tenkt brukt i matematikkundervisningen på 9. trinn, og er tilpasset lærere og elever både med og uten tidligere erfaringer med blokkprogrammering. Oppgavene har lav inngangsterskel og høy takhøyde. Det skiller seg fra tradisjonelle undervisningsopplegg ved at det innebærer ekte simuleringer, hvor det ikke er mulig å beregne sannsynlighet ved regning. Elevene vil måtte gjennomføre eksperimenter og anslå sannsynligheter. Undervisningsopplegget inneholder en lærerveiledning med informasjon om teoretisk grunnlag, gjennomføring, plattform og programkode, og mål for opplegget. Produktet er utviklet gjennom en designprosess med utvikling, utprøving og revisjon der observasjon i klasserom og intervju av lærere inngår. Det endelige produktet er resultatet etter denne prosessen.

**Abstract:**

This thesis presents a didactic product. My product is a teaching plan where students, through four programs, gain experience with experimental probability and the law of large numbers. In the programs, they will perform simulations, build a block-based program, and experiment with unknown probability. The teaching plan is adapted to the methodology around thinking classrooms (Liljedahl, 2020), but can also be implemented in a traditional way. The plan is intended for use in mathematics teaching at the 9th grade level, and is adapted for teachers and students both with and without prior experience with block programming. The tasks have a low entry threshold and high ceiling. It differs from traditional teaching plans in that it involves real simulations, where it is not possible to calculate probability by calculation. The students will have to conduct experiments and estimate probabilities. The teaching plan includes a teacher's guide with information about theoretical framework, implementation, environment and program code, and goals for the plan. The product was developed through a design process with development, testing and revision, including classroom observation and teacher interviews. The final product is the result after this process.

**INNHOLDSFORTEGNELSE**

## OVERSIKT OVER FIGURER



## OVERSIKT OVER TABELLER

# 1. Innledning

Programmering har blitt en integrert del av matematikkundervisningen i norske skoler. Fra barneskolen skal elevene lære om programmering, både som et selvstendig mål og som et verktøy for å løse problemer eller forstå konsepter. Elever på 9. trinn i Norge skal simulere utfall og beregne sannsynligheter ved hjelp av programmering i matematikkundervisningen. Valget av læremidler, programmeringsspråk og oppgaver er opp til den enkelte lærer. I denne masteroppgaven presenteres et didaktisk produkt. Det argumenteres for å bruke dette til å undervise i sannsynlighet og store talls lov gjennom blokkprogrammering.

## 1.1 Problemstilling

I denne oppgaven undersøker jeg problemstillingen «Hvordan kan et interaktivt undervisningsopplegg med programmering og sannsynlighet utformes?»

### *1.1.1 Kriterier*

Jeg har lagt følgende kriterier til grunn for utformingen:

- Opplegget skal fokusere på undervisning i sannsynlighet.
- Opplegget skal involvere programmering, men det skal ikke være en ferdig oppskrift for å bygge et program.
- Opplegget skal kunne benyttes interaktivt av brukere, lærere og elever, som ikke har tidligere erfaring med programmering.
- Opplegget skal være tilpasset metodikken rundt tenkende klasserom (Liljedahl, 2020).

Mitt pedagogiske produkt har fått navnet «undervisning i sannsynlighetsregning med blokkprogrammering», USB.

### *1.1.2 Argumentasjon*

Som en lærer med studiepoeng i programmering og lang erfaring med å undervise programmering valgfag i ungdomsskolen, hadde jeg et solid grunnlag for å utvikle et undervisningsopplegg som benyttet programmering. Undervisning i sannsynlighet kan kobles til bruk av programmering, ved å beregne teoretiske sannsynligheter digital og simulere eksperimenter. Læreplanen nevner også eksplisitt programmering som en del av undervisning om sannsynlighet. Kriteriet om at opplegget kan gjennomføres uten kjennskap til programmering var ment for å gi flest mulig muligheten til å prøve ut opplegget og gjennomføre undervisning i programmering. Bruk av metodikken 'tenkende klasserom' er for

tiden aktuelt og kan være en inngangsport for nye brukere denne av metodikken, og av lærere som ønsker å benytte den på digitale oppgaver. Kriteriet om at opplegget ikke skal være en ferdig oppskrift for å kode et program var ment for å gjøre opplegget mer fleksibelt og gi elevene valg under arbeidet med opplegget.

## 1.2 Hensikt

Hensikten med det didaktiske produktet er å tilby et undervisningsopplegg til lærere som ønsker å bruke programmering for å styrke elevenes forståelse av sannsynlighet. Opplegget skal være åpent tilgjengelig digitalt og kunne fritt deles mellom brukere. Det vil være tilgjengelig på høgskolens sider og på ulike grupper for matematikklærere. Opplegget skal også være nyskapende, og bruke programmering på en ny måte i sannsynlighetsundervisningen.

### *1.2.1 Programmene*

Program 1 (https://scratch.mit.edu/projects/944693740) er et spill som simulerer pilkast. Pilene har en varierende sannsynlighet for å treffe en blink. Elevene skal vurdere sannsynligheten for å treffe i hver runde. Hensikten er å erfare at flere forsøk gir økt sjanse for å gjette riktig. Elevene må selv beregne sannsynligheten ut fra antall treff og antall forsøk, så en annen hensikt er å øve elever i å beregne sannsynlighet.
Program 2 (https://scratch.mit.edu/projects/979669167) Simulerer ballkast. Dette er et ferdigbygget program, og fungerer som en simulering. Hensikten er å la elevene beregne sannsynlighet ved å benytte store talls lov. En ekstra hensikt med programmet er å vise et eksempel på hvordan et program som simulerer en kastebevegelse kan bygges.
Program 3 (https://scratch.mit.edu/projects/953112745) er et puslespill. Programmet kan ikke kjøres når det lastes inn fra lenken. Elevene skal sette sammen kodeblokker og velge parametere slik at de ender opp med sin egen simulering. Hensikten er å bruke programmering til å simulere hendelser, og å anslå sannsynligheten for en hendelse som ikke er mulig å beregne aritmetisk.
Program 4 (https://scratch.mit.edu/projects/999688305) er et ferdig spill hvor elevene selv kaster og prøver å treffe et mål. Spillet regner ut treffprosent automatisk. Hensikten er å åpne for en diskusjon om når man kan bruke prinsippet om store talls lov og ikke.

### *1.2.2 Lærerveiledningen*

Lærerveiledningen består av fem deler (Vedlegg 2). Første del er en introduksjon av opplegget, hvor utstyr, hensikt, læringsmål, bakgrunnskunnskap og informasjon om fag og emne klargjøres. Andre del går nærmere inn på innholdet og hensikten, og har en innføring i den delen av Liljedahls metodikk som vil bli benyttet i opplegget. Tredje del inneholder 'skript' (forslag til muntlig formulering i klasserommet) og fremgangsmåte til gjennomføring av opplegget. Her står også hint og utvidelser til elevgrupper som trenger ekstra hjelp eller større utfordringer. Fjerde del inneholder løsningsforslag og nærmere informasjon om programmene. Femte del er et oppgaveark. Dette kan deles ut til elever dersom man ønsker å bruke tradisjonelle undervisningsformer. Lærerveiledningen oppfyller dermed flere funksjoner som kunne vært vanskelig å bygge inn i programmene direkte. Koblingen til Liljedahls metodikk blir klar gjennom spesielt skript og hint og utvidelser. Fasit og tips til programmene kan hjelpe lærere med mindre erfaring med blokkprogrammering å gjennomføre opplegget på en god måte. Det er også relevant for andre brukere å få informasjon om læringsmål og hvilket trinn opplegget er tenkt brukt på.

## 1.3 Prinsipper

Kriteriene for opplegget har medført at opplegget har klare prinsipper. Et prinsipp er bruken av blokkprogrammering og at elevene skal pusle med kodeblokker. Tidligere forskning rundt dette beskrives i avsnitt (2.1.5). Et annet prinsipp er tilgjengelighet. Opplegget skal kunne brukes av alle, uansett tidligere erfaring med blokkprogrammering eller Liljedahls metode. Opplegget er derfor utviklet med tanke om å ha lav terskel og stor takhøyde, og teorien rundt dette drøftes i (2.1.2). Opplegget antar imidlertid at læreren har en faglig bakgrunn i matematikk, slik at store talls lov og utregning av teoretisk og eksperimentell sannsynlighet ikke gjennomgås i lærerveiledningen til opplegget. Et tredje prinsipp er at opplegget skal være mest mulig åpent for utforskning og at elevene skal ende opp med egne løsninger. For å få til dette er opplegget utviklet for å legge til rette for å prøve og feile, teste ut og justere programmer. Tidligere forskning rundt dette presenteres i (2.1.3). Det siste hovedprinsippet er Liljedahls metode. Metoden har i seg selv ingen direkte sammenheng med programmering og sannsynlighet, men metodikken er relativt ny i Norge og blir prøvd ut i mange klasserom nå for tiden. Opplegget søker derfor å bruke metodikken sammen med programmering. Metodikken setter opp 14 praksiser for matematikkundervisning, og jeg drøfter hvilke av disse jeg har passet inn i opplegget i avsnitt (2.1.4).

# 2. Behov for det didaktiske produktet:

I dette kapitlet klargjøres det hvordan det didaktiske emnet sannsynlighet fremkommer i lærerprofesjonen, og hvilken tidligere forskning undervisningsopplegget støtter seg på.

## 2.1 Lærerprofesjon

I denne delen legger jeg frem hvordan emnet sannsynlighetsregning fremkommer i læreplan, og ser på hvordan det tidligere har blitt gjennomført undervisning i dette sannsynlighet og programmering.

### *2.1.1 Læreplan og styringsdokumenter*

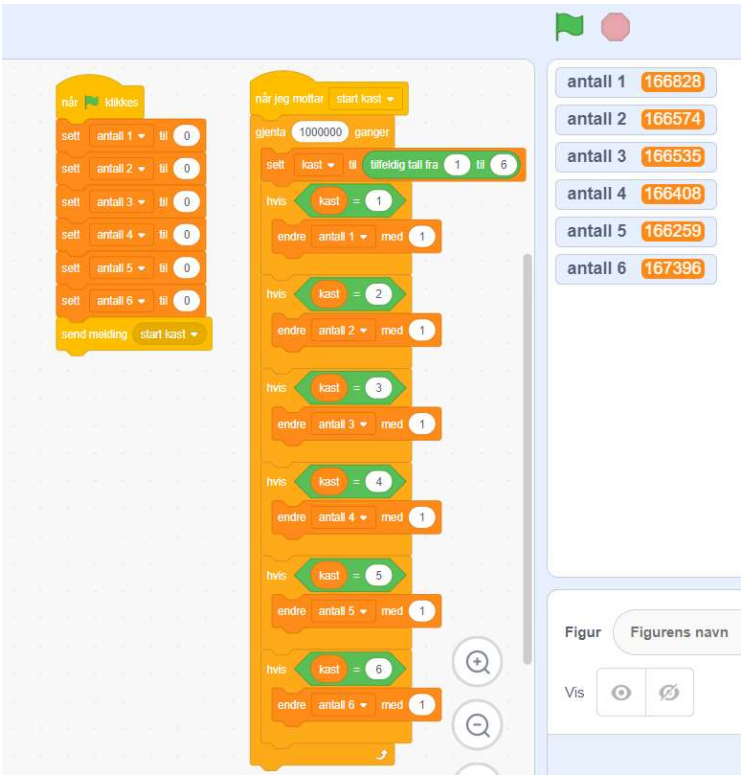
I læreplan for matematikk, LK-20, kompetansemål etter 9. trinn (Kunnskapsdepartementet, 2019) står at elevene skal kunne «beregne og vurdere sannsynlighet i statistikk og spill» og «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering». Videre står det under grunnleggende ferdigheter at «digitale ferdigheter innebærer å kunne bruke programmering til å utforske og løse matematiske problemer» (2019). I utdanningsdirektoratet sin artikkel om algoritmisk tenkning (2019) nevnes det en stegvis problemløsningsmetode og bruk av datamaskin.

### *2.1.2 Sannsynlighet i undervisning.*

Sannsynlighet i matematikk kan forstås som i hvor stor grad det er trolig at et utfall kan forekomme. I sannsynlighetsregning kan man beregne teoretisk sannsynlighet ved å se på forholdet mellom antall gunstige og mulige utfall. Eksperimentell sannsynlighet dreier seg om å beregne sannsynligheter som resultat av forsøk. Man gjennomfører flere forsøk eller hendelser og sammenlikner antall ganger forsøket lykkes med totale antall forsøk. I forbindelse med eksperimentell sannsynlighet kommer prinsippet om store talls lov inn. Denne loven sier at dersom antall forsøk økes, så vil observert sannsynlighet nærme seg den ekte verdien for sannsynlighet. En følge av denne loven er også at store avvik mellom observert sannsynlighet og ekte sannsynlighet vil skje oftere når antall forsøk er lavere (Konold & Kazak, 2008, s.5-6). Programmering er et verktøy som gir muligheter til å drive undervisning i eksperimentell sannsynlighet og dermed store talls lov gjennom muligheten til å gjenta et forsøk mange ganger, og ved å kunne gi dynamisk og fleksibel representasjon av data (Konold & Kazak, 2008, s.31).

### *2.1.3 Tradisjonelle programmeringsoppgaver*

En tradisjonell tilnærming til å nå målet om simulering og programmering kan være å lage et dataprogram som simulerer terningkast. Et eksempel fra et læreverk i matematikk på 9. trinn (Kleivdal, 2020 s. 96-97) er en oppgave der elevene skal lage et program som simulerer kast av en terning med seks sider. Simuleringen gjentas flere ganger og programmet skal telle opp antall utfallene etter 100, 1000 og 10000 iterasjoner. Erfaringen for elevene blir at de trenger mange forsøk for å komme frem til forventet resultat. Dette presenteres i teksten som store talls lov. Et tilsvarende eksempel er hentet fra en digital læringsplattform (Kikora, u.å.). Oppgaven går ut på å stegvis bygge opp et tekstbasert program som på samme måte simulerer terningkast 100, 1000 og 10000 ganger. Spørsmålet i denne oppgaven er «Hva observerer du?» og elevene tar stilling til utsagnene «Når man kaster svært mange ganger, nærmer resultatet seg sannsynligheten» og «Når man kaster få ganger, kan resultatet være et stykke unna sannsynligheten». Et siste eksempel er hentet fra Kunnskapsfilm (2021). Oppgaven sier «Lag en egendesignet terning som ikke har seks sider. […] Du skal lage et program som du kan bruke til å simulere terningkast […] og finne sannsynligheten for de forskjellige utfallene». Figur 1 viser et forslag til kode for å simulere fordeling av terningkast i Scratch.




Figur 1. *Terningkast i Scratch.*(fra https://scratch.mit.edu, selvlaget program).[1]

[1] Alle figurer med kodeblokker er selvlagede program fra Scratch hvis ikke annet er oppgitt

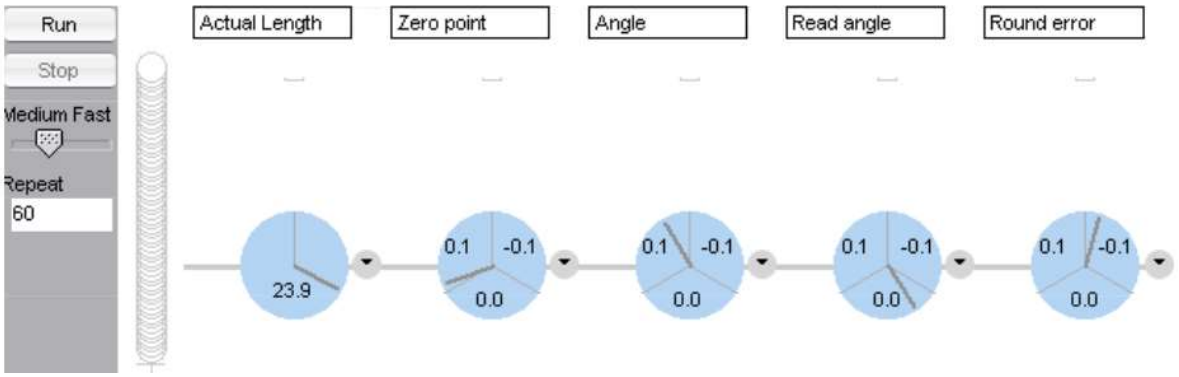
I alle disse eksemplene gjennomføres simuleringer, og antall forsøk kan økes til millioner. Men elevene kan selv beregne den teoretiske sannsynligheten, og oppgaven blir dermed kun å se at resultatet stemmer med en kjent fasit. Et annet poeng ved disse tre eksemplene er at de ikke viser det ikke er visuell representasjon av terninger som triller, annet enn tall på en skjerm.

### *2.1.4 Grad av frihet i programmering*

Med grad av frihet menes her hvor åpen eller styrt en oppgave er. Begrepet har vært kjent lenge, og vært benyttet om utforskende arbeid i naturfag, for eksempel hos Gyllenpalm et al (2010, s.48) men kan også brukes om programmering. Kort sagt deles en oppgave i de tre delene spørsmål, metode og resultat. Hver av disse kan være åpne eller gitt. Grader av frihet i oppgaven tilsvarer antall deler som er åpne. En aktivitet hvor alle delene er lukket har frihetsgrad null. Et eksempel kan være en oppgave som «Følg denne oppskriften til programmet, kjør det med 1000 og 10000 repetisjoner. Når er resultatet nærmest forventet verdi?» har frihetsgrad 0. Flere grader av frihet betyr at undervisningen blir mer utforskende og elevstyrt. Innen programmering er det indikasjoner på at en åpen tilnærming uten et støttende rammeverk rundt elevene virker ineffektivt (Kurland & Pea, 1985). I en studie om elevers arbeid med koding av fysiske objekter (Flø & Zambrana, 2024) ble det vist til gode resultater ved semi-styrt undervisning, eller moderat styrt utforskende undervisning, spesielt for elever som er uerfarne i bruk av programmering

### *2.1.5 Relevans*

Et viktig mål for utviklingsarbeid er at det bør ende opp men noe nyskapende. USB benytter programmering til å undersøke store talls lov i sannsynlighet. Dette er i seg selv ikke nytt. Men jeg har ikke funnet noen eksempler på opplegg med programmering som bruker figurer i bevegelse og registrering av bom eller treff til å belyse store talls lov. Et internettsøk ga for det meste oppgaver med å trille terninger (f.eks Espens klasserom, 2024) (Koding i skolen, 2024). Det opplegget jeg har funnet som likner mest på USB ble presentert av Konold & Kazak (2008), hvor en rekke dreibare skiver som ga ulike avlesninger ble brukt til å simulere måleusikkerhet og store talls lov i naturfag.




Figur 2. *Simulering med tilfeldig utfall (fra https://escholarship.org/uc/item/38p7c94v)*

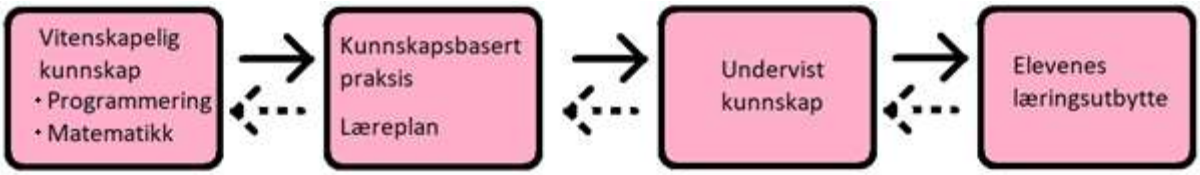
USB kombinerer også Liljedahls metodikk med en programmeringsaktivitet. Dette har jeg ikke klart å finne noen eksempler på, verken gjennom bibliotek- eller internettsøk..

## 2.2 Tidligere forskning

I denne delen viser jeg til sentrale studier og teoretisk grunnlag som har ligget til grunn for utviklingen av USB.

### *2.2.1 Teoretisk rammeverk*

Rammeverket for denne studien er hentet fra Kilhamn & Rolandsson (2023), hvor de oppsummerer erfaringene etter innføringen av programmering som pensum i svenske skoler i 2017. Disse erfaringene blir drøftet i lys av et analytisk rammeverk som er basert på teorien om didaktisk transposisjon (Chevallard, 2006). Transposisjon refererer til prosessen der vitenskapelig kunnskap blir transformert gjennom læreplan og lærernes undervisning og videre til elevers læringsutbytte.




Figur 3. *Didaktisk transposisjon* (fra Kilhamn & Rolandsson, 2023, s. 179, min oversettelse)

Med grunnlag i denne teorien ble det utviklet et rammeverktøy som kan oppsummeres i følgende tabell over sammenhengen mellom praksis og logos:

| | | | |
|---|---|---|---|
| **Praksis 1**<br><br>Valg av oppgaver og innhold | Oppgaven er å lære seg å kommunisere med hverandre og datamaskinen.<br><br>I fokus: Betydningen av symboler og språkets syntaks. | Oppgaven består av utfordrende programmeringsoppgaver.<br><br>I fokus: datalogiske problemløsningsprosesser som feilsøking, fikling og remiksing | Oppgaven består av matematiske problemer, der bruk av programmering kan bidra til løsningen.<br><br>I fokus: klassiske problemløsnings-prosesser og strategier. |
| **Praksis 2**<br><br>Valg av språk og miljø | Språk og miljø velges ut i fra læreplanen.<br><br>Tiden brukes på å skape og tolke stegvise instruksjoner (kode) i det aktuelle miljøet. | Visse lærere velger språk og miljø med enkel syntaks for å lette undervisningen. | Lærere er pragmatiske og velger miljø ut i fra sine og elevenes forkunnskaper, samt hvilken type matematisk modellering som etterspørres. |
| **Logos** | Programmering som en del av digital kompetanse<br>  | Programmering som en delmengde av matematikken<br>  | Programmering som et verktøy for å lære seg matematikk<br>  |

Tabell 1. *Sammenhengen mellom praksis og logos* (fra Kilhamn & Rolandsson, 2023, s. 179. Min oversettelse)

Praksis i denne sammenhengen omhandler både hva som blir gjort og hvordan det blir utført. I rammeverket er praksis delt inn i to deler: oppgavene som gis, og hvilke programmeringsspråk og miljøer som benyttes. Logos, eller lære, handler om begrunnelsen bak valgene som blir gjort. Kategorien logos forsøker å svare på hvilke mål som skal oppnås, hensikten med oppgaver og opplegg, og hvordan praksisene kan bidra til å nå disse målene. I lys av dette rammeverket presenteres programmene som er brukt i USB. Målet med USB var å jobbe tredelt med temaet store talls lov, først ved å demonstrere prinsippet bak store talls lov, deretter la elevene bruke prinsippet til å undersøke hendelser eksperimentelt, og til sist studere tilfeller der store talls lov ikke kunne benyttes. De fire programmene USB består av er utviklet for å lede elevene gjennom disse delmålene.

USB sin målsetting er å være benytte alle tre logos illustrert i tabellen for rammeverket. Alle programmene demonstrerer *bruk av programmering som et verktøy for å lære seg matematikk*. Denne bruken er hovedfokus for USB. Å gjennomføre et statistisk eksperiment ti tusen ganger er vanskelig å få til i et klasserom. Programmering er derfor et effektivt verktøy til å illustrere prinsippet med store talls lov. Programmene er også visuelle, og kan simulere pilkast og ballkast ved å vise grafikk. Bruk av programmering gjør det både mulig å gjenta et forsøk tusenvis av ganger, og sørge for at sannsynligheten for treff er konstant gjennom alle disse forsøkene. I et situasjon der mennesker kaster baller i en kurv vil trolig øvelse gjøre at sannsynligheten for gunstig utfall gradvis øker. I alle de fire programmene er *programmering som en digital kompetanse* til stede. Man kan bruke kunnskap om programmeringsmiljøet og kjennskap til kommandoer i programmeringsspråket til å justere og endre på antall forsøk som gjennomføres, eller legge til en variabel og automatisere utregning av sannsynlighet. I program 3 er det fokus på *programmering som en delmengde av matematikken*. Bygging av kodeblokker slik at programmet fungerer er en grunnleggende programmeringsoppgave. I de andre programmene kan koden endres og forbedres, men program 3 er eneste oppgave hvor elevene må programmere for å komme videre og gjennomføre en simulering.

Praksis 1 ser på hvilke oppgaver som er gitt og hvorfor. USB tar for seg store talls lov i sannsynlighetsregning. Dette valget kan knyttes til læreplanen, men også programmeringens mulighet til å løse problemer som krever mange iterasjoner for å komme frem til en akseptabel løsning. I program 1, 2 og 3 er oppgaven å finne en sannsynlighet ved bruk av programmene. I program 3 er oppgaven å programmere i seg selv. I program 4 dreier aktiviteten seg om å finne moteksempler og argumentere bruksområdet til store talls lov.

Praksis 2 handler om programmeringsspråk og miljø. USB har flere programmer med ulike oppgaver, og det er naturlig at programmeringsmiljøet er det samme gjennom hele opplegget. Prinsippene i opplegget om bruk av grafikk og tilgjengelighet ledet til valget av blokkprogrammering og Scratch.

### *2.2.2 LIST-prinsippet*

LIST står for lav terskel, stor takhøyde. Begrepet er oversatt fra engelske «low threshold, high ceiling» (Nrich, 2019). LIST beskriver oppgaver som er enkle å starte med, men som også gir rom for arbeid på et høyere matematisk nivå. Studier på elevgrupper (Andersen et al., 2021, s.13) har vist at blokkbaserte programmeringsspråk gjør programmeringsaktiviteter mer tilgjengelige. I et notat beskriver Selvig (2016, s.17-18) blokkprogrammering som et enkelt og ufarlig alternativ til tradisjonell programmering, og viser til at en slik alternativ inngang til fagstoffet innebærer lav terskel. Det nevnes også at det er få begrensninger på hvor avanserte prosjekter man kan jobbe med i blokkprogrammering, noe som reflekterer stor takhøyde.

### *2.2.3 PRIMM-modellen*

PRIMM beskrives som en tilnærming til planlegging av programmeringsoppgaver og -aktiviteter (Semblance et al, 2019, s.137). PRIMM står for «predict, run, investigate, modify, make», som kan oversettes til forutsi, kjøre, undersøke, endre og skape. Modellen innebærer at elever først eksponeres for fungerende kode. Modellen er stegvis, beskriver en progresjon i arbeidsmåter over tid og er ment som et alternativ til tradisjonelle aktiviteter som nevnt i (2.1.3) hvor elevene kun kopierer kode. PRIMM kan brukes med både tekst- eller blokkbaserte programmeringsspråk. Det kan også sluttes at opplegg som benytter PRIMM vil ha en lav terskel, siden å kjøre en ferdig kode ikke krever spesiell bakgrunnskunnskap.

### *2.2.4 Liljedahls metode*

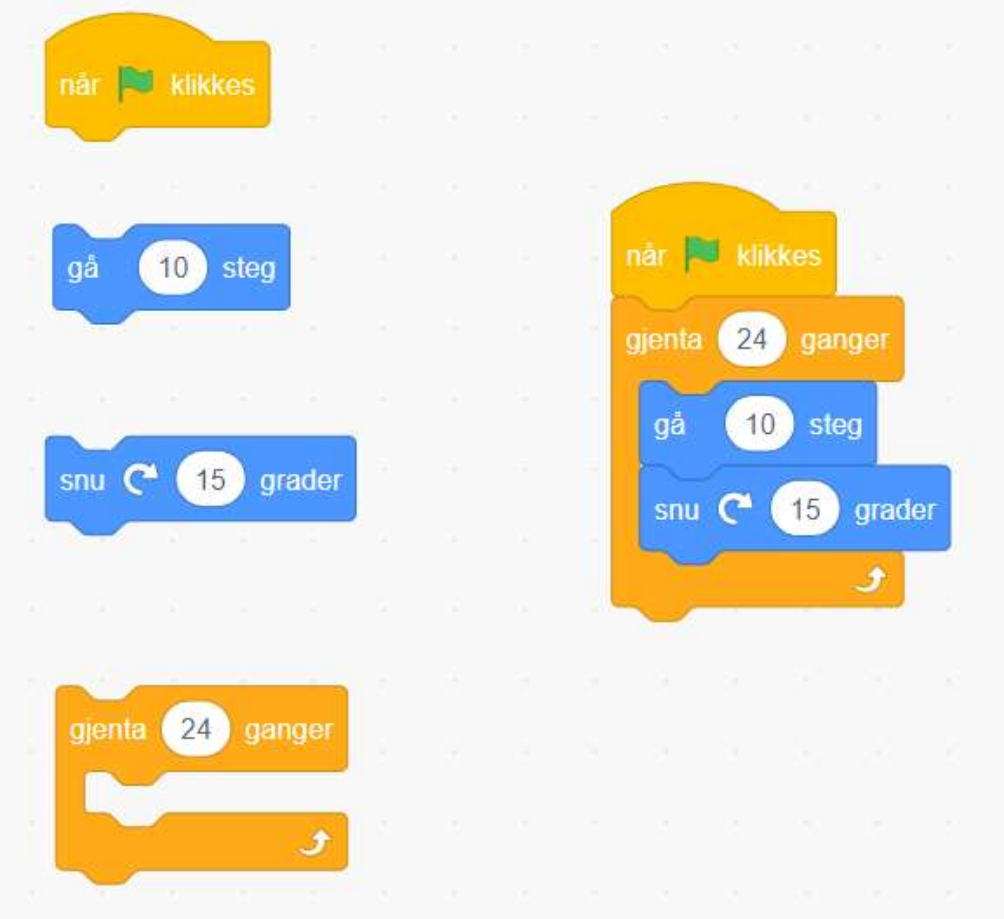
Liljedahl beskriver 14 praksiser for et tenkende klasserom, med mål om å motivere elever til selvstendig problemløsning (Liljedahl, 2020). Praksisene omfatter blant annet typen oppgaver som gis, gruppedannelse, arbeidssted, møblering, spørsmålsbesvarelse, oppgavefordeling, lekser, elevautonomi, hint og utvidelser, undervisningsoppsummering, notattaking, evaluering, formativ vurdering og karaktergivning. Noen praksiser er mindre relevante for

enkeltundervisningsopplegg, som de som omhandler lekser og vurdering over tid. I opplegget fokuseres det på følgende 7 praksiser (min tolkning av Liljedahl 2020):

Praksis 1: Oppgavene vi gir bør føre til at elevene tenker. Det bør være virkelighetsnære oppgaver, og oppgavene bør være åpne i løsningsmetode eller svar.
Praksis 2: Grupper er viktige for læring og utveksling av ideer. Gruppene på tre eventuelt to elever bør deles inn synlig tilfeldig.
Praksis 3: Elevene bør jobbe på loddrette flater er skrift kan viskes ut.
Praksis 5: Lærere bør prøve å ikke svare på en måte slik at elevene stopper å tenke. Lærere svarer kun på spørsmål som leder til at elevene kan tenke videre.
Praksis 6: Oppgavene gis mens elevene står oppreist rundt lærer, før det er gått 5 minutter av timen.
Praksis 9: Underveis bør læreren gi hint eller utvidelser (tilleggsoppgaver), slik at elevene holder seg i flytsonen og fortsetter å jobbe med oppgaven
Praksis 10: Oppsummering bør ta sikte på å få frem elevenes inntrykk, og læreren bør starte med å hente frem resultater og funn alle gruppene har erfart.
I metodedelen 3.1.1 drøftes hvordan praksisene ble inkludert i USB.

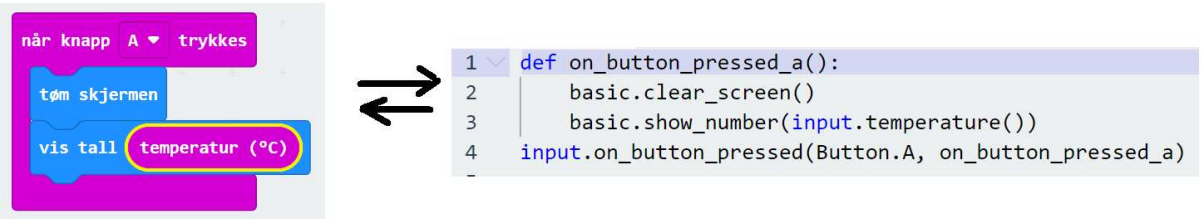
### *2.2.5 Blokkprogrammering*

Blokkprogrammering innebærer bruk av visuelle programmeringsspråk der man setter kodebrikker sammen til fungerende kodeblokker. Utviklingen av blokkprogrammering startet med LOGO, et tekstbasert programmeringsspråk med enkel syntaks, en synlig avatar og egosentriske bevegelsesinstruksjoner. LOGO var også eksplisitt beregnet på unge brukere (Papert, S., 1980). Med egosentrisk menes her at kommandoer fokuserer på objektene som skal bevege seg sitt synspunkt (gå fremover 10 pixler, snu 90 grader til høyre) og ikke fra tredjepersonsperspektiv (‘flytt figuren 10 pixler i retning 45). Videre utvikling har ledet til en rekke blokkbaserte programmeringsspråk slik vi kjenner til i dag, som for eksempel Scratch.



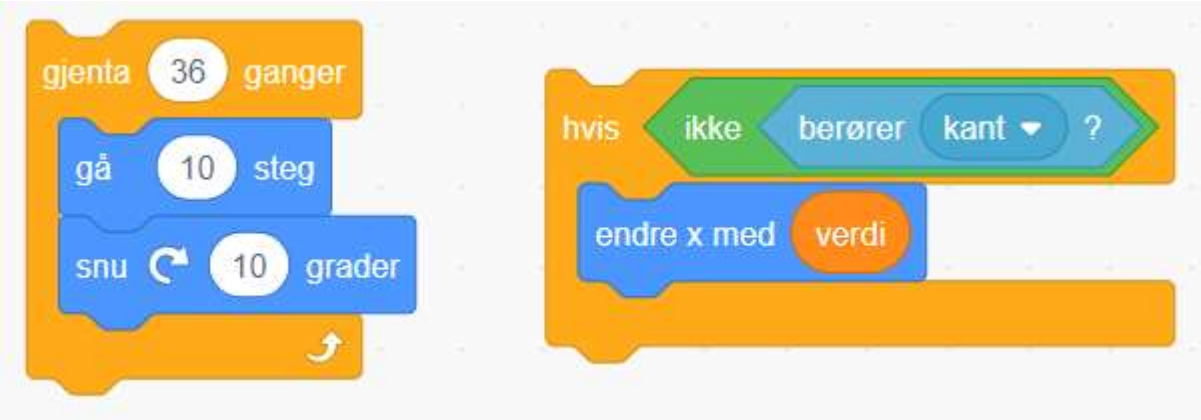

Figur 4. *Kodebrikker og kodeblokker i Scratch*

Scratch er et eksempel på et blokkprogrammeringsspråk designet for barn, og er internettbasert og tilgjengelig på flere plattformer. Det er også oversatt til over 70 språk. (MIT Scratch, u.å.). Bruk av blokker fremfor linjer med kode kan også vurderes i lys av teori om matematisk representasjon (Duval, 2006, s. 107) Duval peker på at vi alltid bruker en representasjon i matematikkundervisning. I blokkprogrammering representerer brikkene kodelinjer i et program. En kodelinje med tekst er igjen en representasjons av endring i strøm og spenning i en krets. Blokkprogrammering kan derfor ses som et valg av representasjon for å tydeliggjøre stegvise kommandoer i et program.




Figur 5. *Blokkbasert og tekstbasert kode* (fra https://makecode.microbit.org/, eget program)

Figur 5 viser samme programkode i to ulike representasjoner. Studier indikerer at nybegynnere finner blokkprogrammering enklere. Dette forklares med at blokkene har forståelig tekst, og at de har former som indikerer sammenkobling (Weintrop & Wilensky, 2015, s. 72).

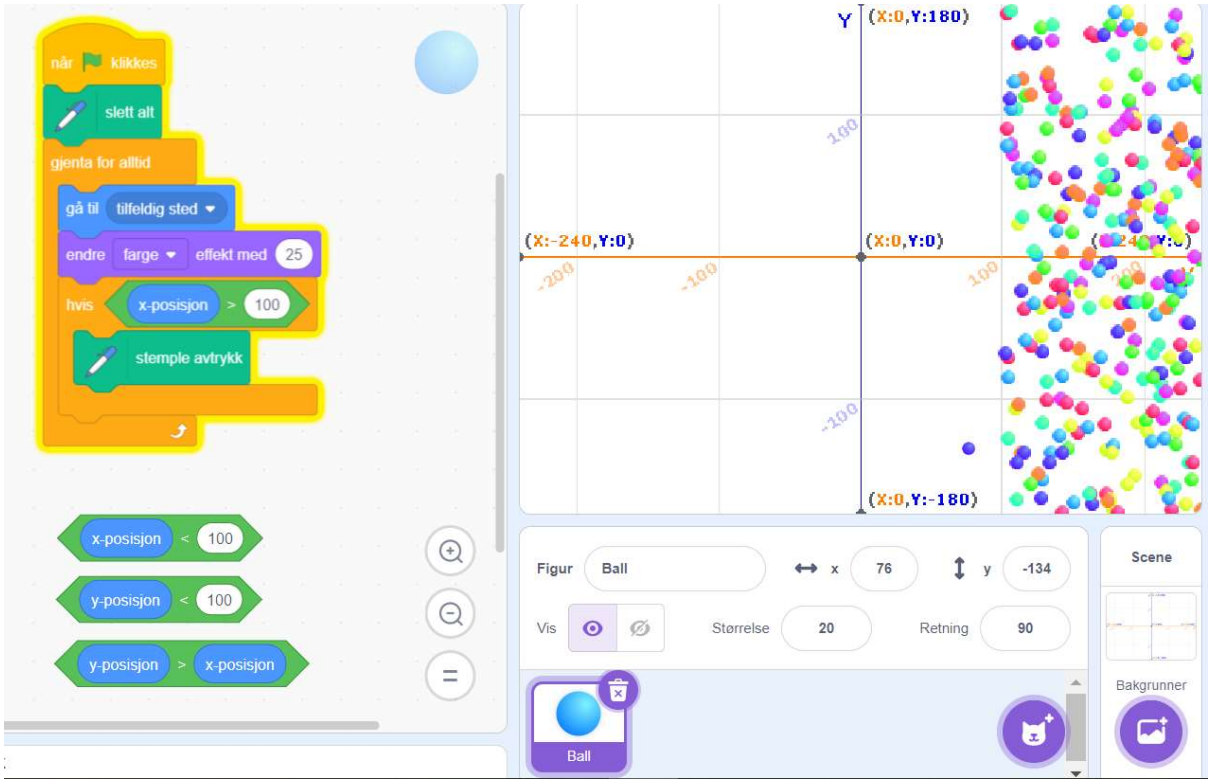



Figur 6. *Kodebrikkers utforming i Scratch*

Kodeblokken til venstre viser hvordan brikker som skal gjentas skal plasseres i en lomme. Til høyre vises hvordan en HVIS-test med et vilkår har en sekskantet form, og parametere og variable en rund form. Det er ikke mulig å plassere verdier og variabler inn i en test, og ikke mulig å sette inn vilkår som en parameter i blokkprogrammering. Alle brikkene har et hakk i toppen og en utstikker nederst som signaliserer at de kan kobles sammen. Brikkenes plassering under hverandre indikerer stegvis progresjonen i programmet.

### *2.2.6 Pusleproblemer*

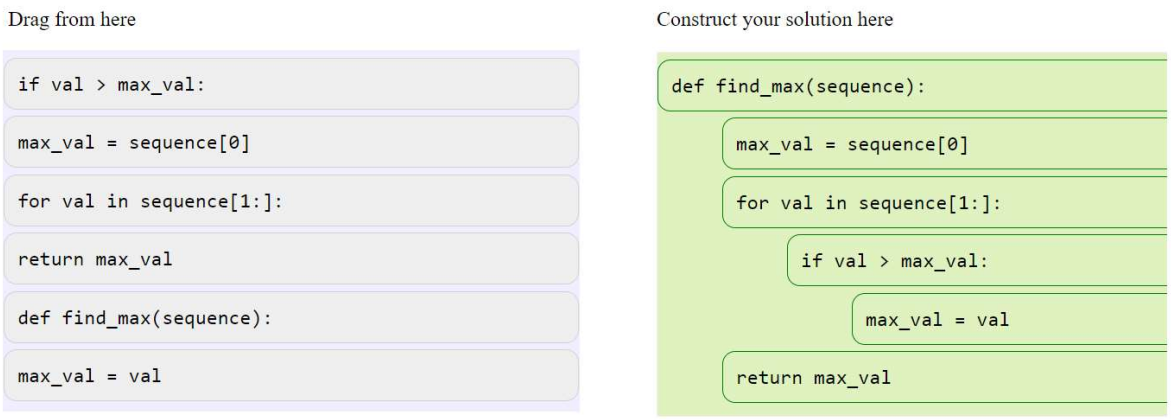
Med pusleproblem ('puzzle') menes her en oppgave som krever manipulering og flytting av objekter for å komme frem til en løsning. Rubiks kube, Brahmas tårn og et puslespill er alle pusleproblemer. Danesi (2020) hevder at pusleproblemer gjennom historien har ledet til oppdagelser blant annet i matematikk, og at mennesker er naturlig tiltrukket lek-orienterte læring fra pusleoppgaver. I programmering har George Gadanidis (2024, s.6-7) flere eksempler der pusling med kodeblokker leder til programmer som løser matematiske problemstillinger.




Figur 7. *Gadanidis' pusleoppgave i Scratch* (fra https://scratch.mit.edu/projects/665007043)

Gadanidis argumenterer for å først gi elever et program som fungerer, deretter et puslespill som vist på figuren. (Gadanidis, 2024, s. 4 – 7). På figur 7 er kodeblokken satt sammen, men de grønne brikkene med vilkår kan plasseres inn i det fungerende programmet.

Et annet eksempel på et pusleproblem er kjent som Parsons' Puzzles (Parsons & Hadlet, 2006). I disse problemene skal man lage fungerende kode ut fra linjer med programkode i en tilfeldig rekkefølge. I et todimensjonalt pusleproblem (Ihantola & Karavirta, 2011, s. 122) må også kodelinjene ha riktig innrykk, slik programmer i programmeringsspråk som Python krever.




Figur 8. *Todimensjonalt parson's puzzle med Python* (fra https://parsons.problemsolving.io/puzzle/cbb39952d55a4be38f935c573a065f9f )

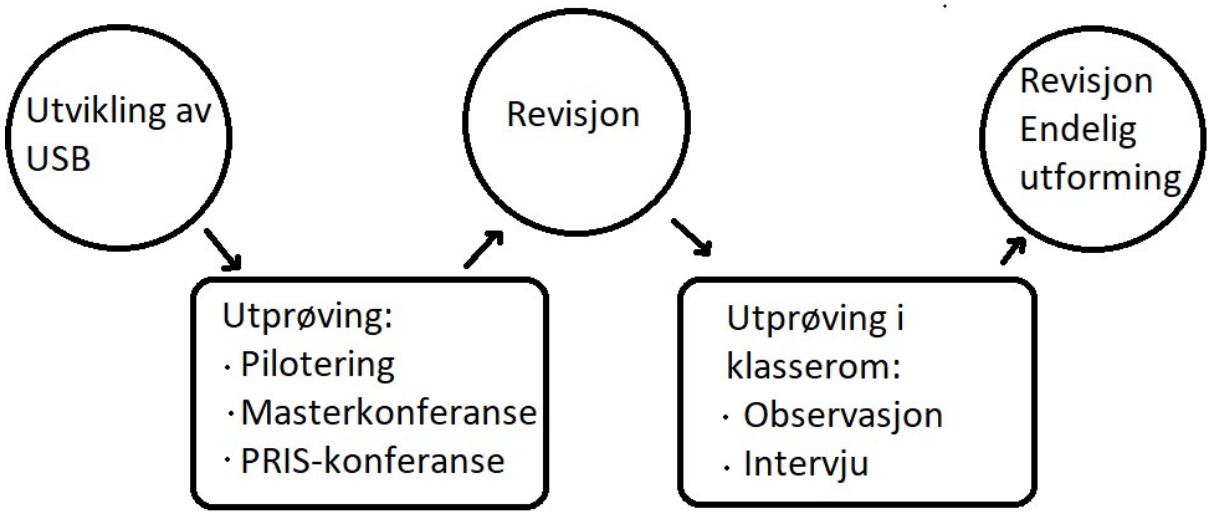
Figur 8 viser hvordan en oppgave kan se ut til venstre. Her skal linjer i et Python-program pusles sammen. Min løsning med korrekt rekkefølge og innrykk av linjene er vist til høyre. Pusleoppgaven kan gjøres mer kompleks ved å legge til ekstra kodelinjer som distraksjoner.

# 3. Metode – systematisk utprøving

I dette kapitlet blir det gjort rede for valg av forskningsdesign, design av produktet, utprøving og revisjoner.

## 3.1 Pedagogisk designforskning

Pedagogisk designforskning er metoden som er valgt i utviklingen av produktet USB. Ifølge van den Akker (2006, s. 5) karakteriseres det metodiske rammeverket av at det gjennomføres en intervensjon i den virkelige verden, at det er en iterativ prosess med en syklisk tilnærming til design, evaluering og revisjon. Metoden er prosessorientert og fokusert mot å forstå og forbedre intervensjonene. Nytteverdien til det som designes er sentral, og intervensjonene skal være forankret i tidligere forskning. Designeksperimenter gjennomgår gjerne tre faser, forberedelse til designeksperimentet, gjennomføring av eksperimentet og retrospektive analyser. (Bjørndal, 2013, s.248). I utviklingen av USB har jeg fulgt denne prosessen. Gjennomføringen gjennomgikk to sykluser med utprøving og revisjon som vist i figur 9.



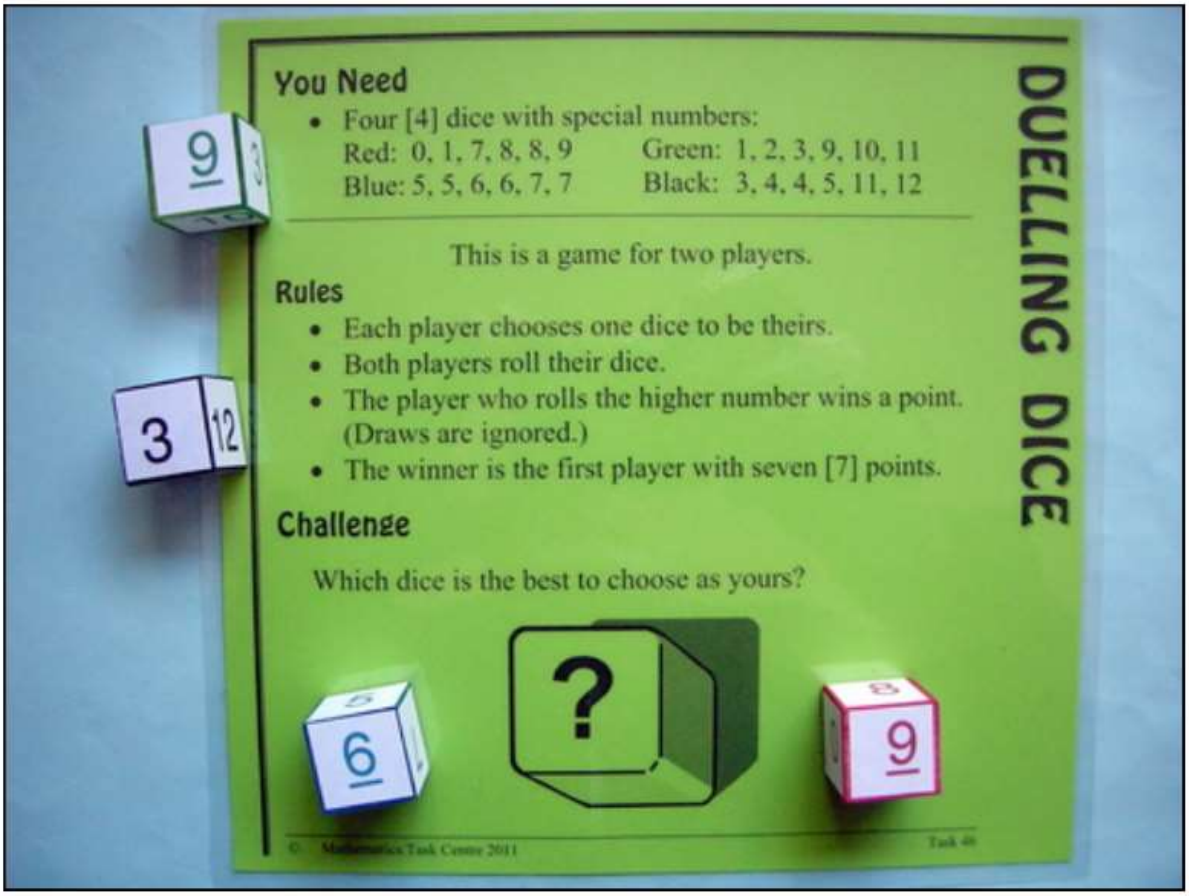

Figur 9. *Fasene i utviklingen av USB*

## 3.2 Design

Formålet med designfasen er å skape et produkt som er nyskapende og som kan danne et grunnlag for videre studier på feltet. (Øgreid, 2021, s. 223). I denne delen vil jeg belyse hvordan det teoretiske grunnlaget har og kriteriene fra problemstilling ledet til utformingen av produktet.

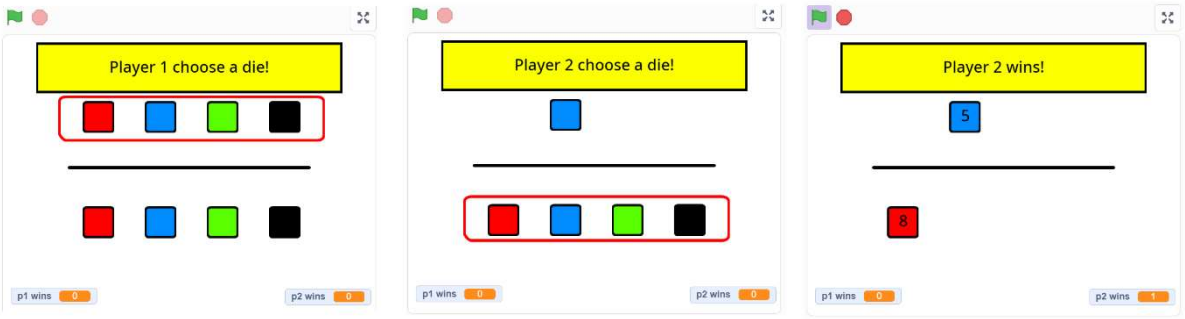
### *3.2.1 Liljedahls praksiser i programmering*

Et prinsipp i undervisningsopplegget er tilpasningen av Liljedahls metodikk til en programmeringsaktivitet. Som nevnt i 2.1.5 har jeg hentet frem 7 praksiser fra Liljedahl som er aktuelle i denne sammenhengen. Praksis 2, 5, 6, 9 og 10 kan benyttes uten spesielle tilpasninger. Det kan nevnes at praksis 2 om gruppedannelse sier at tre elever er best for grunnskoleelever unntatt på svært lave trinn (Liljedahl 2020, s. 44). I litteraturen finnes det mange eksempler på positive effekter av par-programmering (Hannay et al., 2009), og mindre om flere enn to elever på en datamaskin. Praksis 3 som sier at elevene bør arbeide på vertikale flater, ble endret siden få klasserom kan tilby loddrette interaktive tavler til flere elevgrupper. USB sin løsningen er at elevene skal jobbe sammen på en datamaskin, og helst slik at skjermen er synlig for andre grupper. Praksis 1 om oppgavene som gis er den viktigste faktoren som påvirker USB. Tenkende oppgaver i matematikk bør ifølge Liljedahl (2007, s. 248) være engasjerende og utforskende, og nøye tilpasset undervisningssituasjonen. Det første programmet jeg utviklet var en digitalisering av en typisk Liljedahl-oppgave der to spillere velger og kaster terning og ser hvem som får høyest tall.



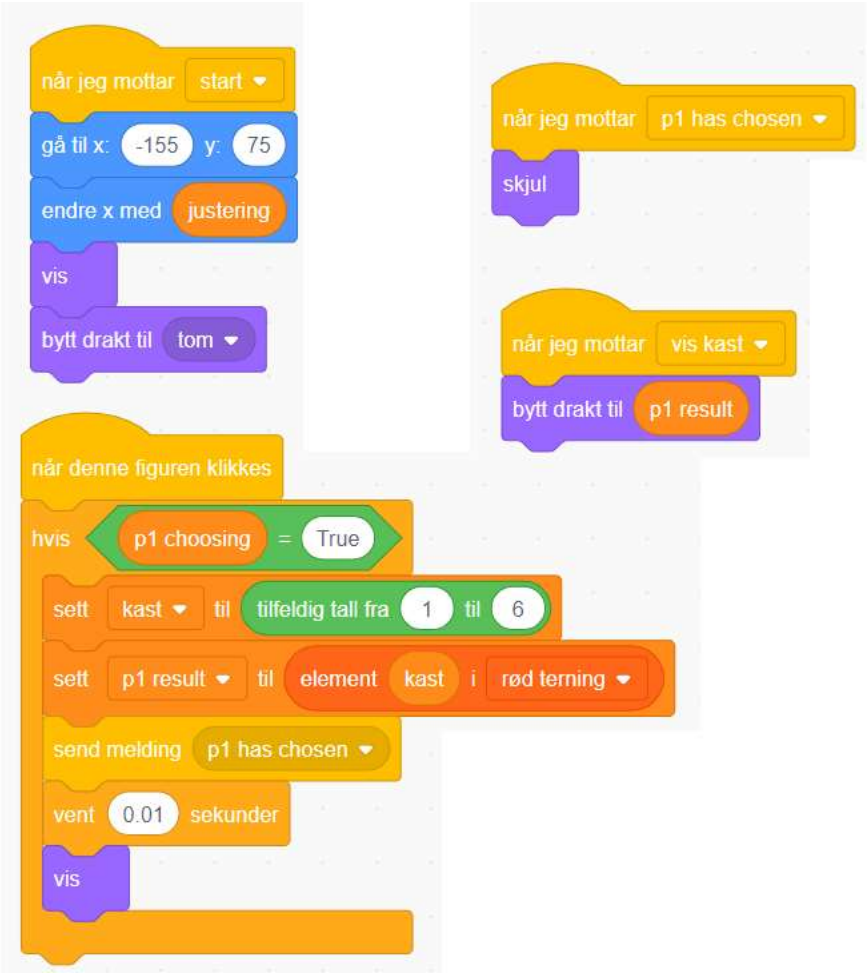

Figur 10. *Dueling Dice* (fra https://peterliljedahl.com/wp-content/uploads/Duelling-Dice.pdf)

Jeg utformet et Scratch-program med navnet «Dueling Dice» som kunne simulere terningspillet, med tanke å la dette være utgangspunktet for et undervisningsopplegg med programmering og sannsynlighet.




Figur 11. *Scratch-programmet Dueling Dice* (fra https://scratch.mit.edu/projects/808064846/)

Dette programmet lot to spillere bytte på å velge terning, rullet terningene og telte opp hvem som vant. Det kan nevnes at terningene var utformet slik at visse farger hadde større sjanse til å slå en annen farge. Etter utviklingen av programmet ble det klart at et undervisningsopplegg bygget på dette ikke ville oppfylle kriteriene for problemstillingen. Programmet kunne brukes til å jobbe med eksperimentell sannsynlighet, og en videreutvikling ville kunne brukes til å se på store talls lov. Men koden ble svært krevende å lese og forstå.

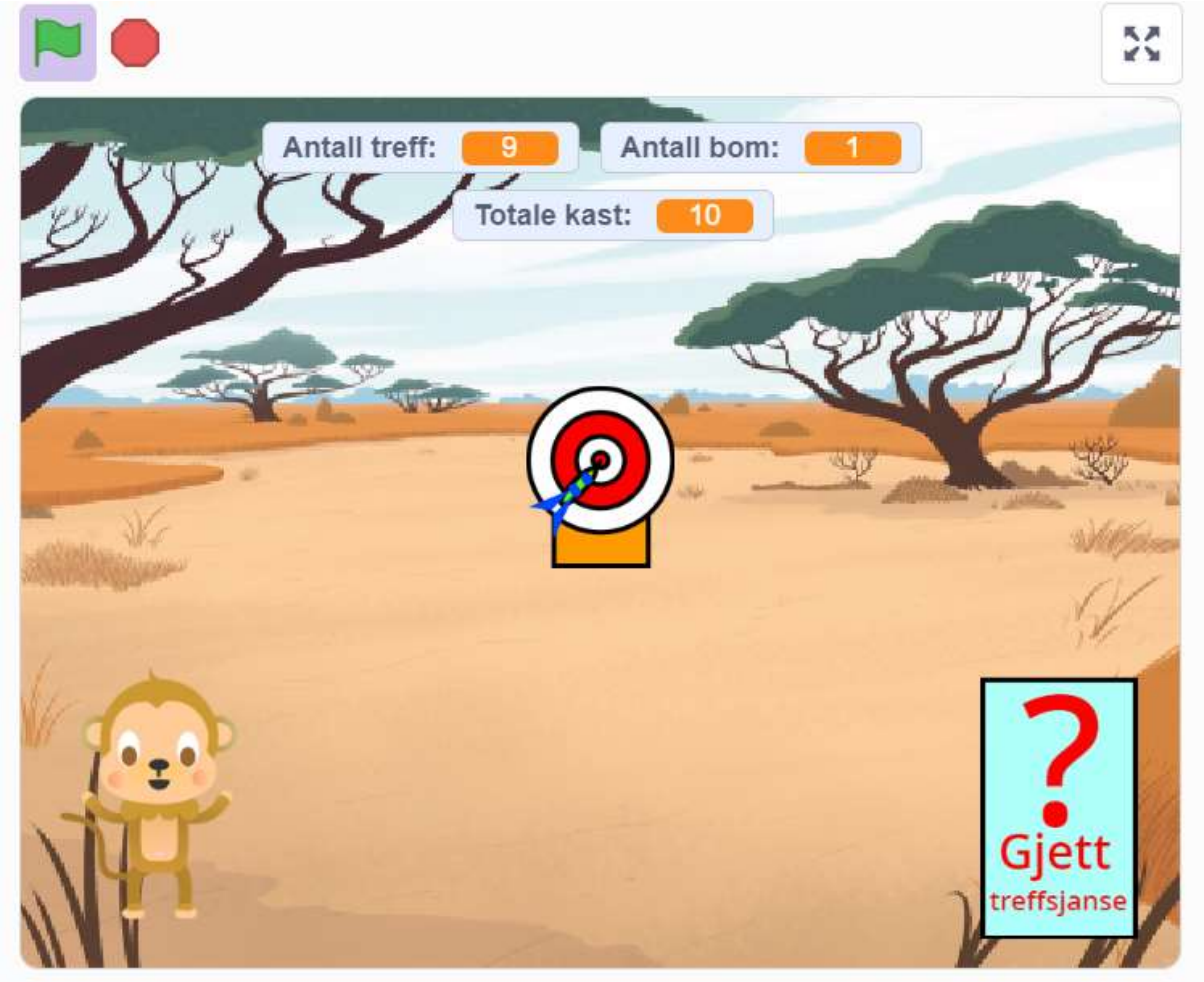



Figur 12. *Eksempel på koden til Dueling Dice-programmet*

For å simulere 4 terninger med ulike verdier på hver side, måtte programmet til slutt bruke listevariable og hadde 8 terning-figurer med ulike drakter og flere simultane hendelser. Vanskelighetsgraden i koden gjorde oppgaven uegnet til en puslespilloppgave, og dermed ville målet om å la elevene programmere falle bort. Programmet benytter en datamaskin til å gjenskape en fysisk aktivitet. I SAMR-modellen (Puentedura, 2006) beskrives ulike måter datamaskiner kan benyttes i aktiviteter. «Duelig Dice»-programmet sin bruk tilsvarer bokstaven S ('substitution'), hvor en aktivitet flyttes over til en digital enhet uten å endre eller forbedre aktiviteten. Det kan derfor argumenteres med for at oppgaven «dueling dice» bør gjennomføres med konkrete terninger, siden programmet ikke oppfyller logos *bruk av programmering for å lære seg matematikk* (2.2.1). I sum medførte disse problemene at jeg forkastet «Duelig Dice» som utgangspunkt for undervisningsopplegget.

### *3.2.2 Første produkt*

I første versjon av opplegget USB valgte jeg å lage to programmer i Scratch. Det første var et ferdig spill, «USB pilkast» der brukerne skulle gjennomføre et antall simulerte pilkast og så gjette hva sannsynligheten for å treffe er. Programmet kombinerer en animasjon hvor elevene ser at en pil blir kastet med et spill det de skal selv gjette sannsynligheten for å treffe. Denne sannsynligheten blir generert på nytt hver gang man spiller. Elevene vil merke at det er nødvendig å gjennomføre mange forsøk for å gjette riktig, og på denne måten bli kjent med prinsippet bak store talls lov.

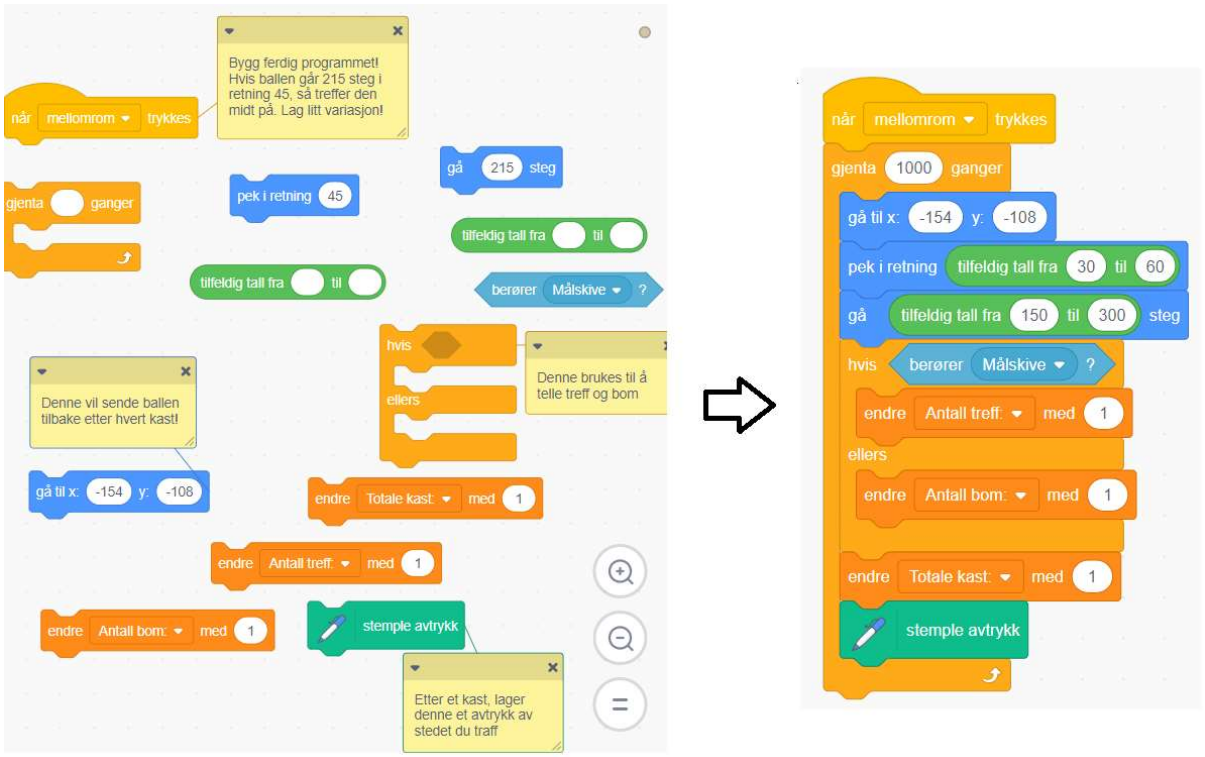



Figur 13. *Spillet «USB Pilkast»*

Etter å ha spilt en gang får man gjennom spillet beskjed om at man kan trykke <pil høyre> for å kaste et stort antall piler uten animasjon. Denne funksjonaliteten ble lagt til for å tillate utforskning av et stort antall repetisjoner og dermed erfare store talls lov. Programmet har en

del kode som vil være vanskelig å lese for de som har lite erfaring med programmering, på grunn av menystyringen i spillet, bruk av logiske variable og at pilkast kalkuleres enten med eller uten animasjon. Hensikten med programmet var derfor ikke at elevene skulle kode, kun bruke spillet som innledning til store talls lov.

Det andre programmet het «USB Bygg selv» og var et program som simulerte ballkast mot en blink. Programmet ar en pusleoppgave, der brukerne fikk plassert ut ferdige kodeblokker som skulle pusles sammen til et program. Det skulle også skrives inn noen parametere i «tilfeldig tall fra_til_»-brikker.




Figur 14. *Kodepuslespill med forslag til fasit*

Selve kodeblokken som skulle bygges hadde kun en løsning, om ikke elevgrupper selv valgte å hente inn og bruke ekstra kodebrikker. Dette gjorde at opplegget mister grader av frihet og blir mer styrt. Valg av parametere i «pek i retning» og «gå ___ steg»-brikkene gjør at det vil bli variasjon sannsynligheten for å treffe i de ferdige programmene. Etter at programmene ble bygget ferdig skulle elevgruppene bruke dem til å simulere ballkast og estimere sannsynligheten for å treffe i sitt eget program. Lærerveiledningen var på to sider, en side med informasjon om opplegget og en side med stegvis fremgangsmåte for gjennomføringen.

## 3.3 Gjennomføring av syklus 1

Opplegget ble testet ut, vurdert, diskutert og revidert i en prosess i tre ulike fora. Tilbakemeldingene ledet så til forbedringer av opplegget.

### *3.3.1 Pilotering*

Pilotering ble gjennomført av meg selv i en gruppe på 15 elever. En økt på 75 minutter ble satt av til gjennomføringen. Alle elevene hadde tidligere erfaring med blokkprogrammering i Scratch. Jeg hadde fem mål med piloteringen. Det første var å vurdere i hvilken grad programmene og det tekniske fungerte etter hensikten. Det andre var å teste og justere skriptet og plan for gjennomføringen. Det tredje målet var å vurdere nivået og tidsbruken på opplegget, og det fjerde målet var å teste og endre observasjonsskjemaet. Det femte målet var å få en oversikt over intervensjoner og instruksjoner som ble gitt under gjennomføringen. Piloteringen ble gjennomført med meg selv som deltagende observatør, og jeg benyttet et observasjonsskjema. (Vedlegg 5) Det tekniske og programmene fungerte stort sett som forventet, men noen programfeil med ble identifisert. Jeg fulgte mitt utkast til skript under gjennomføringen, noterte ned tidsbruk og hvor mange ganger og hvilken type intervensjon jeg ga til gruppene. Det ble tydelig at rekkefølgen på når oppgaven ble gitt og gruppene ble dannet var suboptimal. Det første programmet trengte mindre tid enn 15 minutter, og det andre kunne brukt mer. Det viste seg også at jeg måtte gi detaljerte hint til enkelte av gruppene i puslespilloppgaven, mens andre kunne trengt noe mer å gjøre enn å la programmet kjøre flere repetisjoner.

### *3.3.2 Masterkonferanse*

Masterkonferansen den 12. mars besto av min presentasjon av opplegget for studentgruppen, hvor tre opponenter stilte spørsmål og kom med kommentarer. To av opponentene var masterstudenter, og som potensielle brukere av det didaktiske produktet kunne ha viktige innspill. Det var uklart om alle opponentene hadde rukket å lese veiledningen og prøvd ut programmene i forkant, gitt den korte tidsfristen. Den viktig kommentar som kom inn, var at puslespilloppgaven sannsynligvis ville være svært utfordrende for lærere og elever uten bakgrunn i koding. Dette ble også bekreftet av observasjoner fra piloteringen, hvor noen grupper med god kjennskap til blokkprogrammering endte opp med å få mye hjelp for å få fungerende program.

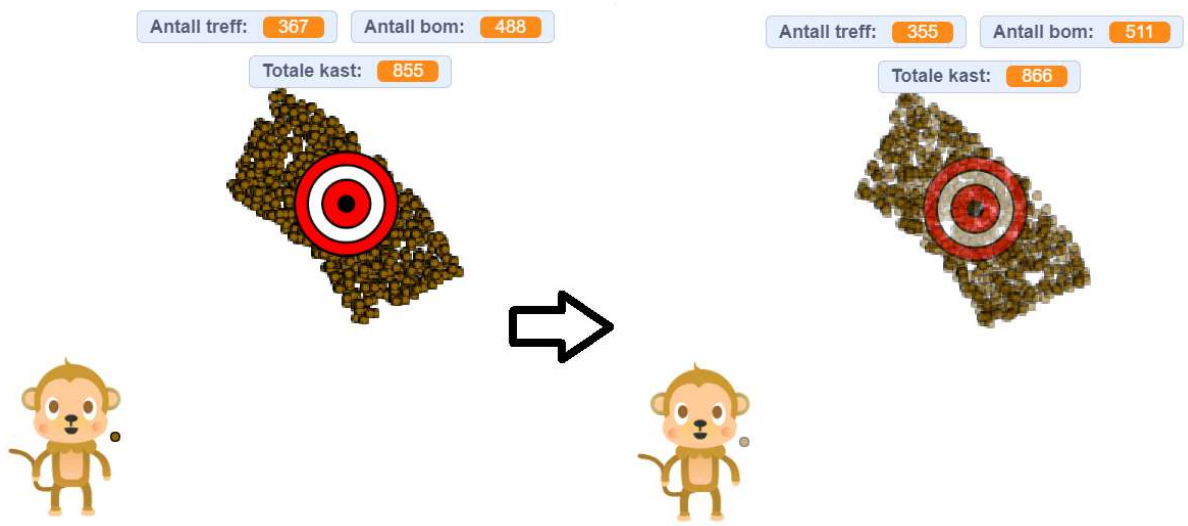
### *3.2.3 PRIS-konferanse*

PRIS, som står for Programmering i Skolen, er et forskningsprosjekt støttet av Tekna, med OsloMet som ansvarlig arrangør. Årets konferanse fant sted 20-21. mars ved Gardermoen. Jeg fikk muligheten til å delta og presentere mitt undervisningsopplegg i en parallellsesjon. Etter en kort presentasjon la jeg ut lenker til programmene og delte ut lærerveiledningen i papirform. Deltagerne fikk prøve ut programmene i en elevrolle, og samtidig gi tilbakemelding på opplegget og programmene, både muntlig og digitalt. Jeg opprettet et nettskjema med synlig QR-kode i forkant av presentasjonen, og oppfordret deltakerne til å komme med tilbakemeldinger underveis. (Vedlegg 8). I tillegg til nettskjemaet ble også muntlige kommentarer under fremlegget og gjennomføringen notert.

Innspill etter konferansen for programmene sin del var få, kanskje grunnet få deltagere på parallellsesjonen. Det ble kommentert at utformingen av puslespillet førte til at opplegget ble styrt, at elevene ville bli sluset mot samme løsning. Det ble kommentert at det var vanskelig å finne puslespillet, fordi det ikke var tydelig hvilken figur koden var på. For undervisningsopplegget sin del så var det klare anbefalinger både muntlig og i nettskjema at bruken av et så lærerstyrt opplegg må begrunnes bedre. I tillegg var det både muntlig og i nettskjema drøfting rundt bruken av Liljedahls metode i opplegget.

### *3.3.4 Endringer etter syklus 1*

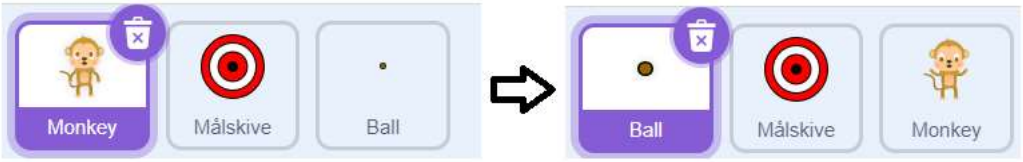
Det ble utformet nye observasjonsskjema (Vedlegg 6) det det ble rom for å notere både hint og utvidelser som ble gitt. Programmene ble kosmetisk forbedret, det ferdige spillet fikk bakgrunn, programfeil ble rettet. En observasjon som ble gjort etter piloteringen var at ballkastene ble dekket over av blinken. Etter en del utprøving ble det bestemt å endre gjennomsiktighet til figurene. Dette muliggjorde å se både blink og avtrykk av ballkast samtidig, og førte til at gjentatte treff på samme sted kunne observeres ved at det ble mørkere jo flere ganger en ball traff.




Figur 15. *Ballkast uten og med gjennomsiktige figurer*

I lærerveiledningen ble rekkefølge på inndeling av grupper og presentasjon av oppgave ble byttet om. Største endring etter pilotering og masterkonferansen var at det ble det utformet et ekstra program, «USB Ballkast». Dette programmet ble i lærerveiledningens plan for gjennomføring puttet inn før programmet «USB Bygg selv», og var i praksis en ferdigbygget versjon av dette. USB Ballkast er Program 2 i det ferdige undervisningsopplegget. Programmet følger Gadanidis (2024, s. 4) sitt prinsipp om å først gi elevene et program som virker, og PRIMM-prinsippet om å la elevene kjøre et program. Det opprinnelige opplegget hadde et program (Program 1) som var klar til å kjøres, men siden koden i dette programmet var både komplisert og fundamentalt ulik koden i puslespillprogrammet, ville ikke PRIMM gjelde her. Program 2 sørger dermed at USB får en lavere terskel for brukerne. Det bør merkes at denne endringen gjør at USB blir mindre åpent, og at gjennomføringen får en ekstra runde med instruksjon og utlevering av kode. For å beholde noe åpenhet og rom for utvidelser i tråd med Liljedahls metode, ble det valgt at et tastetrykk kun gjennomfører 10 simuleringer av ballkast. I lærerveiledningen ble det lagt til en utvidelse om at grupper kan undersøke om det er mulig å øke dette tallet med koding. En alternativ løsning på problemet med for høy vanskegrad i pusleoppgaven som ble vurdert og forkastet, var å delvis eller helt bygge ferdig puslespillet og la kodingen bestå av å velge parametere i programmet. Dette ble vurdert til å være i konflikt med hele problemstillingen for produktet, siden det skulle lages et undervisningsopplegg i programmering.

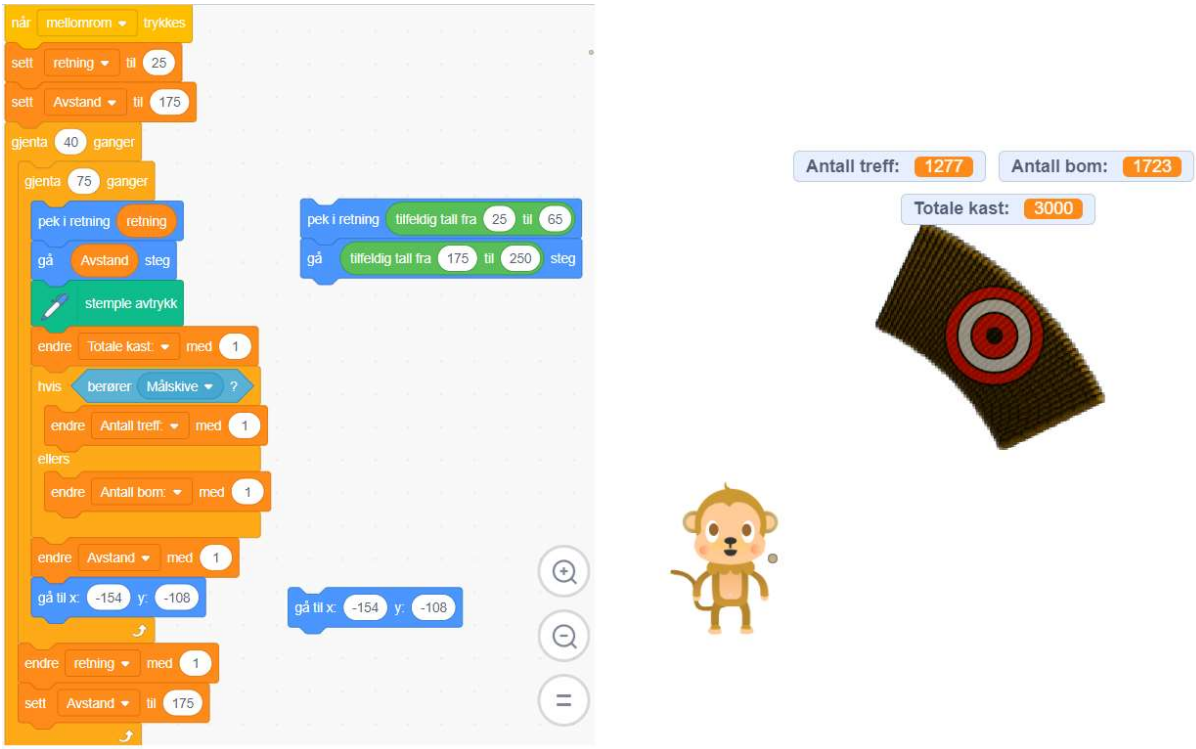
To mindre endringer etter innspill fra PRIS-konferansen var for det første å justere rekkefølgen på figurer i program 3 slik at puslespillet ble synlig med en gang brukere går inn i koden til programmet.




Figur 16. *Rekkefølge på figurer i Scratch*

Dette illustrerer nytten av innspill fra potensielle brukere av opplegget. For min del som visste hvor koden var og hvordan man bytter mellom figurer var ikke dette noe jeg hadde reflektert over å presisere. Lærerveiledningen fikk en setning om at koden befinner seg på figuren «Ball». Denne kjennskapen til Scratch som programmeringsmiljø er et eksempel på logos om programmering som digital kompetanse. Elever som skal programmere selv behøver kompetanse om hvordan programmeringsmiljøet fungerer.

En annen endring kom som et resultat av et innspill om at det var mulig å beregne sannsynligheten i Program 2 eksakt om man kjører programmet gjennom 3000 iterasjoner med systematisk økning av retning og lengde av et ballkast. Man kan dermed beskrive hele utfallsrommet.




Figur 17. *Systematisk beskrivelse av utfallsrommet i «USB Ballkast»*

En løsning på dette ble utrolig nok presentert på neste parallellsesjon på PRIS-konferansen. Om man benytter desimaltall i tilfeldig tall-brikker i Scratch tilfeldige reelle tall i stedet for hele tall, og det er fortsatt umulig å vite helt sikkert hva den ekte sannsynligheten er.


Figur 18. *Desimaltall i tilfeldige blokker*

Innspill til lærerveiledningene etter masterkonferanse og PRIS-konferanse fokuserte på at program 1 var svært komplisert kodet. Lærerveiledningen ble derfor gjort tydeligere på at hensikten med dette programmet ikke var å programmere, men å bli kjent med prinsippet om store talls lov. Det var ellers få innspill på utformingen og innholdet i lærerveiledningene.

## 3.4 Gjennomføring av syklus 2

I denne delen beskrives utprøvingen i klasserom, hvor observasjon av andre lærere sin undervisning, intervju og innkomne resultater bidro til USBs endelige utforming.

### *3.4.1 Etiske vurderinger*

Prosjektet ble godkjent av Sikt i januar 2024 (Vedlegg 3). To lærere sa seg villige til å gjennomføre undervisningsopplegget, og underskrev samtykkeerklæring til å delta i observasjon av praksis og intervju (Vedlegg 4). Klassene som ble observert gjennomgikk opplegget som en vanlig undervisningsøkt. De fikk en gjestelærer eller vikar som gjennomførte opplegget, men deres faste lærer var til stede som observatør. Elevene var heller ikke fokus for observasjonen (3.4.4). Dermed vil ikke gjennomføringen av opplegget hatt negativ virkning på elevgruppene.

### *3.4.2 Datainnsamling*

Hensikten med utprøvingen i klasserom var å forbedre USB. Målgruppen til å bruke USB er lærere som underviser matematikk på ungdomstrinnet. Det ble derfor viktig for utviklingen å observere i hvilken grad produktet fungerte hensiktsmessig for en typisk bruker av produktet. Jeg valgte å gjennomføre en observasjonsstudie med meg selv som delvis deltagende observatør. En delvis deltagende observatør deltar i mye av den sosiale sammenhengen, men ikke selve aktiviteten som gjennomføres. (Dalland et al., 2021, s. 137). Jeg benyttet et revidert observasjonsskjema (Vedlegg 6) med fokus på å observere lærerenes bruk av plan for gjennomføring, intervensjoner og tidsbruk. Dette ga for det meste kvantitative data som tid i minutter og antall intervensjoner.Jeg valgte å utdype observasjonene ved å benytte et kvalitativt intervju i tillegg. Jeg utarbeidet en intervjuguide (Vedlegg 7). Siden hensikten med intervjuet var å få frem lærerens refleksjoner om USB, hadde intervjuguiden åpne formuleringer av typen «hvordan fungerte…» og «har du innspill til…». Noen spørsmål var mer lukkede som «Var instruksjonene […] tydelige?». Intervjuguiden var derfor semistrukturert (Svenkerud, 2021, s.96) og ga rom for å utforske observasjonene fra gjennomføringen.

### *3.4.3 Utprøving i klasserom*

USB ble gjennomført i to parallelle niendeklasser. Begge klassene hadde fått grunnleggende undervisning i beregning av sannsynlighet men ikke store talls lov. I begge klassene var det

varierende kunnskaper om blokkprogrammering, noen har hatt koding som valgfag, andre har hatt minimale erfaringer med dette. Det ble satt av en dobbelttime, 90 minutter, til hver gjennomføring. To matematikklærere gjennomførte opplegget i hver sin klasse. De fikk opplegget to dager i forkant av gjennomføringen. Klassene for gjennomføringen var ikke de vanlige klassene til disse lærerne, de fungerte som vikarer. Dette skyldes praktiske årsaker, siden de ikke selv underviste på 9. trinn.

### *3.4.4 Observasjoner*

Fokus for observasjonen var hvordan lærerne implementerte opplegget, deres bruk av lærerveiledningen. Det var også mulighet å registrere «annet», for eksempel uforutsette utfordringer med programmene. Fokus for observasjonen var på å forbedre USB, derfor ble lærerens handlinger og intervensjoner og ikke elevenes arbeid eller resultater registrert. Under gjennomføringen noterte jeg tidsbruk på de ulike øktene og i hvilken grad lærerne fulgte skriptene og punktene i gjennomføringen. Samtidig observerte jeg lærernes kontakt med enkeltgrupper under arbeidet med programmene. Det ble notert ned antall og type intervensjoner, fordelt på hint og utvidelser som diskutert i 2.1.4. Hint ble registrert hvis læreren ga forslag til løsningsmetode eller brøt en oppgave ned i delmål for å hjelpe elever videre. Utvidelser ble telt om læreren ga ekstra utfordringer til en gruppe. En siste type intervensjon kalte jeg «Klargjøre oppgave». Dette hører ikke til Liljedahls metodikk. Her registrerte jeg om læreren gjentok oppgaveformuleringen muntlig for elevene. Hensikten med denne registreringen var å undersøke om oppgaveinstruksjonene gitt via skriptene i lærerveiledningen var tydelige for elevene. Resultatene etter observasjonen er sammenfattet i tabell 2.

| | Klasse 1 (7 elevgrupper) | | | Klasse 2 (6 elevgrupper) | | |
|---|---|---|---|---|---|---|
| | Økt 1 | Økt 2 | Økt 3 | Økt 1 | Økt 2 | Økt 3 |
| Tidsbruk i minutter | 10 | 10 | 22 | 13 | 8 | 26 |
| Hint | 1 | 3 | 2 | 1 | 2 | 4 |
| Utvidelser | 0 | 1 | 3 | 0 | 0 | 1 |
| Klargjøre oppgave | 3 | 5 | 7 | 3 | 4 | 3 |
| Skript | nei | nei | nei | ja | ja | ja |
| Rekkefølge | ja | ja | ja | ja | ja | ja |

Tabell 2. *Kvantitative data fra observasjonsskjema*

Tidsbruk på øktene gir henholdsvis 42 og 47 minutter elevaktivitet. Resten av tiden gikk til gruppeinndeling, presentasjon av oppgavene, oppsummeringer i fellesskap og å ordne det tekniske med PCer og internett og fordeling av lenker. Når det gjelder hint og utvidelser var det i begge klassene en klart at flere elevgrupper trenger hjelp enn ekstra utfordringer. Dette kan skyldes at lærerveiledningen hadde en eksplisitt fasit og flere forslag til hint enn utvidelser, spesielt til program 1 og 2. Det største problemet med gjennomføringene var hvor ofte læreren måtte klargjøre oppgaven. Dette innebar at læreren muntlig gjentok fra skriptet hva de skulle gjøre, for å hjelpe elevgruppene i gang ville blitt registrert som hint. I klassen hvor skriptet ble fulgt ordrett var det noe mindre behov for klargjøring. I klasse 1 ble ikke skriptene fulgt, i klasse 2 ble de lest opp som skrevet. Jeg observerte i oppsummeringen etter økt 1 i klasse 2 at oppsummeringsspørsmålet som ble stilt til elevene var «Hva var sannsynligheten for å treffe» i stedet for «Hvor mange ganger må vi kaste for å ha en god sjanse til å gjette riktig». Dette ble notert som «annet» og fremkommer ikke i tabellen. En annen observasjon som ikke fremkommer i tabellen var en interaksjon hvor læreren i klasse 2 prøvde å finne ut hvorfor en figur i program 1 ble borte.

### *3.4.5 Intervju*

Intervjuene ble gjennomført umiddelbart etter utprøvingen. Lærer 1 hadde klasse 1, lærer 2 hadde klasse 2. Lærer 1 har 10+ års erfaring som lærer i matematikk, og hadde kjennskap til og erfaring med både Scratch og tenkende klasserom. Læreren brukte ikke skriptene, men tok utgangspunkt i innholdet og formulerte målene med egne ord. Læreren brukte støttedokumentene og fasiten i forberedelse og gjennomføring. Læreren var veldig klar på at hensikten med program 1 fremsto som svært uklar etter å ha lest lærerveiledningen og prøvd programmet. Læreren forsto hvordan programmet fungerte teknisk, men hevdet at det ikke kom frem hva den pedagogiske hensikten var. Læreren ga innspill om å tydeliggjøre hensikten både veiledning og i programmet. Det kom også frem at det var uklart hva som var hensikten med program 3. Lærer 1 indikerte at tidsbruken på hver del var passende. Bruken av tilfeldige grupper og ble løftet frem som positivt. Læreren mente også at bruken av Liljedahls metodikk i pusleoppgaven i økt 3 var lite hensiktsmessig, siden dette var en gjøre-oppgave. Lærer 1 fremholdt at metodikken fungerte greit i de to andre øktene

Lærer 2 er nyutdannet, og hadde kjennskap til men ikke erfaring med bruk av Scratch og tenkende klasserom. Lærer 2 valgte å følge skriptet for igangsetting. Læreren mente det var

tydelig hensikt med programmene. I intervjuet kom det frem at den didaktiske hensikten med program 1 var klar, selv om det kunne vært mer eksplisitt hva oppsummeringsspørsmålene kunne være. Det ble også nevnt at økten knyttet til program 3 var krevende for enkelte grupper, og et innspill var at det kunne vært presisert i veiledningen at program 2 fungerer som en fasit.. Også lærer 2 mente at synlig tilfeldige grupper var positivt, og at det gikk fint med 3 elever i hver gruppe.

### *3.4.6 Endringer etter syklus 2*

Etter gjennomføringen og registreringen av resultater, ble USB utviklet til sin endelige form. I program 1 ble teksten med instruksjoner endret fra «hva er sannsynligheten for å treffe?» til «hvor mange pilkast må vi ha for å gjette rett?». I program 2 og 3 ble det lagt til en tekstmelding om hensikten med programmet som popper opp når programmet kjører.


Figur 19. *Tekstmelding i program 3*

En programfeil med at pilen forsvant etter at piltast høyre blir trykket en gang ble fikset. En større endring var å utvikle og legge til program 4. Hovedgrunnen til dette er ikke direkte knyttet til observasjonene eller intervjuene. Selv om det ikke var fokus for observasjonen, registrerte jeg at elevene så ut til å trives med spill, og det kom spørsmål om det var mulig å kaste pilene selv. Gjennomføringene tok mindre enn 90 minutter i begge klassene, så det var tid tilgjengelig. Spillet er relativt enkelt kodet, og med øvelse blir det lettere og lettere å treffe blinken. I spillet er det ikke mulig å kaste mange piler. Spillet eksemplifiserer dermed en situasjon hvor store talls lov kan ikke benyttes, fordi antall kast er begrenset og sannsynligheten endrer seg underveis i spillet. Elevgruppene vil dermed erfare en annen side ved store talls lov og en grundigere innføring i emnet.

I lærerveiledningene ble program 4 lagt til. Jeg skrev skript, la til lenke, og foreslo utvidelser og forslag til oppsummering, men ellers var det ingen større endringer i strukturen. Hensikten med hvert av programmene ble tydeligere i teoridelen i lærerveiledningen. Skriptet til gjennomføringen av program 1 ble tydeligere på at hensikten med program 1 var å drøfte antall forsøk og ikke sannsynligheten. Det ble lagt til eksplisitte utvidelser, altså forslag til videre arbeid knyttet til hvert program. For å redusere antall elever som ikke forsto hva de skulle gjøre, ble det i lærerveiledningen om gjennomføringen anbefalt å samle elevene etter hver økt, helst stående i en halvsirkel rundt lærer. Dette med å stå er i tråd med Liljedahls metodikk (2020, s. 103) og det tar bort muligheten til å bli distrahert av egen PC-skjerm.

# 4. Refleksjoner

Det å utvikle didaktiske produkter er kjernejobben til enhver lærer. Alle eksempler som ender opp på tavla, utvalget av oppgavene til ukas hjemmelekse og den siste problemoppgaven på mattetentamen er på sin måte didaktiske produkter. I denne oppgaven har jeg presentert et didaktisk produkt. Problemstillingen var å finne ut hvordan et interaktivt undervisningsopplegg med programmering og sannsynlighet kan utformes. Min løsning ble å lage USB. Kriteriene som ble satt opp i forbindelse med problemstillingen er alle synlige i det endelige produktet. USB fokuserer på undervisning i sannsynlighet. Det benytter blokkprogrammeringsspråket Scratch. Scratch er som vist i (2.2.5) et programmeringsspråk som er utviklet med tanke på barn og unge elever som gir gode visuelle tips om bruksmåte og gir oppgavene lav terskel som vist i (2.2.3). Dette er lett tilgjengelig for ulike elevgrupper via en nettleser, så opplegget kan gjennomføres uten spesielt utstyr eller tillatelser. Interaktiviteten er ivaretatt gjennom bruk av elevgrupper og pusleoppgave, og opplegget følger 7 prinsipper i Liljedahls metodikk. Elevene kan klikke på lenken og fikle med programmene uten pålogging. Programmene fungerer i praksis, og ferdige programmer kombinert med en pusleoppgave gjør at elevene jobber seg stegvis fremover ett PRIMM-modellen som beskrevet i (2.2.2)

USB er også et opplegg som behandler store talls lov på en grundig måte. Opplegget legger opp til erfaring, bruk og til slutt drøfting av bruksområde om dette emnet. Bruk av deler av Liljedahls metodikk som beskrevet i (2.2.4) gir økt interaktivitet, siden elevene i grupper vil kommunisere, drøfte og tenke over aktivitetene.

Dette spesifikke undervisningsopplegget kan være verdifull for andre. I første omgang er det et enkelt opplegg som søker å utvikle elevers forståelse om store talls lov. Men det kan videre gi lærere som anvender USB inspirasjon til videre utforskning av programmering. Bruken av ferdige eller halvferdige programmer er for eksempel en effektiv metode for å starte programmeringsopplæring for nye brukere. USB sin bruk av Liljedahls metodikk kan skape interesse for og ønske om å prøve metodikken videre. For brukere med god kompetanse i programmering, kan kodingen av simuleringen være et forslag til videreutvikling av tilsvarende programmer, av både lærere og elever.

USB har noen utfordringer. Produktet kunne gjennomgått flere sykluser og omganger med revisjoner. Alle revisjonene som ble gjort medførte endringer i produktet. Jeg kunne ønsket en syklus hvor utforming av lærerveiledningen var i fokus, selv om praktiske hensyn og tidsbruk ikke gjorde dette mulig. Også programmene ville hatt godt av å bli utsatt for flere runder med elever. Selv under siste utprøving ble det oppdaget programfeil som ble utbedret. Innen rammen på denne masteroppgaven må jeg sette strek et sted og stole på at jeg har gjort nok.

Jeg har også foretatt noen valg i programmene jeg ikke hatt observasjoner til å kunne vurdere i prosessen. For det første så krever program 1, 2 og 3 at elevene selv regner ut eksperimentell sannsynlighet. Programmene kunne gjort dette automatisk, som vist i program 4, men jeg valgte å overlate denne regneoppgaven til elevene. Under prosessen forutså jeg ikke behovet for å trekke frem dette som en problemstilling i intervjuene. Valget ble derfor foretatt etter egen refleksjon. Et argument for å la elevene regne ut sannsynlighet er at med tre elever på hver gruppe så bør de kunne diskutere seg frem til hvordan de finner sannsynligheten. I lærerveiledningen er det også påpekt at det er en fordel at elevene kan regne ut enkle sannsynligheter før de starter opplegget. Et argument mot å la elevene regne ut sannsynligheten er at det kan ta bort fokus fra hovedmålet om å undersøke store talls lov.

Et annet valg var å skjule informasjonen om raske pilkast i program 1, slik at elevene må gjennomføre minst en runde med animasjoner før de kan simulere med tall. Dette var også noe jeg ikke forutså og tok høyde for å undersøke i intervjuene. Valget ble foretatt etter egen refleksjon og etter vurdering av tekniske hensyn med programmet. Det kan nevnes at animasjonene i program 1 ikke er simuleringer, programmet tester to tallverdier først, og deretter lar pilen treffe blink eller fly til siden. Dette ble nødvendig siden brukerne av programmet skal ha mulighet til å fritt veksle frem og tilbake mellom å kaste pil med animasjon og uten animasjon. Siden spillet handler om å gjette sannsynlighet, må det være samme sannsynlighet i animasjonen og sammenlikning av tallverdi. Dette fører i sum til at det er lite overføringsverdi fra koden i program 1 til koden i program 2 og 3.

Det må også nevnes et problem med å gjennomføre mange forsøk for program 2 og 3. I USB simuleres et ballkast ved å bruke bevegelse og registrering av treff/ikke treff av blink, noe som gjør at det tar tid å kjøre mange repetisjoner. På en vanlig PC bruker program 2 omtrent et halvt minutt på å simulere tusen ballkast. Elevenes erfaringer fra program 1 (hvor

repetisjonene ikke tar lang tid) vil være at 5 – 10 tusen repetisjoner er ønskelig. En tradisjonell oppgave med simulering av terningkast og telling som nevnt i (2.2.2) ble gjentatt 1 million ganger på et par sekunder, både med Scratch og Python. Med program 2 og 3 sin nåværende utforming fant jeg ingen måte å redusere denne tiden. Som nevnt under drøfting av praksis 2 i rammeverket, så ble dette et nødvendig resultat av ønsket om å ha grafisk simulering og benytte samme programmeringsmiljø gjennom hele USB. Denne begrensningen i totalt antall forsøk er synliggjort i lærerveiledningen. Dette trenger ikke være en svakhet i opplegget, og jeg vil påpeke noen positive sider ved at det tar tid. For det første vil elevene observere nærmere prosessen, de kan se hvordan antall forsøk øker og tallene endrer seg. Tidsbruken i simuleringen kan føre til at elevene bruker mer tid på å kvalitetssikre koden når de jobber med program 3 før de starter en simulering. De kan observere, reflektere og justere parameterne i programmet sitt. Begrensingen i antall forsøk gjør at elevene får et godt estimat, men vil ikke være nok til at elevene med sikkerhet kan fastslå sannsynligheten i programmet sitt. Dermed vil elevene for alltid undre seg over og aldri få svar på om svaret deres er «rett».

Gjennom dette har jeg svart på problemstillingen til oppgavene. Som nevnt (2.2.5) så finnes det lite tilgjengelig dokumentasjon på slike opplegg. Ved å ha anvendt de grunnleggende kriteriene (se 1.1.1), designet, utprøvd og revidert produktet, vil jeg si at dette nå er et nyskapende produkt for det didaktiske fellesskapet.

# 5. Litteraturliste


Andersen, R., Mørch, A.I., Litherland, K.T. (2021). Learning Domain Knowledge Using Block-Based Programming: Design-Based Collaborative Learning. I D. Fogli, D. Tetteroo, B.R. Barricelli, S. Borsci, P. Markopoulos, G.A. Papadopoulos. (Red.) *End-User Development. IS-EUD 2021. Lecture Notes in Computer Science*, vol 12724. Springer, Cham. https://doi.org/10.1007/978-3-030-79840-6_8

Banchi, H. & Bell, R. (2008). The many levels of inquiry. *Science and Children*, *46*(2), 26–29. http://www.jstor.org/stable/43174976

Bjørndal, K. E. W. (2013). Pedagogisk designforskning: En forskningsstrategi for å fremme bedre undervisning og læring. I M. Brekke & T. Tiller (Red.), *Læreren som forsker: Innføring i forskningsarbeid i skolen* (s. 245–259). Universitetsforlaget.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. & Engelhardt, K. (2016) *Developing computational thinking in compulsory education: implications for policy and practice*. European Commission, Joint Research Centre. https://data.europa.eu/doi/10.2791/792158

Chevallard, Y. (2006). Steps toward a new epistemology in mathematics education. I M. Bosch (Red.) *Proceedings of the fouth Congress of the European Society for Research in Mathematics Education, CERME 4.* Sant Feliu de Guixols, 2005 (s. 21-30). FUNDEMI IQS-Universitat Ramon Llull

Dalland, C. P., Bjørnestad, E. & Andersson-Bakken, E. (2021). I E. Andersson-Bakken & C. P. Dalland (Red.), *Metoder i klasseromsforskning: Forskningsdesign, datainnsamling og analyse* (s. 125–152). Universitetsforlaget.

Danesi, M. (2020). *An anthropology of puzzles: the role of puzzles in the origins and evolution of mind and culture.* Routledge, Taylor & Francis Group

Duval, R. (2006) A Cognitive Analysis of Problems of Comprehension in a Learning of Mathematics. *Educational Studies in Mathematics,* 61, (s.103-131). Doi: 10.1007/s10649-006-0400-z

Espens Klasserom (2022). *Terningkast i scratch del 1*. https://espensklasserom.com/2022/10/16/terningkast-i-scratch-del-1/

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research*, 17(12), 18–32. doi: 10.26803/ijlter.17.12.2

Flø, E.E. & Zambrana I.M. (in-review). Linking instructional design to students' engagement and learning in STEM making activities: the role of psychological distance. *Science Education.*

Gadanidis, G. (2024, 24.mars). *inequalities 2D* [Program]. Scratch. https://scratch.mit.edu/projects/665007043/

Gadanidis, G. (2024, mars). *Mathematics + Computer Programming: Lessons learned from research classrooms.* Innlegg presentert ved PRIS-konferanse, Gardermoen.

Gyllenpalm, J., Wickman, P.O. & Holmgren, S.O. (2010). Secondary science teachers' selective traditions and examples of inquiry-oriented approaches. *Nordic Studies in Science Education*, 6(1). Doi: 10.5617/nordina.269.

Hannay, J. E., Dybå, T., Arisholm, E., Sjøberg, D. I. K. (2009). The effectiveness of pair programming: A meta-analysis. *Information and Software Technology,* 51(7), 1110–1122. https://doi.org/10.1016/j.infsof.2009.02.001.

Hou, X., Ericson, B. J. & Wang, X. (2023). Parsons Problems to Scaffold Code Writing: Impact on Performance and Problem-Solving Efficiency. *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V.2* (s. 665).

Ihantola, P. & Karavirta, V. /2011). Two-dimensional parson's puzzles: The concept, tools, and first observations. *Journal of Information Technology Education. Innovations in Practice,* 10, (s. 119-132) Doi 10.28945/1394

Kilhamn, C. & Rolandsson, L. (2023). Utmaningar och möjligheter. I K. Bråting, C. Kilhamn, L. Rolandsson (Red.), *Programmering i skolmatematiken: möjligheter och utmaningar* (s. 177-196). Studentlitteratur AB.

Kikora (u.å). *Programmere terningkast: 5.6*. (krever pålogging) Hentet 1. mai 2024 fra https://kikora.no/

Kleivdal, K. (Red.). (2020). *Matemagisk 9.* Aschehoug Undervisning.

Koding i skolen (2024). *Sannsynlighet: Oppgave 1*. https://koding.verket.me/programmering-i-matematikk/ungdomstrinn/sannsynlighet/

Konold, C. & Kazak, S. (2008). Reconnecting Data and Chance. *Technology Innovations in Statistics Education,* 2(1). http://escholarship.org/uc/item/38p7c94v

Kunnskapsfilm (u.å) *Opplegg 17: Uniform sannsynlighet*. Hentet 1. mai 2024 fra https://kunnskapsfilm.no/wp-content/uploads/2021/11/Kapittel7_Opplegg_17_uniform_sannsynlighet.pdf

Kunnskapsdepartementet. (2019). *Læreplan i matematikk (MAT01-05).* Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05?lang=nob

Kurland, D. M. & Pea, R. D. (1985). Children's mental models of recursive logo programs. *Journal of Educational Computing Research*, 1, 235–243.

Liljedahl, P. (2020). *Building Thinking Classrooms in Mathematics, Grads K-12*. Corwin.

Liljedahl, P., Chernoff, E. & Zazkis, R. (2007). Interweaving mathematics and pedagogy in task design: a tale of one task. *Journal of Mathematics Teacher Education*, 10, 239–249.

MIT Scratch. (u.å). Scratch—About. https://scratch.mit.edu/about

Nrich (2019). *Low threshold High Ceiling: an Introduction. https://nrich.maths.org/10345*

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books, Inc.

Parsons, D. & Haden, P. (2006, januar). Parson's programming puzzles: a fun and effetcite learning tool for first programming courses." *Proceedings of the 8th Australasian Conference on Computing Education*-Volume 52 (s. 157-163)

Puentedura, R. R. (2006). *Transformation, Technology and Education*. Hentet fra http://hippasus.com/resources/tte/

Selvik, K. (Red.). (2016). *Programmering i skolen*. Utdanningsdirektoratet. https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Semblance S., Waite, J. & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education,* 29(2-3) (s.136-176) doi: 10.1080/08993408.2019.1608781.

Svenkerud, S.W. (2021). Intervjuer i klasseromsforskning. I E. Andersson-Bakken & C. P. Dalland (Red.), *Metoder i klasseromsforskning* (s. 91–103). Universitetsforlaget.

Utdanningsdirektoratet. (2019). *Algoritmisk tenkning*. https://www.udir.no/kvalitet-og-kompetanse/digitalisering/algoritmisk-tenkning/

Van den Akker, J., Gravemeijer, K., McKenney, S. & Nieveen, N. (2006). Introducing educational design research. I J. van den Akker, K. Gravemeijer, S. McKenney & N. Nieven (Red.), *Educational design research* (s. 3–7). Routledge.

Weintrop, D. & Wilensky, U. (2015). To Block or not to Block, That is the Question: Students' Perceptions of Blocks-based Programmering. *Proceedings of the 14th International Conference on Interaction Design and Children,* (s. 69-78). Doi: 10.1145/2771839.2771860

Øgreid, A. K. (2021). Intervensjonsbegrepet i fire kvalitative forskningsdesign. I E. Andersson-Bakken & C. P. Dalland (Red.), *Metoder i klasseromsforskning* (s. 209-237). Universitetsforlaget

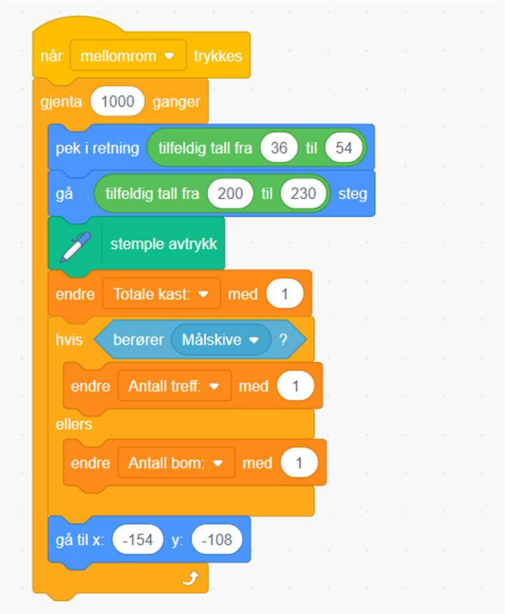
# 6. Vedlegg:

Vedlegg 1: Første utkast lærerveiledning

**Informasjon til lærere om opplegget:**
Opplegget er ment å vare i 60 - 90 minutter. Det kan gjennomføres i to omganger, men det anbefales å gjøre det i ett. Opplegget kan knyttes til prosentregning, finne sannsynlighet og programmering i matematikk, men hovedtemaet er store talls lov innen sannsynlighet. Elevene skal gjøre erfaringer med nytten av å gjennomføre mange parallelle forsøk, og så bruke dette til å si noe om sannsynlighet i et program de lager selv.
Didaktikken i opplegget bygger på "tenkende klasserom" av Peter Liljedahl. (lenke). Han beskriver 14 praksiser for bedre matematikkundervisning i klasserom. I dette opplegget vil jeg fremme følgende ideer:

- Del elevene inn i grupper på 3, og resten i 2. Gjør inndelingen synlig tilfeldig for elevene.
- Elevene mottar muntlig instruksjonen stående i en halvsirkel rundt lærer.
- Under elevenes arbeid, sirkuler mellom gruppene, men etterstreb å ikke besvare spørsmål som vil få elevene til å slutte å tenke selv. "Er dette riktig?" bør ikke besvares med ja eller nei, men "hvorfor tror dere det?", "kan dere sjekke dette på en annen måte?" eller "interessant"
- Under elevenes arbeid, gi hint til grupper setter seg helt fast og ikke kommer videre. Grupper som blir fort "ferdig" kan trenge tips om hvordan de kan utforske videre.

Nærmere informasjon om Liljedahl sine praksiser finnes her: (lenke).
I opplegget er det to programmer. Det første er et ferdig spill som går ut på å gjette hvor stor sannsynlighet det er for at en pil treffer blink. Det er en ny sannsynlighet for hver runde. Det er omtrent umulig å gjette riktig hvis man ikke kaster mer enn 1000 piler. Etter første runde låses det opp ekstra informasjon: Hvis man trykker pil høyre i stedet for mellomrom så kastes hundre piler. Elevene kan bruke dette til å gjennomføre flere forsøk. Elevene kan også endre på antall kast i koden
Det andre programmet handler om å kaste ball, og det er uferdig. Tanken er at elevene setter sammen kodelinjer for å få programmet til å virke. De skal også legge inn tilfeldige variasjoner i programmet slik at ballen treffer eller bommer på blink. Retning 45 og 215 steg vil treffe midt i blinken. En mulig fasit til programmet ser slik ut, men det er vesentlig at elevene selv velger "fra" og "til" i koden for tilfeldige tall.




Elevene vil erfare at det tar tid å kaste ballen (ca 4-5 sekunder for 100 kast) i dette programmet.

**Instruksjon til gjennomføring i klasserom:**

- Elevene deles i grupper på 3 så langt det går, ellers 2. Det er best å ha synlig tilfeldig inndeling, med et digitalt hjelpemiddel eller ved å trekke kort. Hver gruppe på 3 skal ha en PC tilgjengelig.
- Gjør denne lenken tilgjengelig for elevene: https://scratch.mit.edu/projects/944693740/
- Les dette skriptet for elevene:
- "Vi skal jobbe på en litt annen måte i dag. Vi skal se på to programmer i Scratch og finne ut sannsynligheten for at å treffe blink. Er det noen som kan si hvordan vi kan regne ut og beskrive en sannsynlighet? («Gunstige dividert med mulige» eller tilsvarende

I dag skal vi se på situasjoner hvor vi ikke vet sannsynligheten, men skal prøve å finne den ut ved å prøve flere ganger.

I dette første programmet kaster en ape en pil på blink. Vi vet ikke hvilken sannsynlighet det er for å treffe, det blir ny sannsynlighet hver gang vi spiller. Så oppgaven her går ut på å bli enige på gruppa hvor mange ganger må vi kaste pila for å ha en god sjanse for å gjette riktig. Men husk å ikke kaste unødvendig mange piler.

Vi ser på dette i 15 minutter."

- (Veiled grupper som ikke kommer i gang, eller som ikke finner ut pil høyre-snarveien i programmet.)
- Etter omtrent 15 minutter kan du ta en runde, der hver gruppe sier litt om hvor mange forsøk de mener at man kan slippe unna med å klare.
- Gjør neste lenke tilgjengelig: https://scratch.mit.edu/projects/953112745/
- Les dette skriptet for elevene:

"Da vet vi litt om hvor mange forsøk vi trenger. Her er en lenke til del B. Her skal dere bygge et program, og kaste ball. Dere skal bruke tilfeldighet, og velge passende verdier, og kanskje velge størrelse på blinken. Når dere har bygget et program som virker, så vet jo ingen hva sannsynligheten for å treffe egentlig er. Oppgaven deres er å prøve å finne ut så nøyaktig som mulig, hva er sannsynligheten for å treffe med programmet deres. Husk at det ikke vil være et fasitsvar her, siden dere lager deres helt egen versjon av programmet.

Ta et skjermbilde av skriptet (programmet) dere bygget, og skriv hva dere tror sannsynligheten for at ballen treffer er i deres program."

- 10 minutter før full tid kan gruppene igjen beskrive hvordan de bygget programmet, hvor mange forsøk de gjorde og hva de tror sannsynligheten for å treffe med deres program egentlig er.
- Som en avrunding kan det være en diskusjon rundt situasjoner i virkeligheten der det er umulig å vite helt sikkert hva sannsynligheten er, om det er situasjoner hvor man ikke kan gjøre så mange repetisjoner man ønsker. Lykke til med opplegget!

Vedlegg 2: Lærerveiledning USB

# Lærerveiledning

**Velkommen til et Undervisningsopplegg i Sannsynlighet med Blokkprogrammering!**

I dette opplegget vil elevene jobbe utforskende med sannsynlighet og programmering. Opplegget består av denne lærerveiledningen og 4 programmer laget i Scratch. Gjennom programmene jobber elevene med eksperimentell sannsynlighet og store talls lov.

**Om undervisningsopplegget:**

Fag: Matematikk

Trinn: 9. trinn

Aktuelle læringsmål:

- beregne og vurdere sannsynlighet i statistikk og spill
- simulere utfall i tilfeldige forøsk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering

Innhold: Sannsynlighet, Store talls lov, Prosentregning, Programmering (Scratch)

Varighet: ca 90 minutter

**Utstyr:**

- 1 PC tilkoblet internett på hver 3. elev.
- 1 kortstokk eller en annen måte å lage tilfeldige grupper i klasserommet.
- En måte å dele 4 lenker med elevgruppa.
- Anbefalt: En (stor) skjerm til å vise frem programmene
- Anbefalt: Fremgangsmåten på side 3 i denne veiledningen

**Forberedelser:**

- Elevene bør ha en viss kjennskap til hvordan de kan regne ut eksperimentell sannsynlighet
- Ha klar en måte å dele elevene inn i tilfeldige grupper i klasserommet. (Kortstokk)
- Hver gruppe skal bruke 1 PC (f.eks. Den som får «spar» sin PC brukes)
- Ha klar en måte å dele lenkene med elevene, helst 1 om gangen
    - Program 1: https://scratch.mit.edu/projects/944693740/
    - Program 2: https://scratch.mit.edu/projects/979669167/
    - Program 3: https://scratch.mit.edu/projects/953112745/
    - Program 4: https://scratch.mit.edu/projects/999688305/
- Det kan hjelpe å ha en (stor) skjerm å vise frem programmene på.

**Innhold, hensikt og teoretisk grunnlag:**

**Scratch:** I dette opplegget skal elevene prøve ut og programmere ved hjelp av programmeringsspråket Scratch. Elevene arbeider med 4 forskjellige programmer. Nærmere informasjon om programmene kan finnes til slutt i denne veiledningen.

Program 1 (https://scratch.mit.edu/projects/944693740) er et ferdig spill. Elevene ser på en animasjon av en ape som kaster en pil. Etter en del kast skal de prøve å gjette hva sannsynligheten for å treffe *egentlig* var. Denne sannsynligheten endrer seg for hver gang de spiller en runde. Hensikten med dette programmet er at elevene skal erfare store talls lov, siden det er nesten umulig å gjette riktig treffsjanse uten mange forsøk. Det er ikke nødvendig for elevene å programmere her.

Program 2 (https://scratch.mit.edu/projects/979669167) er en ferdig simulering av en ape som kaster ball på en blink. Elevgruppene gjennomfører så mange forsøk de ønsker, for å prøve å estimere sannsynligheten for at en ball treffer blink. Her vil det alltid være samme sannsynlighet for å treffe, med mindre noen grupper endrer på koden.

Program 3: (https://scratch.mit.edu/projects/953112745) er et puslespill. Hensikten er at elevene selv skal programmere ved å sette sammen kodebrikker slik at det blir et fungerende program. De velger selv tallverdier/parametre slik at deres program blir unikt. De skal så jobbe med å undersøke sannsynligheten for å treffe i sitt eget program.

Program 4: (https://scratch.mit.edu/projects/999688305) er et spill hvor elevene selv skal kaste piler på en blink. Dette kan være et utgangspunkt for en diskusjon om når det er mulig å bruke store talls lov og ikke.

**Tenkende klasserom:** Den didaktiske gjennomføringen av opplegget bygger på «Tenkende Klasserom» (Liljedahl, 2021). Denne metodikken består av 14 ulike undervisningspraksiser for å få elevene til å bli tenkende og autonome. I dette opplegget har kombinert metodikken med bruk av PC i programmering. Opplegget har fokus på disse praksisene:

- Oppgavene: Oppgavene gjøres virkelighetsnære og relevante for elevene. Oppgaven gis muntlig mens elevene står i halvsirkel foran lærer. Oppgaven gis på en slik måte at elevene er i gang før det er gått 5 minutter.
- Organisering: Elevene deles i grupper på 3 (2 hvis det ikke går opp). Denne inndelingen bør være tilfeldig, og elevgruppene bør dannes synlig tilfeldig for elevene, f.eks. ved å trekke kort fra en kortstokk. Hver gruppe bruker 1 PC. Gruppene kan plasseres rundt i klasserommet, men helst slik at PC-skjermene er synlige for andre grupper. Elevene bytter på å «styre» PC-en. Elevene *kan* kommunisere med andre grupper, få inspirasjon og ideer fra dem.
- Spørsmål: Lærer bør ikke besvare spørsmål av typen «Er dette riktig» eller «Hva er feil?» med «Ja» eller «Nei». Bedre er «Hvorfor tror dere det?» «Kan det gjøres på en annen måte» eller smile og gå videre.
- Hint og Utvidelser: Grupper som står fast kan trenge hint for å komme videre. Grupper som blir fort ferdige kan få en utvidelse av oppgaven.

Opplegget kan gjennomføres tradisjonelt. Et forslag til oppgaveark til elevene er tilgjengelig til slutt i denne veiledningen.

**Gjennomføring av opplegget (med tenkende klasserom):**

1. Samle elevene i en halvsirkel. Les så dette skriptet (eller bruk egne ord):
   *I dag skal vi undersøke sannsynlighet og programmering. Er det noen som husker hvordan vi kan regne ut sannsynligheten for at noe skal skje?*
   *(*Ta imot svar, løft frem «gunstige delt med mulige»)
   Men *hva er egentlig sannsynligheten for å treffe mål i fotball? Eller treffe i basketball? Er det noen forslag? Er det noe som er mulig å regne ut?*
   (Ta imot svar her også, løft frem «umulig å si»)
   *I dag skal vi se på situasjoner hvor vi ikke vet sannsynligheten, men skal prøve å finne den ut ved å prøve. Nå får en lenke til et program hvor en ape kaster en pil på blink. (*Her kan man vise programmet på storskjerm*) Den kaster, og etter noen kast skal vi prøve å gjette hva sjansen for å treffe egentlig er. Det blir en ny sjanse hver omgang. Og det jeg vil dere skal finne ut er* ***hvor mange ganger vi må kaste pila for å ha en god sjanse for å gjette riktig****. Vi skal nå dele dere inn i tilfeldige grupper. Hver gruppe skal bruke 1 PC. Dere bytter på å styre PC-en, og snakk sammen på gruppa om hva dere gjør og finner ut. Ikke gå videre før alle på gruppa kan forklare hva dere har gjort.*

- Elevene deles i grupper på tre eller to. Gruppene fordeles i klasserommet med 1 PC hver. Gjør denne lenken tilgjengelig for elevene: https://scratch.mit.edu/projects/944693740/
- **Hint** til grupper som står fast:
  - De kan trykke pil høyre for å kaste 50 piler om gangen
  - De kan bruke kalkulator eller runde tall for å regne ut eksperimentell sannsynlighet.
  - Minne dem på at oppgaven er å si noe om antall forsøk for å gjette bra.
  - Be dem prøve flere ganger, med forskjellig antall pilkast .
- **Utvidelse** til grupper som trenger mer utfordringer:
  - Spørre om de kan beregne sannsynlighet når det ikke er runde tall.
  - Prøve å begrense antall forsøk siden piler er dyre.
  - Be dem se på og forklare deler av koden for hverandre.
  - Endre koden under «hvis pil høyre er trykkes» til å kaste 1000 piler.
- Etter ca. 10-15 minutter: La elevene samles i en halvsirkel, gruppevis. Ta en kort runde blant gruppene og be dem dele hva de fant ut. (Løft frem «Vi trenger 1000 eller fler forsøk»). Si så at dette kalles «store talls lov», og at den loven sier at jo flere forsøk vi gjør, jo nærmere kommer vi den ekte sannsynligheten.

2. Les dette skriptet for elevene:
   *Da vet vi litt om hvor mange forsøk vi trenger på grunn av store talls lov. Her er en lenke til neste program, T ape ballkast. Vi skal prøve å finne ut sannsynligheten for å treffe blink i dette programmet. Det er ikke mulig å regne ut. Gjør så mange kast dere trenger, og så tar vi en runde om hva dere tror sannsynligheten for å treffe er. Et hint før dere begynner, det går an å endre et tall i koden så den kaster flere baller*
- Gjør lenke tilgjengelig: https://scratch.mit.edu/projects/979669167
- Etter ca. 5-10 minutter: Ta en kort runde hvor gruppene forteller hva de tror sannsynligheten for å treffe er.

3. Les dette skriptet:

   *Nå skal vi programmere en situasjon. Dere skal selv bygge et program hvor apen kaster ball. Dere må først sette sammen kodebrikkene så programmet funker. Så kan dere bruke tilfeldige tallverdier, forandre på størrelsen til ting og kanskje forandre enda mer. Når dere er ferdige med å bygge programmet deres, så er det ingen som kan vite hva sannsynligheten for å treffe egentlig er. Oppgaven deres er å prøve å finne ut så nøyaktig som mulig, hva er sannsynligheten for å treffe med programmet deres. Husk at det ikke vil være et fasitsvar her, siden dere lager deres helt egen versjon av programmet. Vi jobber i en halvtimes tid med dette.*

- Gjør neste lenke tilgjengelig: https://scratch.mit.edu/projects/953112745/
- **Hint**:
  - Tipse om at koden i program nr. 2 «USB Ballkast» kan brukes for å pusle koden.
  - Elevene kan trenge hint om hvordan de kan pusle programmet.
  - Elevene kan minnes på at de også skal teste sannsynlighetene til programmet.
- **Utvidelser**:
  - Elevene kan endre på utseende til figurene, gjøre ball eller blink større eller mindre. De kan også bruke en annen figur enn ball.
  - Elevene kan legge til mer avanserte bevegelser. Ball kan sprette tilbake ved kanten. Blinken kan bevege seg mens ballen kastes.
  - Elevene kan legge til en egen utregning av sannsynlighet.
- Etter ca. 25 – 35 minutter: Ta en runde der elevene forteller hva de gjorde med sitt program, og hva de tror sannsynligheten for å treffe er. Hvor sikre er de?
- Gjør siste lenke tilgjengelig: https://scratch.mit.edu/projects/999688305

4. Les dette skriptet:

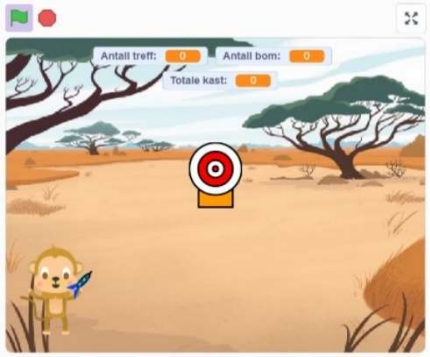
   *Da har vi sett litt på noen simuleringer. Nå skal dere selv prøve å treffe blinken. Dette er et ferdig spill. Det dere skal prøve å finne ut på gruppene er kort og godt om vi kan bruke store talls lov når dere spiller dette spillet.*

- Etter 5-15 minutter: Ta en runde om hva gruppene tror om å bruke store talls lov på spillet. Løft frem ideer som «vanskelig å gjøre mange forsøk» og «vi blir bedre etter hvert». Diskusjonen kan også komme inn på situasjoner fra virkeligheten, som «virker disse medisinene?», «hva blir været i morgen?» og «hva er sjansen for å treffe en papirkurv med et papir?»

5. Etterarbeid: Hvis elevene har scratch-kontoer, kan de remixe programmene og beholde som som egne kopier. Programmene kan også lastes ned og lagres lokalt. Elevene kan eventuelt også ta skjermbilder av den ferdige koden de pusler i punkt 3, selv om det kan være vanskelig å få alt inn i ett bilde. Lykke til med opplegget!

## Informasjon om programmene/fasit

### Program 1 «USB Pilkast»




Trykk mellomrom for å kaste en og en pil. Etter første pil komme «gjett»-figuren frem. Etter du har skrevet inn et svar, får du fasiten. Det kalkuleres en ny treffsjanse hver runde.

Skjult informasjon: Etter første runde gir «info»-knappen mer informasjon. Det står at man kan trykke pil høyre for å kaste mange piler uten animasjon. (Hvis man går inn i koden på apefiguren, under «når pil høyre trykkes» kan dette tallet også økes). Fasit: Antall forsøk bør være 1000 eller mer, gjerne 10 000.

Målet med programmet er å gi øvelse i å beregne sannsynlighet, å vise forskjellen på sannsynlighet og utfall og å vise store talls lov i praksis.

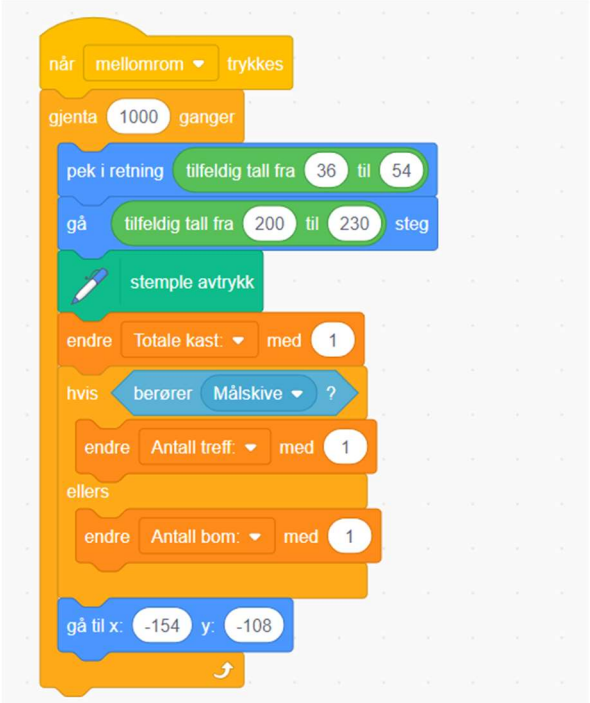
### Program 2. «USB Ballkast»


Programmet er bygget ferdig, men elevene må/bør endre verdien i «gjenta 10 ganger»-blokken. Det tar rundt 3 sekunder å gjennomføre 100 forsøk. Sannsynligheten for å treffe er rundt 41%.

Målet med programmet er å demonstrere hvordan et program som simulerer en faktisk hendelse kan kodes, og gi øvelse i å simulere en hendelse og anslå en sannsynlighet.

### Program 3. «USB Bygg selv»

Programmet må bygges ferdig for å virke. Denne koden er samme som i program 2. I tillegg kan eleene endre størrelse på eller utseende til ballen eller blink. Alle tallverdier med tilfeldige tall bør endres.




Under er et eksempel på hvordan blinken kan flytte seg:

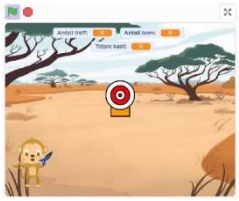

### Program 4: «USB Kaste selv»

Spillet er bygget ferdig. Det har kode for å regne ut sannsynlighet innebygget, som elevene kan kopiere og bruke.

**Tradisjonell versjon: Oppgaveark til elever**

I dette opplegget skal du undersøke hva vi mener med «Store Talls Lov» og undersøke om det er mulig å si noe om sannsynlighet når vi ikke kan regne den ut ved å ta «antall gunstige hendelser dividert med antall mulige hendelser»

1. Følg denne lenken til «USB Pilkast»:
   https://scratch.mit.edu/projects/944693740
   Gjennomfør noen runder med spillet. Gå inn i og endre koden hvis du ønsker. Les ekstra hemmelig info i spillet!
   **Oppgave**: «Hvor mange ganger må vi kaste pilen for å ha en god sjanse til å gjette riktig?»

   Svar her:

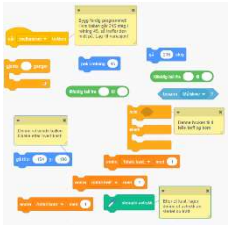
2. Følg denne lenken til «USB Ballkast»:
   https://scratch.mit.edu/projects/979669167
   Gå inn i koden og styr hvor mange baller som skal kastes. Husk at store talls lov sier at «jo flere ganger vi gjennomfører et forsøk, jo nærmere kommer vi den riktige sannsynligheten.»
   **Oppgave**: «Hva tror du sannsynligheten for å treffe blinken er?»

   Svar her:

3. Følg denne lenken til «USB Bygg selv»:
   https://scratch.mit.edu/projects/953112745/
   Dette er et byggesett. Bygg selv et program som kaster en ball, og bestem selv hvilke mulige tilfeldige tall som kan dukke opp.
   Du kan også legge til flere stopp underveis, flytte på blinken, gjøre blinken større eller mindre. Når du er fornøyd med programmet, svar på spørsmålet. Det finnes ingen fasit på denne oppgaven
   **Oppgave**: «Hva tror du sannsynligheten for å treffe blink i DITT program er?»

   Svar her:

4. Følg denne lenken til «USB Kaste selv»:
   https://scratch.mit.edu/projects/999688305
   Kast pilen selv, og se hva sannsynligheten for å treffe blir.
   **Oppgave**: «Kan vi bruke store talls lov på dette spillet?»

   Svar her:

Vedlegg 3: Vurdering fra SIKT



Sikt

# Vurdering av behandling av personopplysninger

| Referansenummer | Vurderingstype | Dato |
|---|---|---|
| 193352 | Standard | 31.01.2024 |

**Tittel**
Undervisningsopplegg i Sannsynlighet med Blokkprogrammering

**Behandlingsansvarlig institusjon**
Høgskolen i Innlandet / Fakultet for lærerutdanning og pedagogikk / Institutt for matematikk, naturfag og kroppsøving

**Prosjektansvarlig**
Jonas Oskarsson

**Student**
Thomas Grønning

**Prosjektperiode**
10.02.2024 - 10.02.2025

**Kategorier personopplysninger**
Alminnelige

**Lovlig grunnlag**
Samtykke (Personvernforordningen art. 6 nr. 1 bokstav a)

Behandlingen av personopplysningene er lovlig så fremt den gjennomføres som oppgitt i meldeskjemaet. Det lovlige grunnlaget gjelder til 10.02.2025.

Meldeskjema

**Kommentar**
OM VURDERINGEN

Sikt har en avtale med institusjonen du forsker eller studerer ved. Denne avtalen innebærer at vi skal gi deg råd slik at behandlingen av personopplysninger i prosjektet ditt er lovlig etter personvernregelverket. Vi har nå vurdert at du har lovlig grunnlag til å behandle personopplysningene.

SAMTYKKE: ALMINNELIGE
Lovlig grunnlag for behandlingen av personopplysninger vil være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 a).

FØLG DIN INSTITUSJONS RETNINGSLINJER
Det er institusjonen du er ansatt/student ved som avgjør hvordan du må lagre og sikre data i ditt prosjekt og hvilke databehandlere du kan bruke. Husk å bruke leverandører som din institusjon har avtale med (f.eks. ved skylagring, nettspørreskjema, videosamtale el.).

Personverntjenester legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til oss ved å oppdatere meldeskjemaet. Se våre nettsider om hvilke endringer du må melde: https://sikt.no/melde-endringar-i-meldeskjema

OPPFØLGING AV PROSJEKTET
Vi vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!

Vedlegg 4: Samtykkeerklæring

# Samtykkeerklæring

## Vil du delta i forskningsprosjektet

## «Undervisningsopplegg i Sannsynlighet med Blokkprogrammering»?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å utvikle et undervisningsopplegg for å undervise en 9. klasse i sannsynlighet og store talls lov med å bruke programmering. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

### Formål

Dette prosjektet er en masteroppgave. I oppgaven vil jeg utvikle et undervisningsopplegg som et didaktisk produkt. Hensikten er å skape et opplegg som kan brukes av lærere i landet for å undervise elever på 9. trinn om sannsynlighet og programmering. Opplegget består av en lærerveiledning og to Scratch-programmer. Det kreves ingen forkunnskaper for å gjennomføre opplegget.

#### Hvem er ansvarlig for forskningsprosjektet?

Høyskolen i Innlandet, HINN, er ansvarlig for prosjektet.

#### Hvorfor får du spørsmål om å delta?

Du er utdannet matematikklærer, eller en lærerstudent med fordypning i matematikk. Du kan dermed gjennomføre opplegget for 9. klasse. Dette spørsmålet går til 2 lærere. Dette er lærere som gjennomfører opplegget.

### Hva innebærer det for deg å delta?

Metoden brukt i denne studien er observasjon av undervisning og intervju i etterkant. Under observasjon registreres elevgruppers arbeid med oppgavene, og kommunikasjon fra lærer til gruppene. Elevgruppene vil anonymiseres fortløpende. Det vil ikke bli gjort videoopptak av undervisningen. Under intervjuet vil det bli stilt spørsmål om hvordan opplegget fungerte. De vil handle om instruksjonene, i hvilken grad de var tydelige og ga et godt utgangspunkt for undervisningen. Det blir også spørsmål om hvordan gjennomføringen av programmeringsoppgaven fungerte, og om opplegget var tilstrekkelig omfattende, gjennomførbart og ga mulighet for

kreativitet. Avslutningsvis blir det stilt spørsmål om innspill til forbedringer og videreutvikling av opplegget.

**Det er frivillig å delta**
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg. Det vil ikke påvirke din stilling som lærer i forhold til skolen.

## Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

- Masterstudent Thomas Grønning og veileder Jonas Oscarsson vil ha tilgang til dine opplysninger.
- Navnet og kontaktopplysningene dine vil jeg erstatte med en kode som lagres separat fra øvrige data.

Du vil ikke kunne gjenkjennes i publikasjon.

## Hva skjer med personopplysningene dine når forskningsprosjektet avsluttes?
Prosjektet vil etter planen avsluttes 10.02.2025. Etter prosjektslutt vil datamaterialet med dine personopplysninger anonymiseres. I oppgaven vil ditt navn bli erstattet med betegnelse «A» eller «B». Anonymiserte opplysninger vil ikke slettes, men kunne gjenbrukes til forskning.

**Hva gir oss rett til å behandle personopplysninger om deg?**
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra HINN har Sikt – Kunnskapssektorens tjenesteleverandør vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

**Dine rettigheter**

Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke opplysninger vi behandler om deg, og å få utlevert en kopi av opplysningene
- å få rettet opplysninger om deg som er feil eller misvisende
- å få slettet personopplysninger om deg
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger

Hvis du har spørsmål til studien, eller ønsker å vite mer om eller benytte deg av dine rettigheter, ta kontakt med:

- *Høyskolen i Innlandet, HINN,* ved masterstudent *Thomas Grønning, 90026755, thomasgronning@gmail.com og veileder Jonas Oscarsson, +4741671628, jonas.oskarsson@inn.no*
- Vårt personvernombud: *personvern@inn.no*

Hvis du har spørsmål knyttet til vurderingen som er gjort av personverntjenestene fra Sikt, kan du ta kontakt via:

- Epost: personverntjenester@sikt.no eller telefon: 73 98 40 40.

Med vennlig hilsen

*Jonas Oscarsson*
(Forsker/veileder)

*Thomas Grønning*
(Masterstudent)

---

# Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet *«undervisningsopplegg i sannsynlighet med blokkprogrammering»*, og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i *intervju*
- ☐ å delta i *observasjon av undervisning*
- ☐ *at mine personopplysninger lagres etter prosjektslutt, til videre forskning*

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet

---

(Signert av prosjektdeltaker, dato)

Vedlegg 5: Observasjonsskjema pilotering

**Observasjonsskjema:**

| | |
|---|---|
| Trenger hint for å komme i gang Del A<br>(Antall grupper) | |
| Trenger hint for å komme i gang Del B<br>(Antall grupper) | |
| Trenger hint for å løse oppgaven Del A<br>(Antall grupper) | |
| Trenger hint for å løse oppgaven Del B<br>(Antall grupper) | |
| Trenger hint for å ferdigstille/presentere oppgaven Del A<br>(Antall grupper) | |
| Trenger hint for å ferdigstille/presentere oppgaven Del B<br>(Antall grupper) | |
| Trenger utvidelser etter B<br>(Antall grupper) | |
| Ferdig løst del A<br>(Tid) | |
| Ferdig løst del B<br>(Tid) | |
| Lærer sine forklaringer gitt til grupper: | |
| Annet: | |

Vedlegg 6: Observasjonsskjema utprøving i klasserom

Observasjonsskjema. Fokus på lærerens handlinger, tid. Avklares i intervju etterpå.

Intro og grupper

| | |
|---|---|
| Tid ved start: | |
| Introduksjon<br>Fulgte skript?<br>Tidsbruk? | |
| Gruppeinndeling<br>Tidsbruk?<br>Annet? | |

Første økt: Pilkast (hva er store tall):

| | |
|---|---|
| Tid ved start av gruppearbeidet: | |
| Intervensjon: Klargjøre oppgave | |
| Intervensjon: Hint | |
| Intervensjon: Utvidelse | |
| Oppsummering felles | |
| Tid ved avslutning av gruppearbeidet: | |

Andre økt: Ballkast 1 (hva er egentlig sannsynligheten)?

| | |
|---|---|
| Tid ved start av gruppearbeidet | |
| Bruk av skript | |
| Intervensjon: Klargjøre oppgave | |
| Intervensjon: Hint | |
| Intervensjon: Utvidelse | |
| Felles runde med svar: | |
| Tid ferdig med gruppearbeidet | |

Tredje økt: Ballkast 2 (bygg eget program):

| | |
|---|---|
| Tid ved start av gruppearbeidet | |
| Bruk av skript | |
| Intervensjon: Klargjøre oppgave | |
| Intervensjon: Hint | |
| Intervensjon: Utvidelse | |
| Felles runde med innspill: | |
| Tid ferdig med gruppearbeidet | |

Annet:

Vedlegg 7: Intervjuguide

# Intervjuguide for Undervisningsopplegg i Sannsynlighet med Blokkprogrammering.

Undervisningsopplegg gjennomført (dato, tid):
Intervju gjennomført (dato, tid):

Skript: «Du har nå gjennomført et undervisningsopplegg i en klasse. I dette intervjuet ønsker jeg å stille spørsmål om dette opplegget. Jeg vil spørre om det skriftlige materialet, om programmene og om gjennomføringen. Jeg vil også spørre om din matematikkfaglige bakgrunn. Jeg noterer selv svarene dine fortløpende. Etter intervjuet vil du få mulighet til å lese gjennom dokumentet. I dokumentet vil ingen identifiserbare personopplysninger fremkomme.»

**Spørsmål om lærerveiledningen til opplegget:**
Var instruksjonene til gruppeinndeling og organisering av klasserom tydelige? Var det noen mangler eller overflødig informasjon?
Fungerte skriptene til å gi klare instrukser til elevene? Var det noen mangler eller overflødig informasjon?
Ga støttedokumentene passe mengde informasjon om mulige elevløsninger? Var det mangler eller overflødig informasjon?

**Spørsmål om programmene:**
Hvordan fungerte program «pilkast» i undervisningssituasjonen? Har du innspill på mulige forbedringer?
Hvordan fungerte program «ballkast» i undervisningssituasjonen? Har du innspill på mulige forbedringer? Var det nok mulighet for kreativitet under elevenes programmering?

**Spørsmål om gjennomføringen:**
Opplegget hadde en tidsramme på 90 minutter (dobbelttime). Har du kommentarer til mengde tid?
Har du kommentarer på vanskelighetsgraden av opplegget? Både for egen del og hvordan det fungerte i klassen?
Totalt sett, kommentarer på gjennomføringen av opplegget?
Hvilke justeringer av opplegget ville du eventuelt gjort før du hadde gjennomført dette med egne klasser?

Vedlegg 8: Nettskjema til PRIS-konferansen.

# PRIS innspill til USB

## Har du innspill eller forbedringsforslag til PROGRAMMENE?

Forslag, innspill og tips til Scratch-programmene

## Har du innspill eller forbedringsforslag til VEILEDNINGEN?

Forslag, innspill og tilbakemeldinger om det didaktiske opplegget