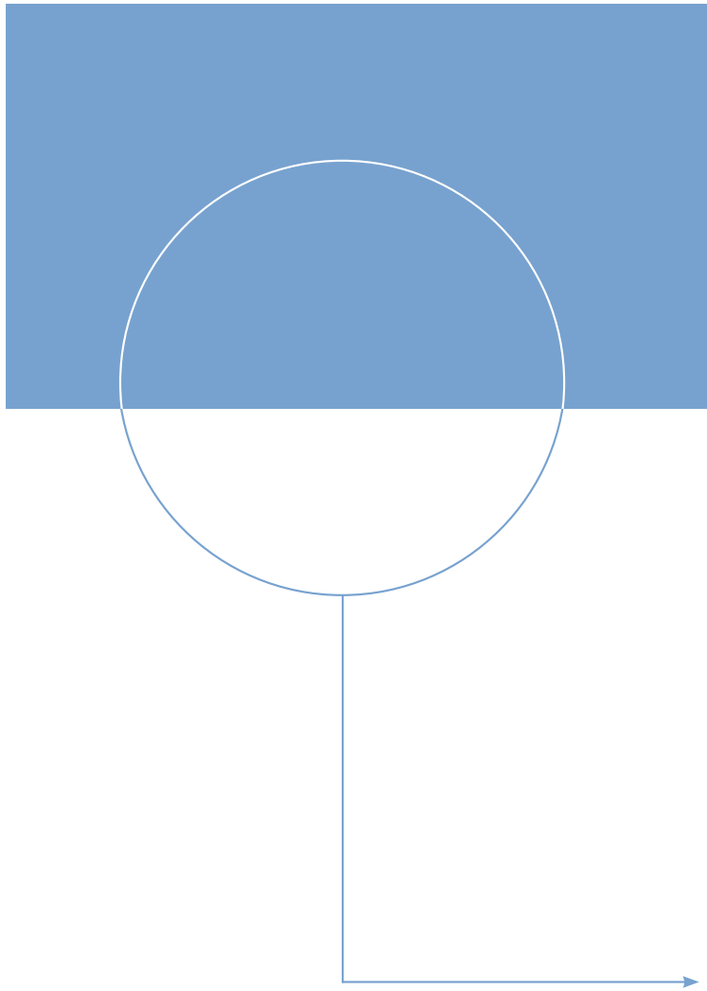


Masteroppgave

Mai, 2020

Ulrik Haugen Wanderås

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunns - og utdanningsvitenskap
Institutt for lærerutdanning

Ulrik Haugen Wanderås

«Okei, vi bare sjekker hvordan det ble»

En kvalitativ studie om bruk av programmering som verktøy i arbeid med geometri

Mai 2020

NTNU
Kunnskap for en bedre verden

NTNU
Kunnskap for en bedre verden

NTNU
Kunnskap for en bedre verden

«Okei, vi bare sjekker hvordan det ble»

En kvalitativ studie om bruk av programmering som verktøy i arbeid med geometri

Ulrik Haugen Wanderås

Masteroppgave i matematikkdidaktikk 5.–10. trinn
Innlevert: Mai 2020
Hovedveileder: Iveta Kohanova

Norges teknisk-naturvitenskapelige universitet
Institutt for lærerutdanning

Sammendrag

Programmering innføres i den norske grunnskolen gjennom Fagfornyelsen høsten 2020. Det argumenteres med at programmering og teknologisk kompetanse er stadig viktigere for å møte den teknologiske utviklingen i samfunnet og arbeidslivet. Siden programmering i matematikkundervisning er nytt i norsk skole vil det være et behov for forskning på fagfeltet. Jeg har i denne kasusstudien undersøkt hvordan et utvalg elever på 6. trinn arbeider med geometrioppgaver i matematikk, da dette er det første temaet hvor kompetansemål i programmering innføres i faget. Det undersøkes både hvordan elevene resonnerer matematisk i løsningen av oppgavene, og hvilke arbeidsmåter de benytter seg av. Utvalget av informanter består av to elevpar som undersøkes ved hjelp av skjermopptak, lydopptak og et semistrukturert intervju.

Funnene fra studien viser at alle elevene resonnerer matematisk i arbeidet med programmeringsoppgavene, men på ulike nivå. Resonnementet som oftest benyttes tar utgangspunkt i figurenes visuelle kjennetegn, mens det mest effektive for å programmere figuren ser ut til å være å ta utgangspunkt i figurenes egenskaper. Scratch bidro til at enkelte av elevene klarte å resonnere abstrakt om sirkelen og figurens egenskaper. Elevenes resonnementer brukes videre for å beskrive deres antatte van Hiele-nivå. Det viser seg at elever på ulike van Hiele-nivå ser ut til å samarbeide godt i arbeidet med programmering på en datamaskin.

Elevenes arbeidsmåter kategoriseres ut ifra rammeverk for algoritmisk tenkning. Funnene fra studien viser at elevene benytter seg av flere arbeidsmåter som går igjen i forskningslitteraturen, i datamaterialet identifiseres abstraksjon, problemnedbrytning, algoritmer og feilsøking. Elevene møter ikke overraskende en rekke utfordringer når de programmerer geometriske figurer. Utfordringene er i stor grad knyttet til å forstå hva som skal programmeres og å lage algoritmer. Dette fører til at programmeringen preges av mye prøving og feiling. Resultatene fra denne studien, og tidligere forskning, tyder på at det tar tid å utvikle effektive arbeidsmåter i programmering.

Abstract

The Norwegian elementary school curriculum will be renewed in 2020, introducing programming. Competence in programming and technology is increasingly important to meet the technological development in both society and work. Programming as a part of mathematics is new to the Norwegian schools, leaving a need for research on the field. In this case study I have studied how a selection of sixth graders work with geometry problems in programming, as this is the first topic in the curriculum where programming is included. This study examines both the students' reasoning in solving the problems and which methods they use to solve the problems. The selection of students consists of two pairs who are examined using screen recordings, audio recordings and a semi-structured interview.

The findings from this study shows that all four students reason mathematically in solving the geometry problems, but at different levels. The most used reasoning is based on the visual characteristics of the figures, while the most effective seems to be based on the properties of the figures. Scratch helped some of the students to reason abstractly about the circle and its properties. Students' reasoning is further used to describe their assumed van Hiele-levels. It turns out that students on different van Hiele-levels seems to collaborate well when working on one computer with programming.

Students' working methods where categorized from the framework for computational thinking. The findings from this study show that the students use several methods that are central in the research literature, this includes abstraction, decomposition, algorithms and debugging. The students encountered several challenges when programming geometric shapes. The challenges are often related to understanding what is to be programmed and to make algorithms, which causes the programming to be characterized by a lot of trial and error. The results of this study and previous research suggest that it takes time to develop effective working methods in programming.

Forord

Masteroppgaven markerer slutten på fem morsomme og lærerike år som lærerstudent i Trondheim. Prosessen med å skrive en masteroppgave har inneholdt både oppturer og nedturer, men jeg kan nå se tilbake på en erfaring jeg ikke ville vært foruten. Mitt ønske var å skrive en masteroppgave som er relevant for meg som fremtidig lærer, og når programmering innføres i nye lærerplaner fra høsten jeg starter i yrket ble det et naturlig valg.

Takk til veilederen min Iveta Kohanova for grundige og konstruktive tilbakemeldinger gjennom arbeidet med masteroppgaven min. Jeg ønsker også å takke læreren som ga meg muligheten til å gjennomføre undersøkelsen i klassen sin, og til alle de positive elevene som deltok.

Til slutt vil jeg takke mine nærmeste, som alltid støtter meg og kommer med gode råd.

Ulrik Haugen Wanderås,

Trondheim 15.mai 2020.

Innholdsfortegnelse

Figurer

Tabeller

1 Innledning

Utviklingen av samfunnet endrer hvilke kompetanser som blir viktige i fremtiden. Teknologi har blitt en mer sentral del av hverdagen, og det har gjennom de siste årene vokst fram en internasjonal bevegelse for å fremme programmering i skolen (Sevik, 2016). Programmering beskrives som en viktig ferdighet for å møte fremtiden. Skolen har et bredt samfunnsmandat, og formålsparagrafen i opplæringsloven slår fast følgende:

> *«Opplæringa i skole og lærebedrift skal, i samarbeid med og forståing med heimen, opne dører mot verda og framtida og gi elevane og lærlingane historisk og kulturell innsikt og forankring» (Opplæringslova, 1998, s. § 1-1).*

Dersom skolen skal åpne dører mot verden og fremtiden er det viktig å fokusere på ferdigheter som er sentrale for fremtiden. Argumenter for programmering i skolen knyttes gjerne til nødvendige ferdigheter for det 21. århundre, fremtidige behov for kompetanse i næringslivet og evne til å forstå hvordan et stadig mer digitalisert samfunn fungerer (Sevik, 2016). Den teknologiske utviklingen går stadig raskere, og teknologiens plass i samfunnet må gjenspeiles i skolen. Dersom målet er at de neste generasjonene skal være best mulig forberedt, vil skolen være en naturlig arena for å utvikle ferdigheter som er nødvendige.

Høsten 2016 leverte en ekspertgruppe utnevnt av Utdanningsdirektoratet rapporten «Teknologi og programmering for alle», der det anbefales at det opprettes et nytt fag innen teknologi og programmering for å møte fremtidens behov og kompetanse på en best mulig måte (Sanne et al., 2016).

1.1 Programmering i skolen

Programmering kan defineres som prosessen knyttet til utvikling og implementering av instruksjoner for dataprogrammer, slik at datamaskinen kan utføre spesifikke oppgaver, løse problemer og støtte menneskelige interaksjoner (Grover & Pea, 2013). Veldig forenklet kan det forklares som å bryte ned et gitt problem i et sett av kommandoer, og så få en datamaskin til å utføre kommandoene. Datamaskinen utfører instruksene du gir den en etter en, akkurat slik de står skrevet, og i nøyaktig den rekkefølgen de er skrevet. Prosessene innen programmering knyttes ofte til matematisk tenkning og algoritmisk tenkning (Grover & Pea, 2013). Det blir derfor ansett som en viktig ferdighet innenfor fremtidens kompetanse i matematikk. Algoritmisk tenking kan forklares som en viktig prosess med å utvikle strategier og fremgangsmåter for å løse et problem, og å vurdere om det er best egnet å løse med eller uten digitale verktøy (Utdanningsdirektoratet, 2019).

Programmering i matematikkundervisning er ikke en ny tanke. Seymour Papert kan sies å ha vært forut for sin tid da han utviklet programmeringsspråket Logo, for å introdusere programmering i skolen på en barnevennlig måte (Papert, 1980). Programmering ble på 80-tallet forsøkt integrert blant annet i den amerikanske skolen, uten at det kan anses som en stor suksess. I nyere tid har imidlertid dette programmeringsspråket ligget til grunn for utviklingen av en rekke programmer for både barn og voksne. Hovedfokuset til slike programmer er at det skal være enkelt å komme i gang med programmeringen (lavt gulv), nesten ingen begrensninger for hvor avansert man kan kode (høyt tak), og ingen begrensinger for hvor komplekse prosjekter man kan arbeide med (vide vegger) (Sevik, 2016).

Det har tidligere blitt diskutert om programmering skal innføres som et nytt fag, eller om man skal integrere det i de eksisterende fagene som matematikk og naturfag. I flere av landene som Norge gjerne sammenligner seg med i undervisningssammenheng er programmering innført i fag som matematikk og naturfag. Blant landene som har innført programmering som en del av læreplanene, er det store forskjeller på hvordan dette er gjort (Sanne et al., 2016). Noen land har integrert programmering i ulike eksisterende fag, mens andre land i undersøkelsen har lagt det inn i læreplanene som et eget teknologifag. Eksempelvis innførte engelsk skole faget Computing i 2014, mens Frankrike og Finland valgte å integrere programmering i matematikkfaget (Sanne et al., 2016).

På kort sikt er det mest realistisk å jobbe med programmering innenfor eksisterende fag og læreplaner, men på litt lengre sikt kan programmering knyttes til egne kompetansemål i fellesfagene (Sevik, 2016). Dette vil sikre både en dypere forståelse og en bredere bruk. Det er bestemt at programmering i første omgang innføres i eksisterende fag i den norske skolen fra 2020. Innføringen av programmering i skolen er et signal om at programmering er en ferdighet alle norske elever skal besitte.

1.2 Problemområdet og forskningsspørsmål

Allerede etter 6. trinn skal elevene ifølge kompetansemål i fagfornyelsen kunne «bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønster» (Utdanningsdirektoratet, 2019). Dersom man ser på tidligere forskning på programmering i matematikkundervisning, ser det ikke ut til å være tilfeldig at geometri blir valgt som en inngangsport. Programmeringsspråk som Logo ble utviklet blant annet for bruk i geometri og bevegelse av en figur i planet. Flere anerkjente forskere fokuserer på geometri i sin forskning på programmering i matematikkundervisning, for eksempel Papert (1980), Resnick et al. (2009) og Clements og Battista (1992). En litteraturstudie av 15 sentrale artikler innenfor programmering i matematikkundervisning, viser også til at geometri er en del av pensum som har en naturlig kobling med programmeringsaktiviteter (Forsström & Kaufmann, 2018). For å undervise i programmering allerede på mellomtrinnet er det viktig å velge programmer som passer for elever i denne alderen. Ulik forskning innen fagfeltet viser at det blokkbaserte programmeringsspråket Scratch egner seg til bruk i matematikkundervisning på skolen, også for yngre elever (Calder, 2010; Fassakis, Gouli & Mavroudi, 2013; Barcelos, Muñoz-Soto, Villarroel, Merino & Silveira, 2018). I denne studien har jeg valgt å benytte Scratch for å undersøke elevenes programmering.

Resonnering og argumentasjon er et av kjerneelementene i fagfornyelsen, her beskrives det at «elevene skal forstå at matematiske regler og resultater ikke er tilfeldige, men har klare begrunnelser. Elevene skal utforme egne resonnementer både for å forstå og for å løse problemer» (Utdanningsdirektoratet, 2019). Flere andre kjerneelementer beskriver også at elevene skal lete etter mønstre og finne sammenhenger. Videre begrunne svarene sine, oppdage sammenhenger, og utforske figurer for å finne sammenhenger (Utdanningsdirektoratet, 2019). Med utgangspunkt i kjerneelementene i matematikk ser det ut som at resonnering er viktig for norske elever. Ulik forskning trekker frem at Scratch er effektivt program for å arbeide med resonnering og problemløsning i matematikk (Brown et al., 2013; Calao, Moreno-León, Correa & Robles, 2015).

Med bakgrunn i at programmering innføres i nye kompetansemål i matematikk allerede på mellomtrinnet, er det interessant å undersøke om elever på 6. trinn resonnerer matematisk

når de programmerer. For å undersøke hvordan elevene gjennomfører selve programmeringen har jeg tatt utgangspunkt i et rammeverk for algoritmisk tenkning, som brukes for å studere hvilke arbeidsmåter elevene benytter. Problemstillingen blir derfor som følger:

«Hva kjennetegner et utvalg 6. klassingers matematiske resonnering, og deres arbeidsmåter i programmering av geometriske figurer?»

Denne problemstillingen skal jeg undersøke ved hjelp av to forskningsspørsmål, der det første undersøker resonnementer i geometri og det andre handler om elevenes arbeidsmåter under programmeringsarbeidet.

1. *Hvilken type matematiske resonnementer bruker elevene i møte med programmeringsoppgaver i geometri?*
2. *Hvilke arbeidsmåter innenfor algoritmisk tenkning benytter elevene i løsningen av programmeringsoppgaver i matematikk?*

For å undersøke forskningsspørsmålene har jeg valgt å ta lyd- og skjermopptak av to elevpar når de arbeider med programmeringsoppgaver, før det ble gjennomført et semistrukturert gruppeintervju med hvert av elevparene.

Det er en mangel på forskning gjennomført med norske elever innenfor dette fagfeltet, og vi vet derfor relativt lite om det matematiske potensialet i slike programmeringsaktiviteter. Etter at fagfornyelsen ble presentert er det klart at programmering inkluderes i matematikk fra høsten 2020, og det vil derfor være interessant å se på hvordan elever på 6. trinn arbeider med programmering for å løse matematikkoppgaver i geometri.

1.3 Oppbygningen av studien

Masteroppgaven består av seks kapittel, som igjen består av flere delkapittel. I kapittel to presenteres forskningsprosjektets teoretiske perspektiv. Dette for å gi leseren kunnskap om det teoretiske rammeverket prosjektet tar utgangspunkt i. Kapittel tre er metodekapittelet, her beskrives og begrunnes valg av metode for datainnsamling, samt gjennomføring og analyse av datamaterialet. Videre i dette kapitlet drøftes studiens kvalitet, og det blir redegjort for etiske betraktninger. Kapittel fire er analyse av datamaterialet, her presenteres først elevenes resonnering, og deretter deres arbeidsmåter. Elevparene presenteres adskilt i analysen. I kapittel fem diskuteres funn fra analysen. Her presenteres interessante funn, som videre diskuteres ut fra det teoretiske rammeverket, for å svare på de to forskningsspørsmålene. Til slutt er kapittel seks en avslutning på denne masterstudien, og inneholder en konklusjon, vurdering av studiens kvalitet, og forslag til videre forskning innenfor fagfeltet.

2 Teoretisk perspektiv

I denne studien undersøker jeg hvordan elever resonnerer, og hvilke arbeidsmåter de benytter seg av, i møtet med programmering av geometriske figurer. Sentrale begreper i studien er forståelse av geometri, resonnering, programmering og algoritmisk tenkning. Den første delen av kapitlet forklarer hva som ligger til grunn for forståelse av geometri, der utgangspunktet er van Hieles nivåer. Deretter beskriver jeg hva et matematisk resonnement er, og ulike begreper som er viktige for oppgaven. For å kunne analysere elevers arbeidsmåter innenfor programmering presenteres rammeverket algoritmisk tenkning. Videre presenteres et overblikk av programmering i matematikkundervisning, med ulik forskning på programmering i skolen, og programmet Scratch. Denne oppgaven bygger på relasjoner mellom programmering og matematikk, og de ulike teoriene som presenteres i dette kapitlet vil senere være utgangspunkt for analyse og drøfting av datamaterialet.

2.1 Forståelse av geometri

Forståelse i matematikk beskrives av van Hiele (1959/1984) som å kjenne sammenhenger mellom teoremer. Først når man forstår betydningen av teoremene forstår man også sammenhengen mellom dem. Geometri er den delen av matematikken som handler om egenskapene til romlige legemer, og figurers form og størrelse. Å ha forståelse i geometri baserer seg på en mer abstrakt tanke av figurene, der resonnementer basert på egenskapene er på et høyere nivå enn den visuelle fremstillingen av en figur (Lehrer, 1998). Den abstrakte forståelsen av figurene er også noe som vektlegges i kompetansemål i skolen. Etter 6.trinn skal elevene blant annet kunne beskrive egenskaper og definisjoner av figurer, forklare hva som er felles, og hva som skiller ulike figurer fra hverandre (Utdanningsdirektoratet, 2019).

Et eksempel på den abstrakte delen av geometri er når man ser for seg en sirkel: Man kan se for seg en tegnet sirkel inkludert fargen på blekket og mer, og ikke den ideelle, perfekte sirkelen. Når man skal resonnere matematisk bruker man den matematiske sirkelen. Den matematiske sirkelen har ingen farge, er ikke noe materie eller noe masse, den er perfekt (Fischbein, 1993). Når man resonnerer om geometriske figurer forklarer Lehrer (1998) at elever kan ta utgangspunkt i det visuelle ved figuren, eller viktige egenskaper, der et resonnement som bruker egenskapene er på et høyere van Hiele-nivå.

Battista (2007) beskriver geometri som et komplekst nettverk av begreper, måter å resonnere på, og representasjonssystemer som brukes for å utforske og analysere figurer og rom. Geometrisk resonnering består i hovedsak av å oppdage og bruke konseptuelle systemer for å undersøke figurer (Battista, 2007). Eksempler på slike systemer kan være å benytte seg av egenskaper for å undersøke ulike trekanter og firkanter. Slike systemer bruker begreper som måling av vinkler og lengde, kongruens og parallellisme. Dette for å kunne konseptualisere spatiale forhold mellom figurene. Det kan for eksempel være å definere et kvadrat som en figur med fire like lange sider med fire rette vinkler. Da har man et begrep som baserer seg på egenskaper for denne figuren, og som kan hjelpe flere å resonnere mer presist om akkurat denne klassen av figurer (Battista, 2007).

Underliggende for det meste av geometri er spatial resonnering, som er evnen til å kunne se, undersøke og reflektere rundt spatiale objekter, bilder, forhold og transformasjoner. Dette inkluderer å lage figurer, undersøke dem og svare på spørsmål rundt dem (Battista, 2007).

2.1.1 van Hiele-nivåene

van Hiele (1959/1984) beskriver at elevers geometriske forståelse utvikler seg i en bestemt rekkefølge, uavhengig av alder. Elever lærer ofte et ferdiglaget system av relasjoner, istedenfor kunnskapen om å lage slike systemer på egenhånd. Siden læreren allerede kan teoremene og deres relasjoner, blir lærerens forklaringer uforståelig for elevene. Lærer og elev snakker forskjellige språk, eller de tenker på forskjellige nivåer (van Hiele, 1959/1984). Pierre og Dina van Hiele beskriver at forståelsen av geometri er utviklet gjennom nivåer som originalt presenteres som diskrete nivå fra 0 til 4, blant annet i «The child's thought and geometry» av van Hiele (1959/1984). Videre vil nivåene presenteres nærmere, beskrivelsen av nivåene tar utgangspunkt i van Hiele (1959/1984), men er også inspirert av Burger og Shaughnessy (1986) og Clements og Battista (1992) sin forklaring av nivåene.

Nivå 0, visualisering. Elevene dømmer figurer etter utseende og kan resonnere om grunnleggende geometriske begreper. Elevene kjenner igjen et bilde av rektangel på grunn av formen, og kan skille det fra et kvadrat. På dette nivået er ikke en rombe et parallellogram, men en helt annen figur. Eleven vil ikke ha kjennskap til alle egenskaper til figuren.

Nivå 1, analyse. På dette nivået er figurene bærere av sine egenskaper. Det vil si at eleven kan gjenkjenne og analysere figurer gjennom deres egenskaper. En figur er et rektangel dersom den har fire rette vinkler, diagonalene er like, og motstående sider er like. Egenskapene er derimot ikke systematisert, og sammenhengen mellom figurer som kvadrat og rektangel er enda ikke lagt merke til.

Nivå 2, abstraksjon. Egenskapene til figurene er systematisert. Det betyr at eleven kan ordne figurer logisk, forstår sammenhengen mellom figurer, og viktigheten av nøyaktige definisjoner. Kvadratet gjenkjennes som et spesialtilfelle av rektangel, fordi det er definisjonen som er gjeldene.

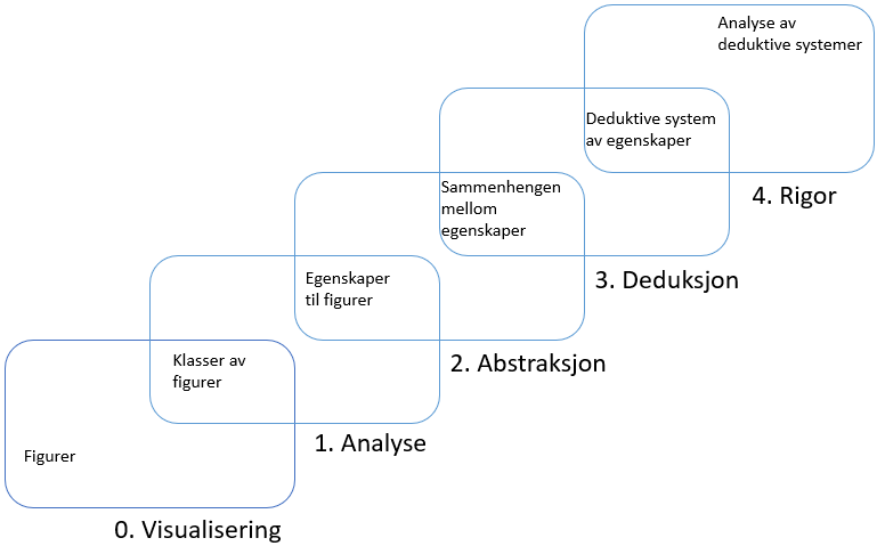
Nivå 3, deduksjon. Eleven resonnerer formelt innenfor et matematisk system. Eleven forstår rollen til og kan ta i bruk postulater, teoremer og bevis.

Nivå 4, rigor. Dette nivået er det mest avanserte som beskrives av van Hiele. På dette nivået kan eleven sammenligne systemer basert på forskjellige aksiomer, og studere ulike geometrier, slik som ikke-Euklidsk geometri. Elever i grunnskolen og videregående når sjelden dette nivået.

Hvert nivå har sitt eget språk, med symboler og et system for sammenhenger mellom symbolene. En beskrivelse som er riktig på et nivå kan virke feil på et annet nivå. To personer som resonnerer på forskjellige nivåer har problemer med å forstå hverandre, fordi ingen klarer å følge tankeprosessen til den andre (van Hiele, 1959/1984). Et eksempel kan være at rektangel kan ha forskjellig betydning for to elever på forskjellig nivå, dersom den ene tenker på det visuelle utseendet, og den andre tenker på det som en bærer av egenskapene (Clements og Battista (1992).

Van de Walle, Karp og Bay-Williams (2015) har illustrert van Hiele-nivåene i en modell (se figur 1). Figuren viser at på hvert nivå blir ideene som skapes fokus eller gjenstand for

tanke på neste nivå. Figur 1 viser til at tankeproduktene på hvert nivå er de samme som gjenstandene for tanke på det neste. Gjenstandene (ideene) må opprettes på ett nivå, slik at forhold mellom disse objektene kan bli fokus på neste nivå (Van de Walle et al., 2015). Et eksempel på dette er at gjenstanden på nivå 0 er figurer, og hva de ligner på, mens gjenstanden for tanke på nivå 0 er klasser av figurer, som ser ut til å være like. Videre er klasser av figurer gjenstanden på nivå 1, og egenskapene til figurer bli gjenstanden for tanke på nivå 1.



Figur 1. Oversatt modell av van Hiele-nivå (Van de Walle et al., 2015, s. 428).

Modellen i figur 1 er oversatt fra engelsk, jeg har imidlertid bevisst valgt å erstatte det Van de Walle et al. (2015) beskriver som uformell deduksjon (informal deduction) med abstraksjon (abstraction), slik det beskrives av van Hiele (1959/1984), fordi kjennetegnene på nivået er like.

Ulik litteratur navngir også nivåene forskjellig, noen bruker de originale 0-4 nivåene, mens andre bruker 1-5. Clements og Battista (1992), som bruker nivåene fra 1-5, beskriver at det også kan eksistere et nivå før det visuelle nivået, som van Hiele (1959/1984) beskriver som nivå 0, slik at den geometriske forståelsen i realiteten kan deles inn i 6 nivåer. Elever som er på dette pre-nivået er ikke i stand til å gjenkjenne mange forskjellige figurer, det gjør at de kanskje kan skille mellom figurer som har kurver, men ikke rette streker. De kan for eksempel skille mellom kvadrat og sirkel, men de klarer ikke å skille mellom trekant og firkant. Det finnes flere som nummererer nivåene fra 1 til 5, blant annet Battista (2007). Denne oppgaven tar utgangspunkt i van Hiele (1959/1984) sin beskrivelse av nivåene fra 0 til 4, men det suppleres også med andre sine tolkninger av nivåene.

van Hieles nivå-teori er mye omtalt, og står fortsatt står sterkt i forskningsfeltet. Likevel presenterer Gutiérrez, Jaime og Fortuny (1991) en alternativ måte å analysere elevers geometriske forståelse og evne til resonnering på, med utgangspunkt i nivåene. Flere forskere har tidligere bestemt elevers nivå ut fra en vurdering av en skriftlig test (Usiskin, 1982), alternativt ved hjelp av et mer omfattende og tidskrevende intervju (Burger & Shaughnessy, 1986). Det slike tester ikke tar høyde for, men som beskrives av Gutiérrez et al. (1991), er at det viser seg at elever kan resonnere på to nivåer samtidig. Denne forskningen baserer seg på tredimensjonale figurer, i motsetning til annen nevnt forskning som tar utgangspunkt i todimensjonale. Det konkluderes med at van Hiele-nivåene derfor ikke kan sees på som diskrete, og at det er nødvendig å studere overgangen mellom nivåer nærmere. Nivåene omtales som en skala på 0-100, der det er mulig gå videre på et nytt

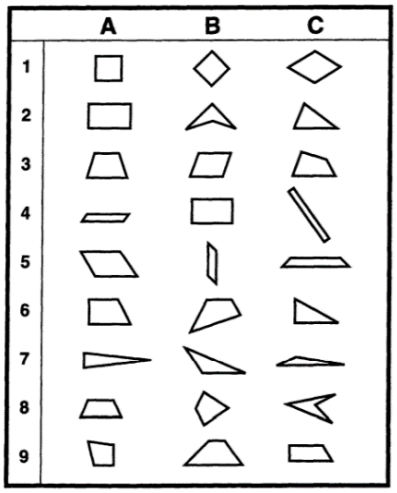
nivå før man er på 100 i det forrige. Forskningen til Gutiérrez et al. (1991) viser at elever kan være på en høyere grad på nivå 3 enn på nivå 2, noe som viser at de ikke nødvendigvis i like stor grad er hierarkiske. Burger og Shaughnessy (1986) beskriver også i sin undersøkelse at selv om van Hiele har strukturert nivåene slik at de er diskrete, så forekommer det tilfeller der det er vanskelig å skille mellom nivåer for enkelte elever. De kan derfor befinne seg på forskjellige nivåer på forskjellige oppgaver.

2.1.2 Matematisk resonnering

Selv om matematisk resonnering er et sentralt tema innenfor læreplaner og matematikkundervisning, er det er ikke alltid klart definert hva matematisk resonnering er. Man antar ofte at alle har en formening om hva man legger i begrepet (Jeannotte & Kieran, 2017). Matematisk resonnering er et begrep som brukes ulikt i forskningslitteraturen. Ofte med en implisitt antagelse om at det er enighet om hva det betyr, men uten en eksplisitt definisjon (Jeannotte & Kieran, 2017; Yackel & Hanna, 2003).

Jeannotte og Kieran (2017) har med bakgrunn i en litteraturstudie forsøkt å klargjøre hva matematisk resonnering er, og hvordan det ser ut i skolen. Her beskrives en modell med ulike trekk som går igjen i litteraturen. Definisjonen de kommer frem til kan beskrives som det å slutte matematiske ytringer fra andre matematiske ytringer. De inkluderer blant annet det å finne mønster, generalisere, og formulere formodninger. I tillegg ulike typer begrunnelser for påstander, inkludert deduktive bevis (Jeannotte & Kieran, 2017). Forfatterne differensierer mellom resonnering og bevis, der sistnevnte blir ansett som en underkategori og prosess av førstnevnte. Resonnering og bevis er nært knyttet til det å forklare og begrunne (Yackel & Hanna, 2003). Jeg velger videre i denne oppgaven å bruke betegnelsen resonnement om elevenes forklaringer og argumentasjon på oppgavene. Altså det elevene sier når de besvarer oppgavene, og spørsmål som stilles rundt oppgaven.

I denne studien er fokuset å undersøke matematisk resonnering om geometriske figurer. Lehrer (1998) sitt rammeverk for geometrisk resonnering presenteres derfor videre. Piaget og van Hiele regnes av Lehrer (1998) som pionerne innenfor forskning på elevers forestillinger om rom og geometriske figurer. Store deler av dette arbeidet ble imidlertid utført for flere tiår siden. Mer moderne forskning har ikke omfavnet det store spekteret av matematiske begreper og mentale ferdigheter som er karakteristiske for det tidligere verket. Lehrer (1998) sitt rammeverk for geometrisk resonnering forsøker å utvikle en mer moderne måte å beskrive barns utvikling av ferdigheter innenfor resonnering av rom på. Formålet var å beskrive utviklingen av spatial resonnering, sånn at lærere kan bruke elevers kunnskap i geometri for veiledning (Lehrer, 1998). I motsetning til tidligere tester av van Hiele-nivå ble det utviklet en test som skulle undersøke elevers tanker om mange flere ulike former og figurer. Elevene skulle sammenligne og beskrive forskjellen på tre grupper av figurer, for å undersøke spatial resonnering (Lehrer, 1998). Oppgaven som ble brukt (se figur 2) for å undersøke elevens forståelse av todimensjonale figurer, var å sammenligne hvilke to figurer som var mest like, og begrunne valget muntlig. Triadene med figurer var designet for å muliggjøre ulike former for resonnering, med bakgrunn i van Hiele hierarkiet (Lehrer, 1998).



Figur 2. Test av resonnering om todimensjonale figurer (Lehrer, 1998, s. 140)

Undersøkelsene gikk ut på at elevene skulle gruppere og sammenligne ulike geometriske figurer. I sine undersøkelser har Lehrer (1998) funnet åtte kategorier om resonneringen rundt geometriske figurer. Kategoriene som videre beskrives mer detaljert er likheter, størrelser, vinkel, retning, omforming, telling, egenskaper og klasse.

Likhet går ut på at figurene ligner på et annet objekt. Den kan ligne på en prototype av figuren, at den for eksempel er kvadratisk. Den kan ligne på ekte objekter, rektangler kan for eksempel ligne på en dør. Hvis elevene fokuserer på *størrelsen* til figuren, figuren kan være tynn, tykk, liten, stor eller lang, er det innunder kategorien størrelse. Kategorien *vinkel* går ut på at elevene bruker vinklene i sine resonnementer, de kan beskrive vinklene som stumpe, spisse eller rette. Kategorien vinkel blir brukt på en slik måte at det ikke er en avgjørende egenskap for klassifisering av figuren. *Retning* går ut på rotasjonen til figuren. Retning kan beskrives i form av horisontal, vertikal, på siden, eller «feil vei». *Omforming* er en metode der eleven forestiller seg at en figur kontinuerlig blir endret til en annen figur. Eleven kan si at man kan rette ut en vinkel, eller dytte figuren ned, og lage den om til noe annet. *Telling* går ut på at elevene teller antall sider eller vinkler, og bruker det som et resonnement. Eksempler kan være at figuren har fire sider, eller at den har tre kanter. Disse seks kategoriene er det Lehrer (1998) kaller visuelle, og elever som bare bruker disse befinner seg på nivå 0 av van Hiele-nivåene.

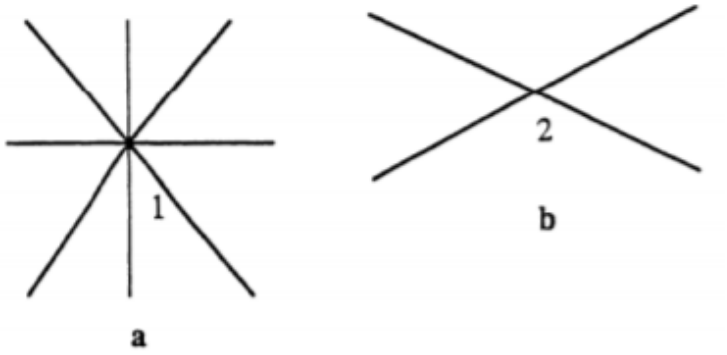
Elever som tar i bruk viktige *egenskaper* som er av betydning for figuren, benytter et resonnement innenfor kategorien egenskaper. Dette er på et høyere nivå og kjennetegner nivå 1. Viktige egenskaper kan være at figurene er mest like fordi begge har fire rette vinkler, eller begge figurene har to parallelle linjer. Elevene som konsekvent brukte *klasse* som resonnement, befinner seg på nivå 2 av van Hiele-nivåene (Lehrer, 1998). Det vil si at elevene mener figurene er mest like på bakgrunn av at de begge for eksempel er rektangler.

2.1.3 Figurale begrep

Fischbein (1993) beskriver to forskjellige kategorier av menneskets mentale realitet, som er mentale bilder og begrep. Der et mentalt bilde er en representasjon av et objekt eller fenomen. Mens et begrep uttrykker en mer abstrakt idé, som noe generelt, og en representasjon av en klasse objekter basert på deres felles egenskaper. Noe av det som gjør geometriske objekter eller figurer spesielle, er at de ikke passer inn i en av disse beskrivelsene. Selv om geometriske objekter kan sies å ha egenskapene til et begrep,

mener Fischbein (1993) at de også besitter egenskaper som et begrep ikke har. Geometriske objekter inkluderer også mentale representasjoner av romlige egenskaper. Mentale bilder besitter ikke det generelle og perfeksjonen, det abstrakte som kreves innenfor matematikk. Derfor presenterer Fischbein (1993) et mentalt fenomen for objekter man undersøker i geometrisk resonnering. Han navngir det figurale begrep (figural concepts). Figurale begreper skal inkludere de spatiale egenskapene, som også besitter begrepsmessige kvaliteter.

Før en elev har utviklet en forståelse som blander den figurale og begrepsmessige forståelsen til et figuralt begrep, kan det oppstå situasjoner som skaper en konflikt mellom de to systemene. Fischbein (1993) viser til et konkret eksempel der en elev tar i bruk det figurale begrepet i en situasjon, mens i en annen situasjon tar den visuelle fremstillingen overhånd over den begrepsmessige begrensningen. Eksemplet som benyttes for å fremheve dette fenomenet er en elev som mener at et punkt ikke har noen dimensjoner og at man derfor ikke kan si at et punkt er større eller tyngre enn andre punkter.



Figur 3. Oppgave hentet fra Fischbein (1993, s. 146).

Når eleven senere får spørsmål om hvilket punkt av 1 og 2 (se figur 3) som er størst og tyngst, tar den figurale fremstillingen overhånd, og eleven svarer: «Point one is bigger because more lines intersect. Point 2 is smaller because less lines intersect. The point have the same weight» (Fischbein, 1993, s. 147). Dette eksemplet viser at en visuell fremstilling kan ta overhånd over den begrepsmessige forståelsen, og at det oppstår en konflikt der de to systemene ikke har dannet et figuralt begrep.

2.2 Algoritmisk tenkning

Computational thinking er et begrep som ble introdusert av Seymour Papert allerede på 1980-tallet (Papert, 1980), men som ble gjenopptatt og popularisert av informatikeren Wing (2006). I sin artikkel om computational thinking argumenterer Wing (2006) for at dette er en egenskap alle har bruk for, ikke nødvendigvis bare om man er dataforsker. Hun beskriver computational thinking som å løse problemer, designe systemer og forstå menneskelige handlinger. Dette ved å bygge videre på begreper som er sentrale innenfor datavitenskap.

Utdanningsdirektoratet (2019) har valgt å oversette computational thinking til algoritmisk tenking i fagfornyelsen. Der beskrives algoritmisk tenking i kjerneelementet «Utforsking og problemløsning», som en viktig prosess med å utvikle strategier og fremgangsmåter for å løse et problem. Og for å vurdere om det er best egnet å løse problemet med eller uten digitale verktøy (Utdanningsdirektoratet, 2019). I denne studien velger jeg å oversette Computational Thinking (CT) til algoritmisk tenkning (AT), selv om det i ulik litteratur er forskjeller i den norske oversettelsen av begrepet. Computational thinking oversettes blant annet til algoritmisk tenkning (Gjøvik & Torkildsen, 2019), algoritmisk tankegang (Sevik, 2016), og algoritmisk tenking (Utdanningsdirektoratet, 2019). Gjøvik og Torkildsen (2019) viser til at det ikke er en fullstendig overensstemmelse mellom norske og utenlandske termer når det er snakk om AT, og at man derfor kan finne ulike definisjoner i forskjellig litteratur. Jeg vil presisere at jeg i denne studien legger det samme i algoritmisk tenkning som i computational thinking.

2.2.1 Komponenter i algoritmisk tenkning

Shute, Sun og Asbell-Clarke (2017) gjennomførte en litteraturstudie som viser til en uenighet i hvordan begrepet AT defineres innenfor forskningsfeltet. Formålet med litteraturstudien var å utforske hva AT beskrives som, hvordan det karakteriseres, utvikle en tydelig definisjon, og et rammeverk som kan benyttes for blant annet vurdering i grunnskolen. Jeg har i denne studien valgt å ta utgangspunkt i Shute et al. (2017) sin definisjon av begrepet algoritmisk tenkning, som bygger på en litteraturstudie av 15 artikler:

> *«The conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts» (Shute et al., 2017, s.5).*

AT beskrives altså som et fundament for å løse problemer effektivt, med eller uten hjelp av datamaskiner. Løsningene som man kommer frem til skal kunne brukes for å løse problemer i andre kontekster, altså andre situasjoner som ligner. Shute et al. (2017) beskriver fire viktige komponenter innenfor AT som går igjen i forskningen: dekomponering, abstraksjon, algoritmer, feilsøking, men legger også til to egne kategorier som er iterasjon og generalisering.

De ulike kategoriene brukes som en systematisk måte å løse ulike problemer på. Kort forklart beskriver Shute et al. (2017) dekomponering som å bryte ned komplekse problemer i mindre deler, og systematisk løse de ulike delproblemene som til sammen utgjør hele problemet. Abstraksjon handler om å finne mønster innenfor problemer og løsninger, som igjen kan brukes for å løse lignende problemer. Abstraksjon beskrives også som det å ekstrahere essensen av et system. Designet av algoritmer utvikler verktøy og prosedyrer som kan brukes for å løse problemer. Feilsøking er å identifisere feil, og fikse feilen når løsningen ikke virker som den skal. Tabell 1 viser en oversikt over de ulike komponentene med tilhørende definisjoner, oversatt fra engelsk til norsk.

Komponenter	Definisjon (Definition)
Problemnedbryting	Dele opp et komplekst problem/system til håndterlige deler. De ulike delene er ikke tilfeldige, men funksjonelle elementer som kollektivt utgjør hele problemet/systemet.
Abstraksjon	Ekstrahere essensen av et (komplekst) system. Abstraksjon har tre underkategorier: 1. Datainnsamling og analyse: Samle inn den mest relevante og viktige informasjonen fra flere kilder, og forstå forholdet mellom de ulike datasettene. 2. Gjenkjenne mønstre: Identifisere mønstre/regler som ligger i dataene/informasjonen. 3. Modellering: Bygge modeller eller simuleringer som representerer hvordan et system opererer, og/eller hvordan et system vil fungere i fremtiden.
Algoritmer	Designe logiske og strukturerte instruksjoner for å gi en løsning til et problem. Instruksjonene kan utføres av en datamaskin eller et menneske. Det er fire underkategorier: 1. *Algoritmisk design:* Lage et sett med strukturerte steg for å løse et problem. 2. *Parallellisme:* Utføre flere steg på samme tidspunkt. 3. *Effektivitet:* Designe færrest mulige steg for å løse et problem ved å fjerne overflødige og unødvendige steg. 4. *Automatikk:* Automatisere utførelsen av prosedyren når det kreves, for å løse liknende problemer.
Feilsøking	Detektere og identifisere feil, og fikse feilen når løsningen ikke virker som den skal.
Iterasjon	Gjenta designprosessen for å raffinere løsningen, til et ideelt resultat er oppnådd.
Generalisering	Overføre CT ferdigheter til et bredt spekter av situasjoner/domener, for å løse problemer effektivt.

Tabell 1. Komponenter innenfor algoritmisk tenkning (Shute et al., 2017, s. 12).

Den oversiktlige fremstillingen av komponenter som presenteres av Shute et al. (2017) passer bra i min undersøkelse, siden hensikten med litteraturstudien er å bistå den pedagogiske utviklingen av AT i skolen. Definisjonen som presenteres av Shute et al. (2017) baserer seg på en analyse av relevant forskningslitteratur, tar for seg de vanligste komponentene som benyttes i AT, og retter seg mot aldersgruppen 5-18 år. Jeg velger likevel å supplere med tidligere forskning knyttet til komponentene AT består av. Bakgrunnen for dette er ønsket om å gå grundigere i hver enkelt komponent. Dette for å få en tydeligere forståelse, noe som vil være nyttig i analysen, men også fordi jeg mener Shute et al. (2017) sin beskrivelse i litteraturstudien fremstår som noe forenklet.

2.2.2 Tidligere forskning på komponentene

For å presentere en mer utdypende forklaring velger jeg å se nærmere på ulik forskning knyttet til komponentene AT består av. Iterasjon og generalisering er komponenter Shute

et al. (2017) selv har valgt å legge til, men siden de ikke er sentrale komponenter som går igjen i litteraturen velger jeg å ikke beskrive de nærmere.

Problemnedbrytning er den første komponenten i tabellen til Shute et al. (2017). Anderson (2016) beskriver AT som en tilnærming til problemløsning som ofte benyttes av programmerere, og at dette er ferdigheter som trengs i fremtiden, på samme måte som lesing, skriving og aritmetikk. Problemnedbrytning brukes for å forenkle prosessen, og hjelpe problemløseren med å identifisere delproblemer som kan løses individuelt (Anderson, 2016). Shute (1991) beskriver problemnedbrytning som en ferdighet innen problemløsning, som er viktig for å lære seg programmering. Det handler om å dele problemer i elementer, dette for å få en oversikt over problemløsningen. Barr og Stephenson (2011) beskriver problemnedbrytning som å bryte problemer ned til mindre deler, som enklere kan løses.

Abstraksjon beskrives på ulike måter i litteraturen, og består av flere forskjellige underkategorier. Shute et al. (2017) deler abstraksjon i datainnsamling og analyse, gjenkjenne mønstre, samt modellering. Informatikeren Wing (2006) beskriver abstraksjon som hovedelementet under AT, der personer innhenter relevant informasjon og forkaster irrelevant data, for å generere mønster, og finne sammenhenger mellom ulike representasjoner. Barr og Stephenson (2011) beskriver abstraksjon som å forenkle fra det konkrete til det generelle, når man finner løsninger. Beskrivelsen stemmer godt overens med Shute et al. (2017) sin forklaring av abstraksjon, som er å identifisere underliggende regler, og gjenkjenne mønstre. Videre beskriver Anderson (2016) at man lager abstrakte representasjoner av mønstret, når det identifiseres slik at det kan brukes for å løse lignende problemer.

Algoritmer beskrives i fire underkategorier som er algoritmisk design, parallellisme, effektivitet og automatikk (Shute et al., 2017). Felles for underkategoriene er at det lages algoritmer som er instruksjoner for å løse et problem. Det kan gjøres av en datamaskin eller et menneske. En algoritme er en serie av instruksjoner som må følges i en bestemt rekkefølge med et definert startpunkt og sluttpunkt (Anderson, 2016). Det poengteres av Anderson (2016) at programmering ikke er nødvendig for en algoritme, men at det kan være en hvilken som helst serie av instruksjoner for å oppnå et ønsket mål.

Feilsøking, eller «debugging», er den siste av komponentene som omtales nærmere. Debugging er en viktig arbeidsmåte innenfor programmering, når man skriver instruksjoner og koder (Papert, 1980). Det går ut på å se etter feil, og deretter rette opp feilen, slik at man kan finne en bedre løsning på problemet. Anderson (2016) beskriver at evaluering av algoritmen er det siste steget i en AT-prosess, og at innenfor programmering er det siste steget feilsøking av koden. Weintrop et al. (2016) beskriver at det finnes mange ulike strategier for å feilsøke. Det kan være systematisk testing, eller å løse problemet flere ganger, men at fellesbetegnelsen for disse strategiene innenfor programmering er feilsøking.

The Barefoot Programme er et program som skal støtte lærere med undervisning av IKT, og bidra til at elever i grunnskolen blir «algoritmiske tenkere». Det defineres seks konsepter, og fem tilnærminger til AT. De seks konseptene er logikk, algoritmer, dekomposisjon, mønstre, abstraksjon og evaluering. De fem tilnærmingene er eksperimenterende, skapende, feilsøkende, utholdende og samarbeidende (The Barefoot Project, 2014). Konseptene som presenteres her har flere fellestrekk med komponentene som beskrives over.

Tabell 1 som presenteres av Shute et al. (2017) beskrives som generell, og fokuserer på det underliggende konseptuelle grunnlaget som kreves for å undersøke problemer via et AT-perspektiv. Og hvordan dette perspektivet kan fremheves og støttes i nåværende skolefag. Det beskrives at denne tilnærmingen blant annet kan brukes i både matematikk, naturfag og engelsk. Weintrop et al. (2016) argumenterer også for å inkludere AT i matematikk og naturfag, der begge beskrives som meningsfulle kontekster for å utvikle begreper og ferdigheter innenfor AT. Den relativt vide tilnærmingen av AT som presenteres av Shute et al. (2017) og Weintrop et al. (2016) er en motsetning til andre modeller som fokuserer på spesifikke fagområder. For eksempel Brennan og Resnick (2012), som begrenser sin modell av AT til begreper innenfor koding (Shute et al., 2017). Selv om Brennan og Resnick (2012) i likhet med Shute et al. (2017) fokuserer på både vurdering og utvikling av AT i skolen, retter de forskningen sin mot programmering. Spesielt for yngre personer, og med bruk av programmet Scratch.

Papert (1980) var tidlig ut med å hevde at datamaskiner over tid ville bli en stadig mer sentral del av hverdagen vår, og at kunnskaper som AT ville bli viktige. Wing (2006) beskrev videre i 2006 at AT ville bli en fundamental kunnskap innenfor alle disipliner, og at det fører til nye muligheter for å løse problemer både i en virtuell og ekte verden. Vi ser nå for første gang at programmering innføres i nye læreplaner i Norge fra 2020. På samme tid innføres også AT i kjerneelementer. Et naturlig miljø å innføre algoritmisk tenkning i skolen er via programmering. Programmering trenger ikke nødvendigvis å skje på en datamaskin med et nytt språk, eller notasjonssystem. En kan jobbe både med programmering og algoritmisk tenkning på mange nivåer og klassetrinn (Gjøvik og Torkildsen, 2019).

2.3 Programmering – et overblikk

Logo var det første programmeringsspråket som rettet seg mot matematikkundervisning, det ble utviklet på slutten av 60-tallet av Wallace Feurzeig og Seymour Papert (Kaput, 1992). Selv om programmering i undervisningen ikke slo helt gjennom den gang, bygger store deler av dagens programmer på deres tanker og ideer, som for eksempel Scratch (Lye & Koh, 2014). Jeg velger videre å presentere et overblikk innenfor programmering, spesielt rettet mot matematikkundervisning og geometri. Dette inkluderer et historisk perspektiv på programmering i matematikkundervisning, og blokkbasert programmering i Scratch.

2.3.1 Programmering i matematikkundervisning

Allerede i 1980 beskriver Seymour Papert, i boken Mindstorms (Papert, 1980), hvordan datamaskiner gradvis vil bli en større del av folks hverdag. Utviklingen av programmeringsspråket Logo ble gjort med den hensikt at det kunne brukes for å innføre programmering i undervisningen, ved hjelp av en skilpadde. Eleven programmerer skilpadda til å lage figurer, og gjennom det oppdager eleven viktige matematiske ideer om figurene. Eksempler på kommandoer for å bevege skilpadden:

```
FD fremover (forward)          BK bakover (back)

LT venstre (left turn)         RT høyre (right turn)

PU penn opp (pen up)           PD penn ned (pen down)
```

De nevnte kommandoene kan kombineres med flere andre kommandoer for å bevege skilpadden på skjermen. Penn ned kan brukes for å tegne der skilpadden beveger seg, dette kan for eksempel brukes for å tegne geometriske figurer.

Mange geometriske begreper er viktige i elevens arbeid med Logo. Det kan være interessant å gi elevene i oppgave å lage en sirkel. Hvordan kan elevene lage det med framover- og roter-kommandoer? Eller hva med regulære polygoner, kan elevene finne ut hvor mange grader de må snu i en sekskant, for å treffe akkurat på utgangspunktet? Det blir viktig å ha en dyp forståelse av begreper som vinkler, kanter og polygoner, dersom man skal klare å programmere skilpadda. Kaput (1992) nevner at Logo har hatt en positiv effekt på læring. Her trekkes det fram at hos barneskoleelever har læring av geometri, klassifisering av figurer, estimering av vinkler og lengder, samt forståelse av vinkelstørrelser og retning, vært positive trender. For de eldre elevene har man funnet økning i van Hiele-nivå og i relasjonell tenkning (Kaput, 1992).

Clements og Battista (1992) viser til at ulike Logo-aktiviteter kan brukes til å oppmuntre elevene til å gå videre til nivå 1 og 2 i van Hiele-hierarkiet. For eksempel kan elever i utgangspunktet bare være i stand til å identifisere et rektangel visuelt, som samsvarer med nivå 0, men ved bruk av Logo er elevene nødt til å konstruere en sekvens med kommandoer for å tegne rektanglet. Det gjør at når man skal konstruere et rektangel ut ifra kommandoer, må elevene analysere de visuelle aspektene ved rektangelet, og reflektere over hvordan komponentdelene er satt sammen. En aktivitet som oppmuntrer til tenking på nivå 1 (Clements & Battista, 1992). Elevene blir på denne måten oppmuntret til å konstruere en definisjonsform for et rektangel, som datamaskinen forstår (Clements & Battista, 1992).

Papert mener Logo er et sted der elever ikke blir dømt, et sted der det er lov til å ta feil (Papert, 1980). I programmering er det å feilsøke eller «debugge» en viktig fremgangsmåte, når man skriver instruksjoner og koder til en datamaskin. Når elever ser etter feil, og deretter retter opp, kan de finne en bedre løsning på problemet. Å lære av feil eller misoppfatninger er viktig i programmering. Det poengteres at det er viktig å ikke kritisere elevene for feil, men å i stedet snu det til noe positivt vi kan lære av (Papert, 1980). Programmer av denne typen gir ikke noen ladet tilbakemelding i det eleven gjør, blir det feil så virker det ikke.

2.3.2 Blokkbasert programmering i Scratch

Blokkbasert programmering gjør at man ikke trenger å kunne skrive i et programmeringsspråk, man bruker istedenfor ulike blokker for å sette sammen program. Dette kan sammenlignes med å sette sammen LEGO-brikker, der blokkene er designet slik at man kun kan sette de som passer sammen (Resnick et al., 2009). Dette gjør at syntaksfeil er umulig, og at eventuelle feil derfor utelukkende skyldes den algoritmiske fremgangsmåten i sammensettingen av blokkene (Foerster, 2016).

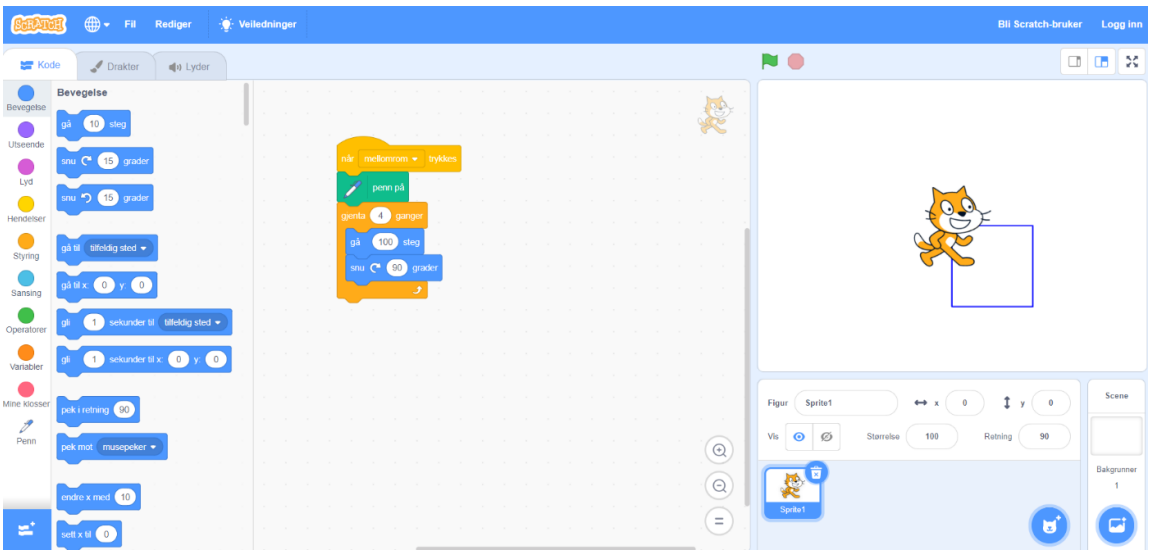
Programmeringsspråket Scratch er utviklet av «Lifelong Kindergarten Group» ved Massachusetts Institute of Technology Media Lab, og er et blokkbasert programmeringsspråk. Scratch er kun blokkbasert, og man ser derfor aldri den mer avanserte programkoden som ligger bak blokkene som brukes.

Figur 4. Sammenligner program i Java og Scratch av Foerster (2016, s. 92).

Programmeringsspråk som C, C++ og Java kan gjøre det vanskelig å komme i gang med programmering (Foerster, 2016). Blokkbasert programmering senker terskelen for å komme i gang, og gjør at man bruker mindre tid på å forstå selve språket. Man får derfor bedre tid til å forstå viktige ideer innenfor programmeringen. Foerster (2016) viser i figur 4 til et eksempel på forskjellen av et program i Java og Scratch.

Scratch kan enkelt kjøres i nettleseren, eller lastes ned til datamaskinen. Programmet kan brukes for å designe deres egne interaktive media. Dette inkluderer historier, spill, animasjoner og simulasjoner. Man setter sammen blokker med programmerings-instruksjoner, på samme måte som man setter samme LEGO-brikker eller puslespill (Resnick et al., 2009). Scratch er i tillegg til dette bygget opp som et nettbasert felleskap, der man kan dele prosjekter på samme måte som man gjør med videoer på YouTube (Brennan & Resnick, 2012).

Scratch inneholder mange ulike klosser, som brukes for å lage ulike program. En kloss er en instruksjon som for eksempel «gå 100 steg» eller «snu 90 grader». En blokk består av flere klosser som er satt sammen, og blir til det programmet man lager. Figur 5 viser et eksempel på hvordan man kan lage en blokk som tegner et kvadrat i Scratch. Katten som vises i det høyre feltet blir videre i oppgaven kalt for Felix[1].



Figur 5. Skjermbilde av program som tegner et kvadrat i Scratch.

[1] Katten blir gjerne tildelt ulike navn i ulike undersøkelser, men navn som ser ut til å gå igjen er blant annet Scratch cat og Scratchy.

2.4 Tidligere forskning

2.4.1 Programmering og geometri

I en studie av førskoleelever konkluderte Fessakis, Gouli og Mavroudi (2013) med at elevene på få uker, gjennom arbeid med Scratch, utviklet matematiske ferdigheter i form av forståelse for vinkler, grader, telling og orientering i rutenett.

Calder (2010) beskriver Scratch som både morsom og enkel å bruke for arbeid med problemløsning. Samtidig er det også et nyttig miljø for å lære matematiske konsepter. Han viser til funn i sin studie av førsteklassinger, at elevene i ulik grad utviklet en forståelse av vinkler, posisjon med utgangspunkt i koordinater og spatial forståelse.

Brown et al. (2013) gjennomførte en studie med elever på ungdomskolen. Formålet var å undersøke bruken av Scratch for å lære ferdigheter i problemløsning. Funn fra studien viser at elevene viste engasjement i programmeringsøktene. Scratch viste seg å være et effektivt verktøy for resonnering og problemløsning i matematikk.

Foerster (2016) forklarer at elevene i sin studie hadde begrenset tilgang på datamaskiner, og derfor ble nødt til å samarbeide, noe som viste seg å være fruktbart. Elevene ble nødt til å diskutere programmene med hverandre, og Scratch bidro med et felles matematisk språk for diskusjonen.

I en eksperimentstudie av 6. klassinger analyserer Calao, Moreno-León, Correa og Robles (2015) effekten av å bruke Scratch for utvikling av matematiske ferdigheter. Resultatet fra studien viser at det er en statistisk signifikant utvikling i forståelsen av matematisk kunnskap i eksperimentgruppen som arbeidet med Scratch. Det konkluderes med at Scratch bidrar til å forbedre elevenes prestasjoner innenfor matematisk modellering, resonnering og problemløsning.

2.4.2 Programmering og algoritmisk tenkning

Elever som lærer seg programmering opplever ofte utfordringer med å analysere et problem, planlegge og lage løsninger (Hazzan, Lapidot & Ragonis, 2015). Hazzan et al. (2015) mener problemet kommer av at introduksjonskursene fokuserer for lite på å lære problemløsning, og at elevene derfor ofte utvikler egne, ineffektive strategier. Strategiene domineres ofte av en prøv- og feil-mentalitet, der studentene prøver seg frem når de lager programmer (Hazzan et al., 2015). Selv om forskningen hovedsakelig baserer seg på utdanningen av lærere på videregående skole, poengterer forfatterne at kunnskapen også kan anvendes på ulike nivå.

En studie som ser på AT i lys av problemløsning, har undersøkt effekten av å introdusere AT tidlig i nybegynnerkurs i programmering (Fernández, Zúñiga, Rosas, & Guerrero, 2018). Undersøkelsen viste at AT hadde en positiv effekt på ferdigheter innen abstraksjon, problemnedbrytning, bruken av algoritmer og generelle problemløsningsferdigheter. Det er viktig å påpeke at studien ble gjennomført med studenter innenfor høyere utdanning.

Atmatzidou og Demetriadis (2016) beskriver at funn fra deres studie viser at elever over tid ser ut til å nå det samme nivået i ferdigheter innenfor AT, uavhengig av alder og kjønn. De viser også til at det tar tid å utvikle ferdighetene innenfor AT, og at elevenes resultater forbedret seg signifikant mot slutten av aktiviteten. Det konkluderes med at det er nødvendig med mange økter for å utvikle elevenes ferdigheter innen AT.

Ifølge Wohl, Porter og Clinch (2015) utvikles også forståelsen av feilsøking over tid i et sakte, men stadig økende tempo. Det beskrives derfor som en lite effektiv måte å løse problemer på, dersom man har begrenset erfaring.

Barcelos, Muñoz-Soto, Villarroel, Merino og Silveira (2018) har gjennomført en litteraturstudie med 42 artikler, der målet var å identifisere hvordan forholdet mellom matematikk og AT kommer til syne i litteraturen. Deres analyse viser at Scratch var det programmeringsmiljøet som flest benyttet seg av, og at matematikkferdigheter utvikles i aktiviteter ved bruk av dette programmeringsspråket. Scratch virker å være et sentralt program, som ofte brukes innenfor både AT, programmering og matematikk.

3 Metode

I løpet av undersøkelsen har jeg tatt en rekke metodiske valg, som har påvirket studiens datainnsamling og analysemetode. Valgene mine har dermed fått konsekvenser for studien. I metodekapittelet beskrives og begrunnes planleggingsfasen, gjennomføringen og etterarbeidet i forbindelse med datainnsamlingen i denne studien. Det beskrives og argumenteres for valg av innsamlings- og analysemetode, som er relevante for å svare på forskningsspørsmålene. Videre reflekteres det over gyldigheten av resultatene, samt etiske overveielser.

3.1 Metodisk tilnærming

Med en hypotese om at programmering kan berike undervisning i geometri, og oppmuntre elevene til å lære definisjoner og viktige egenskaper ved geometriske figurer, valgte jeg å ta i bruk en kvalitativ tilnærming for å grundig undersøke et utvalg elever på 6. trinn. Kvalitative forskere har med seg et paradigme i undersøkelsen. Dette er et sett av antagelser eller syn på verden, og som er med på å styre forskningen (Postholm, 2005). I studien jeg har gjennomført vil paradigmet være et konstruktivistisk syn på kunnskap. Det går ut på at eleven konstruerer kunnskap gjennom egenaktivitet, men at det også er i interaksjon med omverdenen. I matematikkdidaktikken forklares kunnskap i matematikk ved at individet konstruerer og strukturerer personlig kunnskap ut fra erfaringer, opplevelser og oppfatninger, gjennom møte med omverden (Botten, 1999).

Creswell og Poth (1998) definerer kvalitativ forskning som en undersøkelse av menneskelige prosesser i deres naturlige setting. Slike studier kalles derfor ofte naturalistiske. Kvalitativ forskning innebærer å utforske menneskelige prosesser eller problemer i en virkelig setting, å forske kvalitativt innebærer derfor å forstå deltakernes perspektiv (Postholm, 2005). Teori er den kvalitative forskerens redskap for å forske på, og dermed forstå praksis i ulike kontekster. Postholm (2005) beskriver at det er en nær sammenheng mellom forskerens teoretiske ståsted, spørsmålene som stilles, metoder som blir valgt, og dermed måten materialet blir samlet inn, analysert og tolket på. Innenfor kvalitativ forskning er det en konstant interaksjon mellom teori som studeres, og data som samles inn. Slike studier befinner seg derfor ofte mellom det som beskrives som en induktiv og deduktiv tilnærming. En induktiv tilnærming innebærer at forskeren tar hensyn til situasjonelle betingelser, mens en deduktiv tilnærming vil være at forskeren har utarbeidet et sett med variabler som ikke endres underveis (Postholm, 2005). Innen kvalitativ forskning har forskeren ofte noen undersøkelsesspørsmål klare på forhånd. Disse gir uttrykk for egne antagelser, med utgangspunkt i teori eller erfaringer, men disse kan endre seg underveis i forskningen (Postholm, 2005). Teori og forskerens egne opplevelser og erfaringer gir retning for forskningsarbeidet, og påvirker derfor forskningsfokuset (Postholm, 2005). Min studie befinner seg et sted mellom en induktiv og deduktiv tilnærming. Dette fordi det på forhånd av datainnsamlingen ble formulert en problemstilling med utgangspunkt i teori, men datamaterialet gjorde at jeg underveis måtte gjøre endringer på blant annet problemstillingen. Denne oppgaven inneholder derfor elementer av både induktiv og deduktiv metode.

3.1.1 Kasusstudie

Når man skal bestemme seg for hvilken metode som passer best for en undersøkelse er den viktigste faktoren forskningsspørsmålet som stilles. Kasusstudier er generelt foretrukne når spørsmål som «hvordan» eller «hvorfor» stilles, når den som undersøker har lite kontroll over det som skjer, og når fokuset er et fenomen i en virkelighetsnær kontekst (Yin, 2009). Kasusstudier brukes mye innenfor utdanningsforskning. Det som kjennetegner kasusstudier er at forskeren henter inn mye informasjon fra noen få enheter, gjennom en detaljert og omfattende datainnsamling (Christoffersen & Johannessen, 2012). En kasusstudie handler i korte trekk om å samle inn mye data om et avgrenset fenomen, for å beskrive, forklare, forstå, vurdere og utforske det. Kasusstudier kan bidra til å gi leseren et mer virkelighetsnært og tydelig bilde av mennesker innenfor en gitt kontekst, sammenliknet med hva bare abstrakte teorier kan gjøre (Cohen, Manion & Morrison, 2017).

Det som kjennetegner en god forsker innenfor kasusstudier, er egenskapen til å samle inn datamateriale som passer for formålet til studien, og egenskaper til å gå i dybden til et fenomen (Cohen et al., 2017). Forskeren må være god til å stille spørsmål, høre på andre, trekke slutninger og være fleksibel. Det er også viktig å ta en klar retning i datainnsamlingen, og bruke metoder som egner seg. Noe som er viktig for at forskeren skal ta riktige valg både før, underveis og i etterarbeidet, er at personen har fagkunnskap på det området som undersøkes. Det bygger også troverdighet ovenfor informantene som undersøkes (Cohen et al., 2017). Det er ingen fasit for hvordan kasusstudier gjennomføres, og forskeren har derfor relativt frie hender (Stake, 1995). Stake (1995) definerer et kasus som et avgrenset system, og skiller mellom tre ulike typer kasusstudier:

Intern kasusstudie - Der vi ønsker å lære om akkurat dette spesielle tilfellet. Forskeren studerer kasuset, uten å på forhånd ha identifisert et problem som kasuset skal representere.

Instrumentell kasusstudie - Der kasuset er et instrument for å lære om noe annet. Forskeren identifiserer et problem som skal undersøkes, før det velges et avgrenset kasus for å illustrere problemet.

Multippel kasusstudie - Der flere kasus har noe til felles. Forskeren identifiserer først et problem som skal undersøkes, før det velges flere avgrensede kasus for å illustrere problemet.

Jeg har i min masteroppgave valgt å gjennomføre en multippel kasusstudie. Beslutningen om å gjøre en kasusstudie ble gjort fordi jeg ønsket å undersøke nærmere hvordan elever arbeider med programmering. Dette for å få en mer utfyllende kunnskap om hvordan det kan gjennomføres allerede på mellomtrinnet. Hvert av de to elevparene jeg undersøker er en egen kasus, der dataene analyseres ved at hvert kasus presenteres adskilt. En fordel med kasusstudier er at forskeren har mulighet til å benytte seg av mange forskjellige metoder for datainnsamling. I denne masterstudien er lydopptak av elevsamtaler, skjemopptak, samt intervjuer, verktøy for å samle empiri. Det skyldes at forskningsspørsmålene søker svar på hvordan elevene resonnerer når de programmerer geometriske figurer, og hvordan selve programmeringen utføres. Det var viktig å bruke egnede redskaper for datainnsamling til prosjektet, som gjør det mulig å få innsikt i hvordan elevene arbeider med programmering i matematikk.

3.2 Utforming av programmeringsaktiviteter

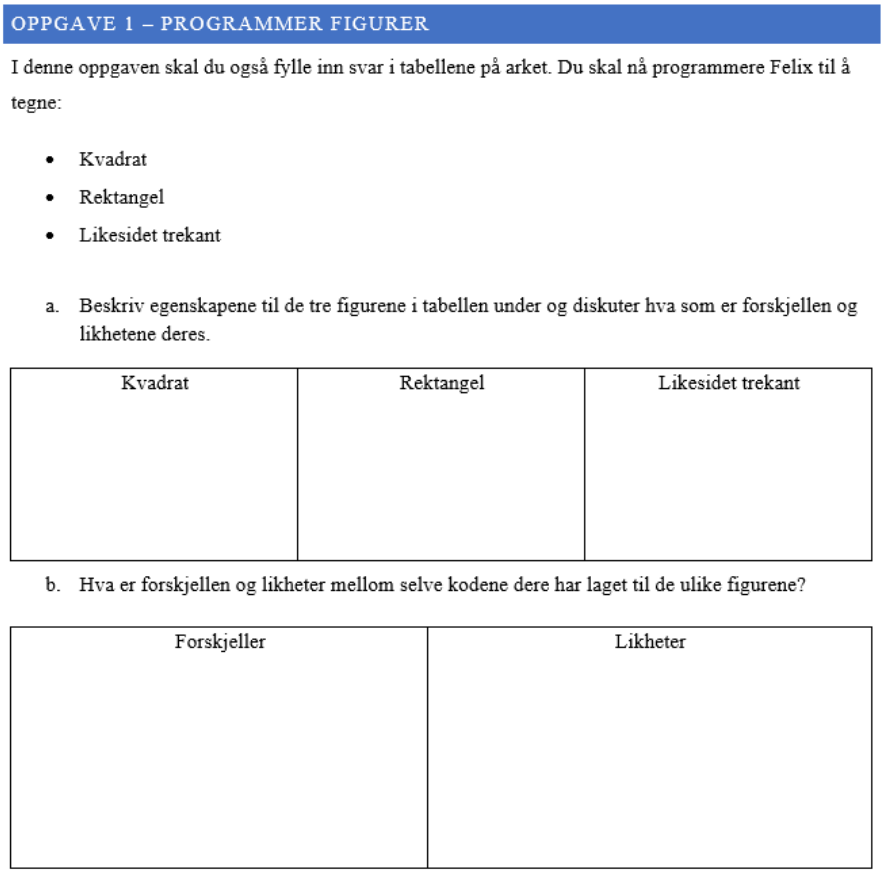
I planleggingsfasen kan det være utfordrende å finne oppgaver med riktig vanskegrad, som skal fungere både matematisk og programmeringsmessig. Spesielt vanskelig kan det være dersom man ikke kjenner elevene fra før av, og derfor ikke kjenner ferdighetsnivået. For å lage programmeringsaktivitetene valgte jeg å ta utgangspunkt i nye kompetansemål fra fagfornyelsen, hente inspirasjon fra tidligere forskning, samt rådføre meg med elevenes matematikklærer.

Jeg har latt meg inspirere av det overordnede arbeidet i Scratch til Brennan og Resnick (2012), beskrivelsen av geometriske oppgaver for 6.- og 7. klassinger av Foerster (2016) og ScratchMaths prosjektet til Benton, Hoyles, Kalas og Noss (2016). Litteraturstudier innenfor programmering gjennomført av både Forsström og Kaufmann (2018), og Barcelos et al. (2018), viser til at visuelle programmeringsspråk som Logo og Scratch ofte brukes innenfor geometri i planet. Det egner seg godt til å arbeide med firkanter, trekanter, sirkler og vinkler. Scratch er spesielt godt egnet til å arbeide med geometri. Dette fordi det er visuelt, og har et stort bibliotek med klosser for å behandle og konstruere geometriske figurer, ved bruk av blant annet lengder, vinkler, retning, koordinater og rotasjon.

Jeg valgte å gjennomføre programmeringsaktivitetene i to deler. Hovedmålet for den første undervisningsøkten var å gi elevene oppgaver som gjorde at de kunne utforske og lære seg programmeringsspråket, uten fokus på matematikk. Denne økten vil ikke bli brukt som datamateriale i denne studien, men kan finnes som vedlegg 1. Den andre undervisningsøkten (se vedlegg 2) fokuserte på å koble sammen programmeringen med matematikk, spesielt geometriske figurer. På den måten var det mulig å arbeide mot kompetansemålet der elevene skal kunne «bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønster» (Utdanningsdirektoratet, 2019). Hver av de to undervisningsøktene varte i 120 minutter. Elevene arbeidet sammen i par med en datamaskin tilgjengelig. Det var en uke mellom de to øktene, og dette ga meg tid til å eventuelt endre på opplegget mellom øktene. Jeg endte likevel med å ikke endre noe på oppgavene etter gjennomføringen av den første økten. Dette fordi nivået på oppgavene virket å passe godt med både programmeringsnivå hos elevene, og tiden jeg disponerte. Videre vil oppgavene fra den andre økten beskrives, og det begrunnes hvorfor akkurat disse ble valgt for å besvare forskningsspørsmålene.

3.2.1 Oppgave 1

Den første oppgaven tar utgangspunkt i at elevene skal programmere Felix til å tegne et kvadrat, et rektangel og en likesidet trekant. Formålet med denne oppgaven er at elevene skal utforske figurene, samt diskutere forskjeller og likheter i figurenes egenskaper, slik det beskrives i kompetansemål fra 6. trinnet. Eleven skal kunne *«beskrive eigenskapar ved og minimumsdefinisjonar av to- og tredimensjonale figurar og forklare kva for eigenskapar figurane har felles, og kva for eigenskapar som skil dei frå kvarandre» (Utdanningsdirektoratet, 2019)*. Det er for eksempel ikke nok å bare vite hvordan et kvadrat ser ut, elevene må også vite egenskapene, for å kunne programmere figuren effektivt.



OPPGAVE 1 – PROGRAMMER FIGURER

I denne oppgaven skal du også fylle inn svar i tabellene på arket. Du skal nå programmere Felix til å tegne:

- Kvadrat
- Rektangel
- Likesidet trekant

a. Beskriv egenskapene til de tre figurene i tabellen under og diskuter hva som er forskjellen og likhetene deres.

Kvadrat	Rektangel	Likesidet trekant

b. Hva er forskjellen og likheter mellom selve kodene dere har laget til de ulike figurene?

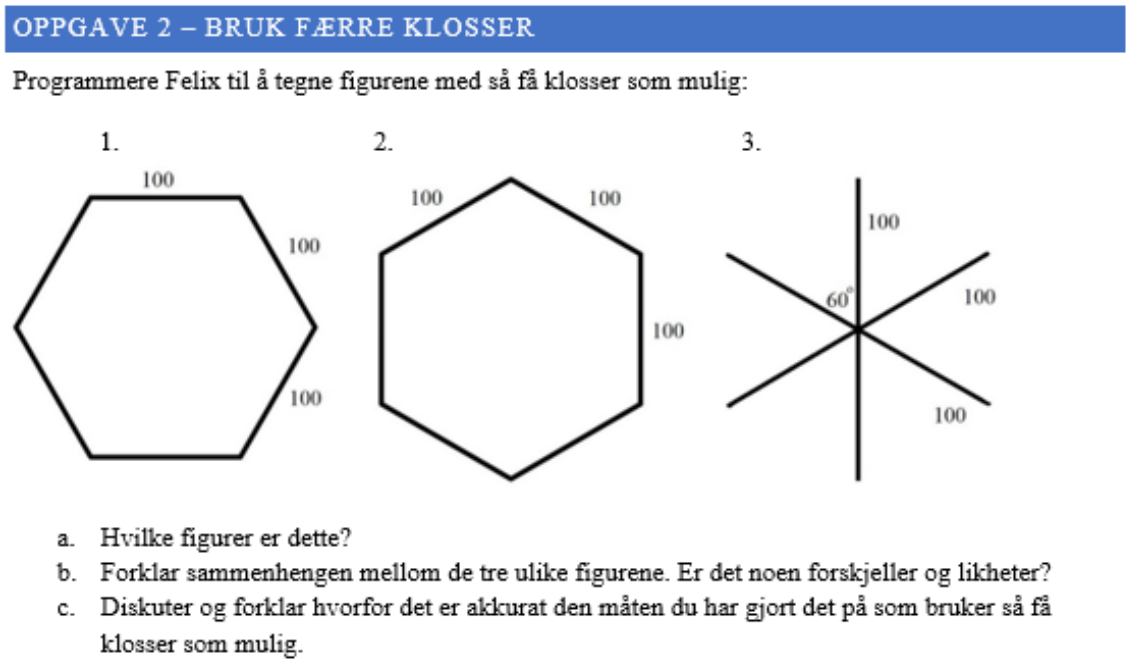
Forskjeller	Likheter

Figur 6. Oppgave 1 fra datainnsamlingen.

3.2.2 Oppgave 2

Formålet med oppgaven er at elevene skal utforske de geometriske figurene, for å finne egenskapene til en regulær sekskant, og sammenhenger mellom ulike rotasjoner av samme figur. Elevene må enten regne ut, eller prøve seg frem til hvor mange grader Felix må snu for å lage en regulær sekskant. På samme måte som for figurene i oppgave 1 er det ikke tilstrekkelig å vite hvordan en sekskant skal se ut, eller å kunne tegne den. For at man effektivt skal kunne lage en regulær sekskant i Scratch må man vite egenskapene til figuren.

I denne oppgaven blir elevene bedt om å bruke så få klosser som mulig. Det ble gjort for å få elevene til å se verdien av å bruke løkker med gjenta-klossen, når kodene blir lengre og mer omfattende. Den tredje figuren i oppgave 2 er ingen typisk geometrisk figur, det vil derfor være interessant å høre elevenes resonnementer rundt hvilken figur dette er, og hvilke egenskaper den har. En baktanke med utformingen av den tredje figuren var at dersom man setter figuren «inn i» den andre figuren, kan man oppdage at den regulære sekskanten består av seks likesidede trekanter.



Figur 7. Oppgave 2 fra datainnsamlingen.

3.2.3 Oppgave 3

Hovedmålet med oppgave 3 er at elevene skal utforske egenskapene til polygon med mange sider, som til slutt blir så lik en sirkel som mulig. Denne oppgaven legger til rette for å tenke abstrakt rundt sirkelen. Hvordan en sirkel defineres varierer innenfor hvilken del av matematikken man befinner seg. Innenfor programmeringsspråk som Logo og Scratch kan det være naturlig å kunne tenke at en sirkel inneholder et uendelig antall sider. Innenfor Euklidsk geometri defineres sirkelen på denne måten: «A circle may be described with any center and distance» (Burton, 2010, s. 146). Sirkelen har altså uendelig mange punkter i en gitt avstand (radius) fra et punkt (sentrum), og derfor ingen kanter. Når elevene resonnerer rundt sirkelen bør man derfor være klar over at det kan forekomme flere ulike forklaringer på antall sider til figuren.

Jeg valgte å designe oppgaven på en utforskende måte, der elevene får oppgitt koden til en regulær tolvkant, og oppfordres til å bruke den for å lage en figur som er så lik en sirkel som mulig. Siden oppgaven ber elevene om å ta utgangspunkt i en tolvkant der de kan endre antall sider, lengden på, og størrelsen på vinkelen, blir det lagt opp til en definisjon av sirkel med et uendelig antall sider. Siden dette egentlig blir en polygon med et uendelig antall sider, og ikke en sirkel, har jeg bevisst beskrevet figuren som «en figur som er så lik en sirkel som du klarer». Det er ikke mulig å lage en sirkel med framover- og roter-kommandoer i Scratch, men man kan lage en figur som er veldig nært en sirkel.

Figur 8. Oppgave 3 fra datainnsamlingen.

3.3 Datainnsamling

Jeg har i denne studien valgt å benytte meg av lyd- og skjermopptak for å samle inn datamateriale. Lydopptak er av elevsamtaler, og skjermopptak av elevenes dataskjerm under arbeid med programmering. Videre er det også tatt lydopptak under intervjuer etter programmeringsøkten. I dette delkapittelet beskriver jeg utvalget av informanter, konteksten for studien, og de ulike innsamlingsmetodene som har blitt brukt i undersøkelsene.

3.3.1 Utvalget av informanter

Jeg fikk tidlig i arbeidet kontakt med en lærer på 6. trinn som var interessert i programmering i matematikk, og som sa seg villig til å la meg gjennomføre en undersøkelse med sine elever. Jeg gjennomføre programmeringsøktene med totalt 52 elever på 6. trinn, elevene var fordelt på tre klasser. De ble plassert i par av deres egen lærer, utgangspunktet var å plassere elever som gikk godt overens sammen. Dette for å fremme en god diskusjon. Videre valgte jeg å se nærmere på to elevpar i hver av klassene, noe som totalt ga meg totalt seks par med elever. At elevene ble satt sammen i par var for å legge til rette for at det ble en fruktbar samtale, samt å gi mulighet for diskusjon mellom elevene. Tidligere forskning av Foerster (2016) viser at to elever som samarbeider med en datamaskin var fruktbart, og at Scratch bidro med et felles matematisk språk.

Min datainnsamling foregikk over to dobbelttimer i løpet av to uker, på samme skole, i samme klasse og med samme elever. Etter gjennomføringen av datainnsamlingen valgte jeg tilfeldig ut to av elevparene som danner empirien min. Den tilfeldige utvelgelsen ble gjort før jeg transkriberte lydopptakene. Årsaken til at jeg valgte å trekke to tilfeldige elevpar var at det ble for mye datamateriale å analysere, dersom jeg skulle gjort det med

alle seks parene. Siden kvalitative kasusstudier baserer seg på rikelig med informasjon om hvert kasus, var det mer hensiktsmessig å gå dypere i to av kasusene.

Totalt hadde 75% av elevene i undersøkelsen min arbeidet med programmering før introduksjonsøkten jeg hadde med dem. Elevene hadde enten prøvd programmering på skolen, eller gjennom Kodeklubben (www.kodeklubben.no), noe som ikke nødvendigvis trenger å være tilfellet for alle 6. klassinger i Norge. Av de tre som hadde prøvd programmering var det i Scratch og/eller Micro:Bit. Resultatet av at undersøkelsen min ble gjennomført på akkurat denne skolen, gjenspeiler muligens at elevene sannsynligvis har over gjennomsnittet erfaring innenfor programmering. Dersom undersøkelsen skulle blitt gjennomført med elever fra en annen skole, burde det muligens blitt brukt mer tid på den første økten.

3.3.2 Intervju

Samtaler har alltid vært en vesentlig del av menneskers livsverden, og språket fungerer som et medierende hjelpemiddel mellom mennesker (Postholm, 2005). Ved hjelp av språket har vi kunnet få kjennskap til den verden som vi lever i, svare på hverandres spørsmål og gi uttrykk for tanker. Ulike former for intervju utgjør viktige redskaper som en forsker kan benytte seg av. Dette for å utvikle forståelsen innenfor et praksisfelt (Postholm, 2005). Målet til det kvalitative forskningsintervjuet er å forsøke å forstå verden ut fra intervjuobjektets synspunkter, og på den måten knytte mening til opplevelser og å avdekke personens livsverden (Kvale & Brinkmann, 2009).

På forhånd av intervjuet hadde jeg tenkt ut noen temaer jeg ønsket at elevene skulle komme inn på, men jeg var også åpen for at de kunne ta opp andre temaer. Dette kaller Kvale og Brinkmann (2009) for et semistrukturert intervju. Formålet med intervjuet var å få mer utfyllende informasjon om elevenes forståelse av de geometriske figurene, samt hvordan de arbeidet med programmering. Intervjuet ble også en mulighet til å stille spørsmål dersom noe var uklart fra datainnsamlingen. Intervjuene ble gjennomført dagen etter datainnsamlingen, i de samme parene som under programmeringen. En utfordring med å gjennomføre et gruppeintervju kontra et intervju med én og én, vil være at elevene kan bli påvirket av hverandres svar. Jeg forsøkte å skape trygge rammer rundt intervjuet, ved å bruke de samme parene, og gjennomføre det på et grupperom med tilgang til elevenes klasserom.

3.3.3 Skjerm- og lydopptak

For å ta opptak av dataskjermen hadde jeg på forhånd lastet ned programmet Screen Recorder Lite. Dette lot meg ta opptak av alt som skjedde på skjermen, mens elevene programmerte. Hensikten med skjermopptaket var å supplere lydopptaket, slik at det også var mulig å se hva som skjedde på skjermen når elevene pratet sammen. Eller når de gjorde noe i programmet, uten å beskrive det muntlig. Lydopptaket ble transkribert av meg, og gjennomgått to ganger, for å sikre at transkripsjonen er så presis og korrekt som mulig. Jeg har valgt å ikke rette på elevenes grammatiske feil, men jeg har oversatt ulike dialekter til bokmål, da jeg mener at det ikke har noen betydning for å kunne svare på forskningsspørsmålene mine. Transkripsjonen inneholder skjermbilder fra skjermopptaket. Dette for å vise hvilke blokker med koder elevene brukte, og hvordan figurene de programmerte ser ut.

3.3.4 Pilotundersøkelse

For å kvalitetssikre begge undervisningsoppleggene var det hensiktsmessig å gjennomføre en pilotstudie på forhånd av datainnsamlingen. Jeg gjennomførte piloten i god tid før datainnsamlingen skulle finne sted, slik at jeg hadde tid til å gjøre endringer dersom det ble nødvendig. Piloten ble gjennomført med to elevpar på 6. trinn på en annen skole i samme kommune. Elevene hadde ikke brukt Scratch før, men den ene eleven var vant til å bruke «LEGO Mindstorms».

Jeg gjennomførte også en pilotundersøkelse med to masterstudenter som går 5.året med matematikkdidaktikk. Her ble begge undervisningsøktene gjennomført, og studentene arbeidet sammen i par med en datamaskin. Dette ble hovedsakelig gjort for å teste intervjuguiden, og slik at jeg fikk mulighet til å øve meg på rollen som intervjuer. Begge studentene har undervist på 6. trinn i matematikk, så det ble også en god mulighet til å diskutere vanskelighetsgrad og utforming av oppgavene.

Som følge av pilotundersøkelsen med elevparene i 6. klasse tilførte jeg en tabell, som elevene skulle føre inn svar på oppgave 1. Jeg valgte å tilføre tabellen for å skape en diskusjon mellom elevene. Oppgave 3 ble lagt til etter gjennomføringen av pilotundersøkelsen, fordi jeg ønsket å undersøke om elevene var i stand til å resonnere abstrakt om sirkelen. I etterkant av piloten med masterstudentene endret jeg på noen av formuleringene i intervjuet, samt på oppgavearket, slik at det ble enda tydeligere for elevene hva de skulle gjøre.

3.4 Metode for analyse av datamaterialet

Jeg har i studien min valgt å benytte meg av tematisk analyse. Tematisk analyse er en fleksibel metode for å identifisere og analysere mønstre i kvalitativt datamateriale (Braun & Clarke, 2006). Braun og Clarke (2006) skiller mellom induktiv og deduktiv tilnærming til analyse, men påpeker at analysen også kan ligge mellom de to ytterpunktene. Den induktive tilnærmingen tar utgangspunkt i datamaterialet, og lager temaer for analyse ut ifra respondentenes uttalelser. På den andre siden tar en deduktiv tilnærming utgangspunkt i en teoristyrt analyse, som baserer seg på forhåndsdefinerte kategorier (Braun & Clarke, 2006). I motsetning til den induktive, er den deduktive tilnærmingen mer drevet av forskerens interesse og forhåndsantagelser. Å gjøre analysen med utgangspunkt i teori gir fordelen med å kunne gå dypere inn i enkelte aspekter av dataene, men gir dermed en mindre detaljert beskrivelse av hele datasettet generelt. I denne studien benytter jeg meg av en deduktiv tilnærming til tematisk analyse. Jeg valgte å analysere datamaterialet ut ifra forhåndsbestemte kategorier fra det teoretiske rammeverket, fordi jeg allerede hadde jobbet ut i fra en problemstilling som la føringer for studien (Braun & Clarke, 2006). Analyseprosessen i denne studien tar utgangspunkt i det Braun og Clarke (2006) beskriver som seks faser av tematisk analyse:

- Fase 1: Å bli kjent med datamaterialet sitt.
- Fase 2: Koding av datamaterialet.
- Fase 3: Søking etter kategorier.
- Fase 4: Gjennomgang av kategoriene som er funnet.
- Fase 5: Definering og benevning av kategoriene.
- Fase 6: Skrive opp og fortelle om de ulike kategoriene.

Selv om fasene presenteres som en «steg-for-steg» guide fra 1-6, hevder Braun og Clarke (2006) at det ikke nødvendigvis må bli sett på som en lineær prosess, hvor hver fase må være gjennomført før man kan gå videre til neste:

> *"… analysis is not a linear process of simply moving from one phase to the next. Instead, it is more recursive process, where movement is back and forth as needed, throughout the phases" (Braun & Clarke, 2006, s. 83).*

Jeg har likevel valgt å benytte meg av alle fasene i denne rekkefølgen. Som en uerfaren forsker ønsker jeg å ha en konkret og systematisk tilnærming til analyse av datamaterialet.

Fase 1: Å bli kjent med datamaterialet

Den første fasen handler om å gjøre seg kjent med datamaterialet. Den første bearbeidelsen av datamaterialet var å transkribere, for så å lese gjennom transkripsjonen. Deretter ble datamaterialet gjennomgått på nytt, for å notere ned helhetsinntrykk og potensielle tema og mønstre. Den andre gangen jeg leste gjennom transkripsjonen markerte jeg elevuttalelser som handlet om selve programmeringen i en farge, og uttalelser om geometri i en annen farge. Denne prosessen bidro til at jeg justerte problemstillingen min, og tilførte forskningsspørsmålet, som handler om resonnering i arbeid med geometriske figurer.

Fase 2: Koding av datamaterialet

Den andre fasen går ut på å kode datamaterialet. Jeg benyttet meg av ulike fargekoder, for å få en bedre oversikt over hvilken informasjon som gjentok seg på tvers av transkripsjonen av skjerm- og lydopptakene, samt i intervjuene. Lehrer (1998) sine kategorier for resonnementer, og Fischbein (1993) sin beskrivelse av figurale begrep, dannet grunnlaget for å undersøke det første forskningsspørsmålet. Kodene jeg benyttet i dette arbeidet var likheter, størrelser, vinkel, retning, omforming, telling, egenskaper og figurale begrep. For å undersøke det andre forskningsspørsmålet, og elevenes arbeidsmåter innenfor programmering, benyttes Shute et al. (2017) sitt rammeverk for algoritmisk tenkning, for å kode datamaterialet. Kodene jeg benyttet var dekomponering, abstraksjon, algoritmer og feilsøking.

Fase 3: Søking etter kategorier

Den tredje fasen handler om å søke etter kategorier i datamaterialet. Her prøver forskeren å få et mer helhetlig blikk, og vurderer hvordan forskjellige koder kan kombineres for å skape hovedtemaer eller kategorier. I følge Braun og Clarke (2006) begynner man her med litt lett analysering. Et eksempel er at når elevene beskriver geometriske figurer basert på kodene likheter, størrelser, vinkel, retning, omforming eller telling, blir det definert som et resonnement innenfor kategorien visuelle kjennetegn.

Fase 4: Gjennomgang av kategoriene som er funnet

Den fjerde fasen er gjennomgang av kategoriene som er funnet. Gjennomgangen skal bidra til at forskeren danner seg et bilde av kodingen. Det innebærer å vurdere om arbeidet i fase 1 og fase 2 har blitt utført på en god måte, og om dette har en sammenheng (Braun & Clarke, 2006). Fasen deles inn i to nivåer. Nivå 1 handler om å se nærmere på individuelle temaer. Nivå 2 gjelder å se nærmere på hele datasettet. Forskeren vil da kunne se validiteten til hvert tema i forhold til datasettet, og om temaene nøyaktig reflekterer meningene i resultatet som helhet.

Fase 5: Definering og benevning av kategoriene

Fase 5 går ut på å definere og gi navn til hver av kategoriene, som representerer resultatene. Dette var allerede gjort, da jeg benyttet meg av en deduktiv tematisk analyse, med utgangspunkt i teori. Kategorien *visuelle kjennetegn* er slått sammen av flere temaer, og defineres i analysekapitlet. Der beskrives det også hva som skiller kategorien fra *egenskaper*. Braun & Clarke (2013) hevder at i denne fasen skal forskeren finne essensen av temaene man ønsker å presentere i resultatdelen.

Fase 6: Skrive opp og fortelle om de ulike kategoriene

Siste fase av analyseprosessen er å skrive opp, og fortelle om de ulike kategoriene jeg har funnet i datamaterialet. På samme måte som i fase 5 var temaene basert på teori, og jeg har beskrevet hver kategori i analysen, med utgangspunkt i det teoretiske rammeverket.

3.5 Kvaliteten til studien

Innenfor kvantitativ forskning er kvantifisering og målbarhet noe av det viktigste, og subjektive data må klassifiseres etter objektive kriterier. Hensikten er å identifisere sammenhenger der gyldighetsområdet er så omfattende som mulig. Kvantitative studier støtter seg til statistiske generaliseringer. Det vil derfor være relativt enkelt å gjennomføre den nøyaktig samme studien flere ganger (Yin, 2009). Kvalitativ forskning vil derimot alltid være påvirket av forskerens forståelse og bakgrunn (Nilssen, 2012). Innenfor kvalitativ forskning stilles det derfor krav om at studien skal være troverdig og pålitelig. Siden forskning i stor grad påvirkes av konteksten, vil det være utfordrende å gjennomføre den samme undersøkelsen på nøyaktig samme måte flere ganger. Gjennom å være nøye i beskrivelsen av hvordan jeg har gått fram i undersøkelsen, har jeg som mål å overbevise leseren om at jeg har foretatt fornuftige valg. Målet mitt som en kvalitativ forsker er å forsikre leseren om at bildet som blir gitt i denne studien ikke er feilaktig. At det ikke er en forvrenging av de faktiske forhold, samt å unngå misforståelser (Nilssen, 2012). Det ligger et stort ansvar på forskeren, som hele tiden må vurdere hvilke innsamlingsmetoder som er hensiktsmessige og etisk forsvarlige, både når det gjelder forskningsspørsmål og informanter.

3.5.1 Validitet, reliabilitet og bekreftbarhet

Validiteten til forskning omhandler gyldigheten i undersøkelsen, og om svarene vi finner faktisk svarer på spørsmålene vi stilte innledningsvis i forskningen. Postholm (2005) beskriver at validitet er mer avhengig av mangfoldet i informasjonen, og forskerens evne til å analysere, enn utvalgets størrelse. Utvalget i min studie mener jeg ble tilstrekkelig begrenset, da jeg reduserte antall informanter fra seks elevpar til to. Guba (1981) presenterer fire kriterier for å sikre kvaliteten i en kvalitativ studie. De fire kriteriene er kredibilitet (indre validitet), overførbarhet (ytre validitet), reliabilitet og bekreftbarhet. Validitet deles ofte inn i indre og ytre validitet.

Kredibilitet handler om at studien fremstår som sannsynlig eller tillitvekkende for leseren (Guba, 1981). I min studie benyttet jeg flere datainnsamlingsmetoder, som er et redskap for å sikre studiens kredibilitet (Guba, 1981). Dersom ulike innsamlingsmetoder ser ut til å vise samme mønster, vil det være med på å underbygge resultatet, og følgelig styrke sikkerheten om et gyldig resultat.

Dersom studien har en høy overførbarhet kan den anvendes i andre kontekster, med andre deltakere (Guba, 1981). Resultatene fra studien kan generaliseres, og regnes for å gjelde en større mengde data enn det studien undersøkte, dersom den har høy overførbarhet. Jeg har i beskrivelsen av informantene (kap. 3.3.1) beskrevet elevenes erfaring innenfor programmering som mest sannsynlig over gjennomsnittet for 6. klassinger i Norge. Jeg mener den ytre validiteten til undersøkelsen vil styrkes ytterligerligere, når kompetansemål innenfor programmering i fagfornyelsen (Utdanningsdirektoratet, 2019) integreres for alle elever på 6. trinn, slik at utgangspunktet for elevene utjevnes.

Reliabilitet handler om at de samme resultatene ville blitt oppnådd dersom studien hadde blitt gjort på nytt med de samme (eller lignende) deltakerne, og i samme (eller lignende) kontekst (Guba, 1981). Funnene i en kvalitativ studie er avhengige av den konteksten forskningen fant sted (Nilssen, 2012). Funnene i min studie kunne derfor vært annerledes, dersom jeg hadde gjennomført den samme undersøkelsen med andre 6. klassinger. Men det er viktig at funnene mine gir mening i lys av det innsamlede materialet (Nilssen, 2012). Jeg har i denne oppgaven beskrevet hvordan jeg gjennomførte undersøkelsen, og hvordan jeg har analysert datamaterialet. Gjennomføringen er derfor tydelig for leseren. Det er viktig å være grundig, nøyaktig og ærlig i gjennomføringen av studien i kvalitativ forskning, for å sikre reliabilitet.

Det siste kriteriet som beskrives av Guba (1981) er bekreftbarhet, noe som handler om i hvilken grad forskeren evner å være objektiv i undersøkelsen. Dette beskrives som i hvilken grad man kan stole på at funnene i undersøkelsen kommer fra deltakerne. Og at de ikke er påvirket av faktorer som motivasjon, interesse og lignende fra forskeren (Guba, 1981). Kritikere til casestudier hevder at forskeren kan være for subjektiv i sin tolkning av datamaterialet (Yin, 2009). For å bidra til en større objektivitet i analysen av datamaterialet ble det analysert gjennom eksisterende teori.

3.6 Etiske overveielser

Etiske betraktninger omhandler også kvaliteten på forskningen, og er et viktig tema innenfor enhver undersøkelse som skal gjennomføres. Når det gjennomføres undersøkelser som involverer barn er det spesielt viktig å ta hensyn til deres opplevelse. Kvalitativ forskning innebærer å utforske menneskelige prosesser eller problemer i deres naturlige setting. Det vil derfor være et nært forhold mellom forsker og forskningsdeltakere (Postholm, 2005). Forskeren må foreta etiske avgjørelser både før, i løpet av og etter samhandlingen med informantene. Gjennomtenkte etiske betraktninger bør gjennomsyre forskerens handlinger gjennom hele prosessen. Forskeren blir nødt til å løse etiske dilemmaer, i forhold til situasjonen han eller hun er i (Postholm, 2005).

I forkant av datainnsamlingen ble undersøkelsen min godkjent av Norsk senter for forskningsdata (NSD). Dette måtte gjøres på bakgrunn av at datamaterialet kunne inneholde sensitive personopplysninger. Det ble laget et samtykkeskjema til elevenes foresatte, fordi elevene var under 18 år (se vedlegg 4). Der ble forskningsprosjektet mitt beskrevet, formålet til studien, hvordan jeg skulle samle inn datamaterialet, og hva som ville skje med personopplysninger etter prosjektet avsluttes. I skjemaet kunne foreldrene enten samtykke i at barna deres deltok, eller ikke samtykke. Kvale og Brinkmann (2009) beskriver dette som et informert samtykke.

Under gjennomføringen av datainnsamlingen ble elevparene som brukes i studien tatt med ut av klasserommet. Dette for at jeg skulle kunne ta lydopptak under arbeidet, uten at elever som ikke hadde godkjent samtykke ble med. For å verne om elevenes personopplysninger ble det brukt godkjente lydopptakere, og lydfilene ble slettet etter at de var blitt transkribert. De anonymiserte transkripsjonene var kun tilgjengelig for meg.

4 Analyse

Målet med dette kapittelet er å gi innsikt i hvordan elevene resonnerer, samt deres arbeidsmåter innenfor programmeringen, for å kunne svare på problemstillingen min. Denne ble presentert tidligere i oppgaven, og er som følger: *«Hva kjennetegner et utvalg 6. klassingers matematiske resonnering, og deres arbeidsmåter i programmering av geometriske figurer?».* Denne problemstillingen skal jeg undersøke ved hjelp av forskningsspørsmålene:

1. *Hvilken type matematiske resonnementer bruker elevene i møte med programmeringsoppgaver i geometri?*
2. *Hvilke arbeidsmåter innenfor algoritmisk tenkning benytter elevene i løsningen av programmeringsoppgaver i matematikk?*

På bakgrunn av at det er to forskningsspørsmål, deles også analysekapitlet i to deler. Delkapittel 4.1 presenterer elevenes matematiske resonnering. Delkapittel 4.2 presenterer elevenes arbeidsmåter. Datamaterialet ble analysert gjennom koding og kategorisering, ved bruk av en deduktiv tematisk analyse.

Jeg omtaler meg selv som F i transkripsjonen, gjennom hele analysen. Dersom elevene gjør noe som ikke kommer frem i transkripsjonen, men som har betydning for datamaterialet, beskrives det av meg i en parentes. For eksempel betyr «(avbryter)» at eleven avbryter den andre.

4.1 Resonnering

I denne delen av analysen presenteres skjermbilder av de geometriske figurene og sitater fra transkripsjonen av lydopptaket, som handler om matematiske resonnementer innenfor kategoriene. Interessante ytringer fra det semistrukturerte intervjuet brukes, dette for å supplere resultatet. Den tematiske analysen førte til tre hovedkategorier: *visuelle kjennetegn, egenskaper* og *abstrakt resonnement*. Videre presenteres kategoriene innenfor resonnering for hvert av elevparene adskilt.

4.1.1 Elevpar 1 – Ola og Per

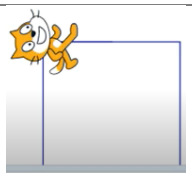
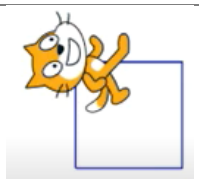
Elevpar 1 består av to gutter. Begge hadde prøvd programmering før de to øktene de gjennomførte. Elevene omtales videre i studien med de fiktive navnene Ola og Per. Per prøvde programmering på skolen da han gikk i 4. klasse, mens Ola deltar på et kurs i regi av Kodeklubben en gang i uken, der det arbeides med Micro:Bit. Ingen av elevene hadde tidligere brukt programmeringen for å arbeide med matematikk.

Visuelle kjennetegn

Kategorien visuelle kjennetegn tar utgangspunkt i flere av kategoriene innenfor geometrisk resonnering, som beskrives av Lehrer (1998), samt van Hiele-nivå 0. Sistnevnte beskrives som visualisering av van Hiele (1959/1984). I datamaterialet identifiseres kodene *likhet, størrelse, retning* og *telling*. Det kan være vanskelig å skille mellom kategoriene *visuelle kjennetegn* og *egenskaper*. Det som i denne studien skiller om resonnementet tilhører

kategorien *visuelle kjennetegn* eller *egenskaper,* er at det ikke benyttes egenskaper som er betydelige for figurens definisjon.

Det første eksemplet fra datamaterialet er et utdrag fra løsningen av oppgave 1. Utdraget viser til koden *likhet* når elevparet arbeidet med å lage var et kvadrat. De to elevene har sammen klart å lage et program som tegner et kvadrat, men siden hele kvadratet ikke vises på skjermen ser ikke elevene seg ferdig med oppgaven.

Linje nr.	Hvem	Sitat	Skjermbilde
65	Per	Ja, vi fikk det til!	
68	Ola	Men det her er feil, vet du hvorfor?	
69	Per	Hvorfor?	
70	Ola	Man ser ikke hele kvadraten	
74	Ola	Istedenfor 200 steg så putter vi bare 100, da kan man se kvadraten, fordi det er jo viktig at man ser kvadraten så vi ikke tegnet bare et tak	
75	Per	Eller kanskje en pipe, haha	
		(endrer antall steg fra 200 til 100)	
77	Ola	Å, jeg gjorde feil, nei faktisk ikke	

Ola mener at det er feil fordi «man ser ikke hele kvadraten». Utdrag 74 og 75 fra dialogen mellom elevene viser at de diskuterer hva annet det kan være, når man ikke ser hele kvadratet. Elevene uttrykker at figuren ligner på virkelige objekter fra deres hverdag, som for eksempel et tak eller en pipe. Resonnementet elevene gjør kategoriseres som *likhet*, da figuren beskrives som andre virkelige objekter elevene har kjennskap til.

Videre viser dialogen et eksempel på hva som kan være avgjørende for at et resonnement kategoriseres som *visuelle kjennetegn,* og ikke kategorien *egenskap*. Elevene får spørsmål om å beskrive hvorfor det er et kvadrat de har laget. Ola sier i utdrag 90 at «alle sidene er likesidet, alle sidene er like lange».

87	F	Så dere får den til å tegne?
88	Per	En firkant, et kvadrat!
89	F	Hvorfor er det der et kvadrat da?
90	Ola	Det er på grunn av at alle sidene er likesidet, alle sidene er like lange

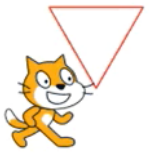
En beskrivelse som bare baserer seg på lengden til sidene har blitt kategorisert som koden *størrelse.* Det er et resonnement som bruker visuelle kjennetegn. Det er en forklaring som ikke er entydig for kvadratet, og tilfredsstiller derfor ikke viktige egenskaper for figuren.

I neste eksempel viser dialogen at begge elevene resonnerer med utgangspunkt i figurens *størrelse*.

96	Ola	Rektangel, men hvordan skal vi lage rektangel, er det en sånn lang firkant …
97	Per	… det er en firkant, en kvadrat bare at to av sidene er lengre enn de øverste, de to på sidene

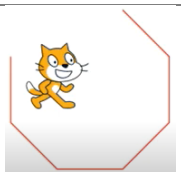
Ytringen til Ola i utdrag 96 viser til når han skal beskrive et rektangel. Eleven beskriver figuren med utgangspunkt i *størrelsen,* siden rektanglet beskrives som en lang firkant. Per forklarer i utdrag 97 at det er et kvadrat, der to av sidene er lengre enn de to andre. Dette kategoriseres også som et resonnement med utgangspunkt i *størrelse*. Det er en beskrivelse som gjøres med bakgrunn i lengden på sidene. Elevene påpeker også at det er de to på sidene som må være lengre enn de øverste. *Størrelse* kategoriseres som et *visuelt kjennetegn.* Dette fordi det er et kjennetegn man kan se og/eller måle ut ifra figuren.

Neste eksempel er fra når elevene skal lage en likesidet trekant. Ola mener figuren de lager er opp ned.

209	Per	Ja! Vi klarte det 
212	F	Hvor mange grader var det da?
213	Per	120!
219	F	Ja, det ser ut som en likesidet trekant det der?
220	Ola	Ja, opp ned
221	F	Opp ned ja? Hvilken vei skal den egentlig være da?
224	Ola	Den flate siden skal være nederst

I utdrag 220 og 224 fra transkripsjonen beskriver Ola at trekanten de har programmert er «opp ned» fordi «den flate siden skal være nederst». Dette kategoriseres som et resonnement med utgangspunkt i *retning.* Det er rotasjonen til figuren som avgjør om den er riktig vei, eller det eleven beskriver som «opp ned».

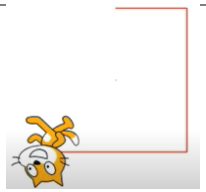
I oppgave 2 skal elevene lage en regulær sekskant. Utdraget fra transkripsjonen viser at Ola i utdrag 266 bruker *telling* av sidene for å identifisere hvilken figur det er.

264	Ola	Ah, ah, ah yey! Nei!	
265	Per	Nei!	
266	Ola	Det er uansett ikke en sekskant, 1, 2, 3, 4, 5, 6, da er det en åttekant	
267	Ola	Da må vi ta større grad. Vi tar 60!	
268	Per	Ja, fordi det er halvparten av det vi hadde ista	
269	Ola	Er det?	
270	Per	Ja, av 120	
271	Ola	Nei, vi tok 45	
272	Per	Ja, men når vi lagde likesidet trekant	
		(endrer fra snu 45 grader til 60 grader)	
274	Ola og Per	Yey! Vi klarte det! (roper på F)	

Dette eksempelet blir kategorisert som koden *telling.* Det som avgjør Ola sitt resonnement, er antall sider figuren har. Selv om tellingen stoppet på 6 ser han at det blir en åttekant dersom man setter inn to sider til. Eleven foreslår deretter å rotere 60 grader. Per beskriver i utdrag 268 og 272 at det er halvparten av rotasjonen de gjorde for å lage likesidet trekant. Det kan se ut til at eleven har sett en sammenheng med antall grader Felix vender og hvilken mangekant man får. Dette kan være en begynnende forståelse for sammenhengen mellom størrelsen man roterer, og antall sider i regulære mangekanter.

Egenskaper

Elevene tar i bruk *egenskaper* i sine resonnementer når de bruker egenskaper som er betydelige for den geometriske figuren sin definisjon. Egenskaper kan være blant annet vinkler, sidekanter og parallelle linjer. Matematiske resonnementer med utgangspunkt i egenskapene til figurene ble viktige i elevparets arbeid. For å underbygge dette vises det først til et eksempel fra dialogen under programmeringen av et rektangel.

134	Ola	Okei da går den 100 steg, så snur den 90 grader, så går den 200 steg, så snues den 90 grader. Herregud det her ble veldig fargerikt, jeg liker ikke farger	
135	Ola	Okei, vi bare sjekker hvordan det ble	
136	Ola	Åja, jeg endret ikke på 200 steg	
137	Per	Ånei, den der må vi ha til 100, og så må vi ha snu 90 grader, og så 200 steg igjen	

140	Per	Men nå må vi ha snu 90 grader, så gå 200 steg igjen
150	Per	Men sjekk hvordan det er nå da, vi sjekker om det er en rektangel nå
151	Ola	Ja, se her
152	Per	Jeg tipper det er det, nesten 100%
153	Ola	Da trykker vi bare her, vi er ferdige!
160	Per	Det der ble ganske bra da! Vi er ferdige! (roper på F) 

Elevene inkluderer både parvise like lange sider, og vinkler på 90 grader, noe jeg mener skiller resonnementene fra kategorien *visuelle kjennetegn*. Utsagn 134, 136, 137 og 140 viser hvordan elevparet i samarbeid resonnerer seg frem til oppbygningen av et rektangel. Dialogen plasseres innenfor kategorien *egenskaper.* Dette fordi parvise sider med like lengder og vinkler på 90 grader er avgjørende egenskaper for definisjonen til figuren. I motsetning til beskrivelsen av rektangel i forrige delkapittel som «en lang firkant» (utsagn 96), og som «en kvadrat bare at to av sidene er lengre enn de øverste» (utsagn 97), bruker elevene både lengden på sidene og vinklene når figuren lages i Scratch. Noe som kan tyde på at Scratch bidrar til å legge vekt på egenskapene til figuren når den programmeres.

På oppgave 3 prøver elevene å lage en sirkel. Dette gjør de ved å lage sidelengder på ett skritt, og snu en grad. Problemet angripes med prøving og feiling. Det resulterer i at elevene oppdager at instruksjonen, å gå ett skritt og snu en grad, må gjentas 360 ganger.

621	Ola	(gjenta) 200 ganger var ikke nok? 
624	Per	… jeg tror 300 kanskje er litt for mye
625	Per	What? 
626	Per	Okei, 350?
629	Per	Åååå wow. Det der er det værste 
630	Ola	360?

632	Ola	Vi klarte det perfekt!	
633	Per	Ja, vi klarte det!	
634	F	Hvor mange ganger måtte han gjenta det?	
635	Per	360, vi prøvde 350	
636	Ola	Åja! Fordi det er 360 grader	
638	F	I hva da?	
639	Per	I en hel …	
640	Ola	… sirkel	

Utsagn 636 og 640 viser at Ola uttrykker at instruksjonen må gjentas 360 ganger. «Fordi det er 360 grader» i en sirkel. Resonnementet baserer seg på vinkelsummen, som er en viktig egenskap for sirkelen. Utdraget over viser til elevenes arbeid med å lage en sirkel, ved å rotere en grad og gå ett steg. Elevene endrer antallet ganger det gjentas, til det Ola beskriver som en sirkel, etter å ha gjentatt 360 ganger. Det er viktig å påpeke at dette egentlig er en 360-kant. Det diskuteres nærmere i neste kategori som er *abstrakt resonnement*.

Abstrakt resonnement

Fischbein (1993) beskriver at figurale begrep skal inkludere spatiale egenskaper, som også besitter begrepsmessige kvaliteter. Kategorien *abstrakt resonnering* er basert på koden *figurale begrep*. Abstrakte resonnement beskriver elevenes resonnementer på et mer abstrakt nivå enn *egenskaper* gjør. Det identifiseres i elevenes resonnementer om figurer som ligner på en sirkel.

632	Ola	Vi klarte det perfekt!	
633	Per	Ja, vi klarte det!	
641	F	Så, er den der sirkelen mer perfekt enn den dere tegnet i stad?	
642	Per	Ja	
643	Ola	Ja	
644	F	Går det an å tegne en helt perfekt sirkel i Scratch?	
645	Per	Nei	
646	Ola	Dersom du tar 0.5, vi må bare doble det her til 720!	
648	Ola	Nei, det går ikke an å lage en perfekt sirkel	
649	Per	Den der var helt perfekt jo	

650	F	Er den der mer perfekt?
651	Ola og Per	Ja!
652	F	Er den mest perfekt?
653	Per	Ja! Eller det går an å ta 0.1
654	Ola	0.1!
657	Ola	Nei, det går ikke an å få en perfekt sirkel
659	Per	Hvor lang tid tar 0.01 da?
660	F	Det kommer til å ta veldig lang tid ja. Blir det der en perfekt sirkel da?
661	Per	Ja. Nei! Det går an å ta 0.01

Ola uttrykker i utdrag 632 at «Vi klarte det perfekt!», mens han senere, i 648, forklarer at «Nei, det går ikke an å lage en perfekt sirkel». Videre, i 657, at «det går ikke an å få en perfekt sirkel». Eleven ser ut til å ha oppdaget at man alltid kan gjøre rotasjonen og hver side mindre og at det derfor aldri blir en perfekt sirkel. Eleven uttrykker i 646 at man kan endre fra 1 til 0,5 dersom man endrer antallet ganger det gjentas fra 360 til 720. Elevene endret både rotasjonen og antall steg til 0,5 og gjenta til 720, for å få figuren i utdrag 649.

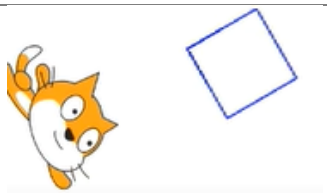
4.1.2 Elevpar 2 – Ane og Mia

Elevpar 2 består av to jenter med de fiktive navnene Ane og Mia. Ane har prøvd programmering noen dager på NTNU for flere år siden. Det var ikke med Scratch, og hun har ikke programmert etter det. Mia kan derimot ikke huske å ha prøvd programmering før.

Visuelle kjennetegn

I datamaterialet til Ane og Mia forekommer kodene *likhet, størrelse* og *telling* innenfor kategorien *visuelle kjennetegn*. Det følgende eksemplet fra datamaterialet viser til flere eksempler av koden *likhet*.

145	Mia	Det der blir jo en trekant	
147			
148	Ane	(flytter bort Felix) Nei! Det ble en firkant	
149	Mia	Det her er vanskelig	
150	F	Er det en firkant?	
151	Mia	Det må jo telles som et kvadrat	
152	F	Fordi? Skal det telles som et kvadrat?	
153	Ane	Nei	
154	F	Hvorfor ikke?	

155	Mia	Vi må prøve på nytt. Først må vi slette
		(elevene endrer på koden)
160	Ane	Jaaaa! 
161	F	Hvilken figur ble det der da?
162	Ane	Firkant!
163	Mia	Et kvadrat
164	F	Et kvadrat, hvorfor er det kvadrat?
165	Ane	Fordi den er firkantet

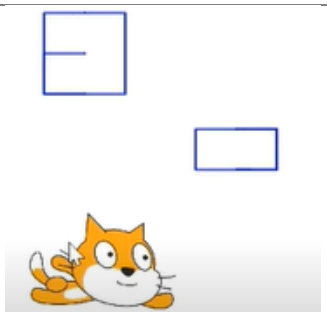
Ane beskriver i utsagn 148 at figuren ble en firkant, Mia beskriver videre, i 151, at figuren «må telles som et kvadrat». At Mia mener figuren må telles som et kvadrat kan tolkes som at hun egentlig ser at det ikke er et fullstendig kvadrat, men at figuren ligner. Det er viktig å presisere at begge resonnementene er feil. Dette fordi figuren ikke kan defineres som en mangekant, siden figuren ikke er lukket. Ane mener derimot at det ikke kan telles som et kvadrat. Årsaken til at figuren i utdrag 160 er et kvadrat er ifølge Ane, i 165, «fordi den er firkantet». Resonnementene i både utsagn 148, 151 og 165 kategoriseres som koden *likhet,* siden resonnementene kun baserer seg på utseendet til figurene.

I neste eksempel forklarer elevene hva de skal programmere i oppgave 2. Utsagn 321 viser at Ane bruker *telling* for å identifisere hvilken figur den andre figuren i oppgaven er.

316	Mia	Ehm, sekskant?
317	F	Sekskant ja, hva med den her?
318	Ane	Femkant? Åttekant?
320	Mia	En sekskant?
321	Ane	1,2,3,4,5,6, det er en sekskant

Utdrag 321 fra dialogen over har fått koden *telling.* Det som er avgjørende for Ane sitt resonnement er antallet hjørner/sider figuren har.

For å beskrive forskjellen mellom rektanglet og kvadratet legger elevene vekt på størrelsen til figurene i resonnementene. Men på ulike måter. Utsagn kodes med *størrelse* dersom det fokuseres på størrelsen til figuren. Størrelsen er noe man kan se og/eller måle.

228	Mia	Hva skjer der? Vent, nå har vi et kvadrat	
237	F	Hva er forskjellene og likhetene mellom de to figurene der da?	
238	Mia	Forskjellene er …	
239	Ane	… den der er dobbelt så stor som den …	
240	Mia	… hysj! Et kvadrat er at den er, nei et rektangel er at den er lang og kort	
241	F	Okei, lang og kort?	
241	Mia	Ja, jeg vet ikke hvordan man skal forklare det	
243	F	Ja, hva er forskjellen på sidene?	
244	Mia	De er korte, også er det en lang. Jeg klarer ikke å forklare	

Utsagn 239 viser at Ane beskriver kvadratet som dobbelt så stort som rektanglet, mens Mia, ut ifra utsagn 240 og 244, forsøker seg på en generell beskrivelse. Ane sitt resonnement tar utgangspunkt i koden *størrelse* når hun uttrykker at «den er dobbelt så stor som den». Noe som i dette tilfellet er riktig, siden kvadratet består av to like rektangler. Resonnementet Mia uttrykker i 240 og 244 er ufullstendige beskrivelser, med utgangspunkt i størrelsen på sidene til rektanglet. Det skiller resonnementet fra å bli kategorisert som *egenskaper*.

Egenskaper

Under programmeringen vises det ikke til et eneste eksempel fra kategorien *egenskaper*. Under intervjuet fremkommer det imidlertid resonnementer som har utgangspunkt i egenskapene til figurene. Videre presenteres derfor aktuelle utdrag fra intervjuet.

Det første eksemplet fra intervjuet viser til to resonnementer kodet med *egenskaper*. Elevene får spørsmål om hvilken figur man får, dersom man tegner den tredje figuren inn i den andre figuren, i oppgave 2.

192 Ane: Det er trekanter

193 F: Ja, det er trekanter

194 Ane: Er det likesidete trekanter? Eller er ikke alle det? Eller er det. Det er en, eller det er flere likesidete trekanter

195 F: Det er flere likesidete trekanter ja, hvor mange?

196 Ane: Seks!

197 F: Så da er faktisk sekskanten bygd opp av seks likesidete trekanter. Hvor stor er vinklene i de likesidede trekantene?

198 Ane: 60

199 F: Er du enig?

200 Mia: Jeg er usikker

201 F: Det er 60 ja …

202 Ane: … (avbryter) til sammen er det 180

Ane beskriver i utdrag 192 at det er trekanter inne i figuren. I utsagn 194 ser det ut til at hun resonnerer seg frem til at det er flere likesidede trekanter. Videre svarer Ane 60 når hun får spørsmål om hvor stor vinklene i de likesidede trekantene er. Hun legger til at det til sammen er 180. Utsagnene som beskriver at vinkelstørrelsen er 60 (grader), og at det (vinkelsummen) til sammen er 180 (grader), kategoriseres som *egenskaper.* Resonnementet tar utgangspunkt i viktige egenskaper for definisjonen til en likesidet trekant.

Det neste eksemplet viser til elevparets beskrivelse av et kvadrat og hva som skiller kvadratet fra rektanglet. Flere av resonnementene til Ane regnes samlet som et resonnement som benytter seg av *egenskaper*.

89 Ane: Kvadrat er et annet ord for firkant. Men den har fire like sider

90 F: Ja, den har fire like sider. Er det noe mere som bestemmer at det er et kvadrat?

91 Mia: Dersom man bare sier firkant kan det være masse forskjellig

92 F: Har du noen eksempler på hva det kan være dersom man bare sier firkant?

93 Mia: Det kan være et rektangel

…

107 Ane: Det må også være fire like kanter

108 F: Fire like kanter? Tenker du da på hjørnene du pekte på?

109 Ane: Ja

110 F: Vet dere hvor stor vinklene må være?

111 Ane: 90 grader

112 F: Og det må det være på hvor mange av hjørnene?

113 Ane: Alle fire

Ane beskriver samlet i løpet av utdragene 89, 107 og 111 at kvadratet har fire like sider, med fire like hjørner, der alle hjørnene er 90 grader. Det samlede resonnementet til eleven beskriver minimumsdefinisjonen til kvadratet. Selv om ingen av utsagnene hver for seg

kategoriseres med koden *egenskaper*, mener jeg det er relevant å se på utsagnene under samme spørsmål i intervjuet som et resonnement bestående av flere oppdelte argumenter.

Abstrakt resonnement

Oppgave 3 var ment for å fremme abstrakte resonnementer. Elevene skal endre koden for en tolvkant, slik at den tegner en figur som ligner på en sirkel. Eksemplet under viser hvordan Ane beskriver tolvkanten som en «runding», og at den følgelig har ingen sider.

422	Ane	Det blir en runding
423	F	Hvor mange sider har den?
424	Ane	0
425	F	0? prøv å tegne den da
426	Ane	Ja, men det skal være en runding, den har jo ingen sider
427	F	Ja, men når det ikke ble en runding nå, hvor mange sider har den?
428	Ane	8 sikkert
429	F	Prøv å tell da
430	Ane	1,2,3,4,5,6,7,8,9,10,11, 12-kant

Som tidligere eksempler har vist ser det ut til å være en utfordring for elevparet å ikke la det visuelle ta overhånd. Ane beskriver tolvkanten som «det blir en runding», men er likevel i stand til å telle at den har tolv sider. Det ble derfor interessant å spørre elevene om hvilke egenskaper en sirkel har, samt om de klarte å lage den i Scratch.

219 F: Hvilke egenskaper har en sirkel da?

220 Mia: Den har ingen kanter …

221 Ane: … (avbryter) og den er rund! En ball

222 F: Klarte dere å lage en sirkel i Scratch?

223 Mia og Ane: Ja!

224 F: hvordan Klarte dere det?

225 Mia: Først så fikk vi feil …

226 Ane: …vi laget den, så ble det en sekskant (elevene laget egentlig en tolvkant), så endret vi tallene til det ble en sirkel

227 F: Hvor mange kanter ble det?

228 Ane: Den hadde seks, så ble det null!

…

235 Mia: Når vi fikk tallet høyere ville den bli rund

Mia og Ane ser ut til å være enige om at en sirkel ikke har noen sider, ut ifra intervjuet og programmeringen. Samtidig mener begge elevene, utdrag 223, at man kan lage en sirkel

i Scratch. Ane beskriver i 226 at «så endret vi tallene til det ble en sirkel». Mia beskriver i 235 at «når vi fikk tallet høyere ville den bli rund». Likevel er begge elevene, sett ut ifra utdrag 220 og 228, sikker på at en sirkel ikke har noen kanter. Det kan derfor se ut som at det er en konflikt mellom den visuelle fremstillingen, og den begrepsmessige forståelsen av sirkelen. Det kommer ikke tydelig frem hvordan elevene mener man kan lage en sirkel i Scratch.

4.2 Algoritmisk tenkning

I denne delen av analysen presenteres utdrag fra programmeringen, skjermbilder av kodene, samt utdrag fra intervjuet, som handler om elevenes arbeidsmåter. Med bakgrunn i komponentene beskrevet av Shute et al. (2017) ble de fire kategoriene etter den tematiske analysen som følger; *abstraksjon, problemnedbrytning, algoritmer* og *feilsøking*. Videre presenteres kategoriene, adskilt for hvert av elevparene.

4.2.1 Elevpar 1 – Ola og Per

I datamaterialet til Ola og Per identifiseres kategoriene *abstraksjon, problemnedbrytning, algoritmer* og *feilsøking*. Videre presenteres eksempler innenfor hver av kategoriene.

Abstraksjon

Shute et al. (2017) beskriver kategorien abstraksjon som å trekke ut relevant informasjon, avdekke trender og mønstre, og modellere et system som kan programmeres. Med bakgrunn i tidligere forskning baserer kategorien abstraksjon seg på det som er felles i beskrivelsene til Shute et al. (2017), Barr og Stephenson (2011), Anderson (2016) og Wing (2006). Felles for beskrivelsene av abstraksjon er identifisering av mønstre, både i problemet og/eller løsningen, samt å bruke det for å løse lignende problemer. Kategorien abstraksjon baserer seg derfor, i analysen, på elevenes evne til å trekke ut relevant informasjon av et problem. For deretter å identifisere mønstre, som kan brukes til å løse lignende problemer.

Det første eksemplet viser til når elevene skal lage et rektangel. Ola uttrykker at de bare trenger å endre litt på koden fra når de lagde kvadratet.

96	Ola	Rektangel, men hvordan skal vi lage rektangel, er det en sånn lang firkant …
97	Per	… det er en firkant, en kvadrat, bare at to av sidene er lengre enn de øverste, de to på sidene
98	Ola	Åja, da trenger vi bare å bytte litt på koden

Elevene har allerede laget kvadratet, og fortsetter med å beskrive hvordan rektanglet skal se ut. Ola uttrykker i 98 at «da trenger vi bare å bytte litt på koden». Han har gjenkjent et mønster. Det gjør at de kan bruke koden til kvadratet for å lage rektanglet. Dette kategoriseres som *abstraksjon* fordi eleven har identifisert et mønster, og bruker dette for å løse andre problemer.

Eksemplene under viser at det ved flere anledninger er en sammenheng i hvordan elevparet går frem for å løse oppgavene. Ola mener ved flere tilfeller at antall grader

rotasjon er viktig når de skal programmere figurene. Det første sitatet er hentet fra arbeid med trekanten.

183	Ola	Ja, men det er litt vanskelig, fordi … hvor mange grader er en trekant?

Videre vises det til et eksempel på at Ola også ønsker å finne rotasjonen når man skal lage sekskanten.

242	Per	Er ikke det her bare en, 2, 2, 2 (teller høyt), sekskant? En sekskant!
243	Ola	Hvor mange grader kan han snu seg da?

Ola ser ut til å identifisere et mønster i arbeidet med å lage de ulike geometriske figurene. I utsagn 183 og 243 stiller han spørsmål ved hvor mange grader rotasjonen må være for å lage både trekanten og sekskanten. Per ser en annen sammenheng mellom rotasjonen for å lage trekanten og sekskanten, som det videre vises til.

267	Ola	Da må vi ta større grad. Vi tar 60!
268	Per	Ja, fordi det er halvparten av det vi hadde tidligere
269	Ola	Er det?
270	Per	Ja, av 120

Det ser ut til at Per legger merke til at rotasjonen for sekskanten er halvparten så stor som for den likesidede trekanten. Selv om det ikke formuleres en generell regel for rotasjon, og hvilken figur som lages, kan det se ut som Per har lagt merke til et mønster.

Problemnedbrytning

Den neste kategorien innenfor AT som beskrives av Shute et al. (2017) er problemnedbrytning. Kategorien kjennetegnes av å dele oppgaven i mindre og mer håndterlige deler. Før de ulike delene samlet brukes for å løse oppgaven.

Elevene fikk under intervjuet et generelt spørsmål: Hva er det første dere pleier å gjøre når dere får en oppgave? Ola forklarer i utdrag 320 at «av og til så deler jeg den i mindre deler», dersom det er en stor oppgave.

317 F: Hva er det første dere pleier å gjøre når dere får en oppgave da?

318 Ola: Jeg vet ikke

319 Per: Finne ut en måte å regne ut den på

320 Ola: Det første jeg pleier å gjøre er å se på oppgaven, og så av og til så deler jeg den i mindre deler

321 F: Dersom det er en stor oppgave ...

322 Ola: ... mm

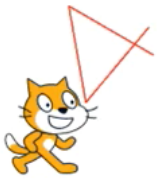
Ola ser på bakgrunn av svaret ut til å være kjent med arbeidsmåten problemnedbrytning. Lydopptaket fra programmeringen viser ett konkret eksempel på arbeidsmåten: Elevene deler opp den tredje figuren i oppgave 3 i mindre deler, for deretter å bruke de ulike delene til å programmere hele figuren.

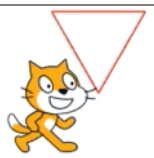
299	Ola	Men jeg skjønner ikke, hvilken figur er det?
300	F	Ja, hvilken figur er det? Hva tror dere?
301	Per	Egentlig er det ikke så vanskelig, jeg tror jeg vet hva vi kan gjøre. Vi kan lage en strek …
302	Ola	… det går an med gjenta-klossen, men det her blir vanskelig
303	Per	Ja, men det går an, fordi vi kan ta en strek og så resetter vi det, og så tar vi en ny strek, og så resetter vi det, og så tar vi en ny strek

Ola uttrykker at oppgaven kan bli en utfordring. I utdrag 302 sier han «men det her blir vanskelig». Per ser ut til å oppdage at figuren kan deles opp i tre streker, som kan programmeres hver for seg. Utdrag 303 viser hvordan Per mener figuren kan deles i tre. Hver av de tre strekene kan tegnes separat. Arbeidsmåten for å løse denne oppgaven kategoriseres som *problemnedbrytning.* Per deler problemet i mindre og mer håndterlige deler, de ulike delene til sammen utgjør hele figuren. Programmeringen av selve figuren vises det til under *algoritmer*.

Algoritmer

En algoritme er en presis beskrivelse av en endelig serie operasjoner, som skal utføres for å løse et problem, eller et sett med problemer. En utfordring med algoritmer er at de må følges nøyaktig i riktig rekkefølge. I Scratch kan elevene lage algoritmer med instrukser som Felix skal gjennomføre. Shute et al. (2017) deler algoritmer i fire underkategorier. Jeg har imidlertid bare identifisert *algoritmisk design* i datamaterialet. Eksemplet under viser hvordan elevparet bruker gjenta-klossen for å lage den likesidede trekanten.

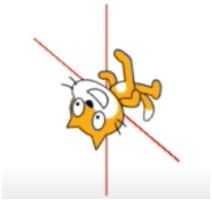
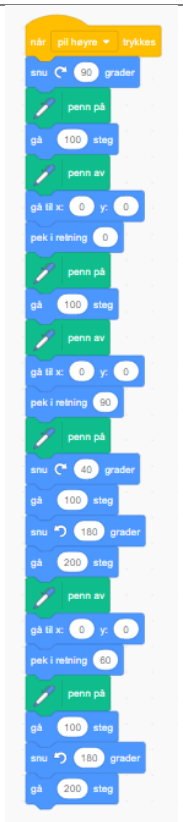
207	Per	Å, det var 130 ja, det ligner på et 4-tall. Men helt på slutten må vi ha en 90 tror jeg	

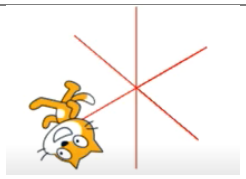
208	Ola	Okei, den her var for skarp (endrer til 120)	
209	Per	Ja! Vi klarte det	

Gjenta-klossen gjør at elevene kan lage en kode som er kortere, og dermed mer effektiv. Koden i utdrag 208 inkluderer også flere instruksjoner som er uvesentlige for figuren, men som de ønsket å ha med. For eksempel at fargen på pennen er rød, hvor Felix skal starte (0,100) og hvor han skal avslutte (-100,0). Elevene instruerer også Felix, «vent 1 sekunder», mellom hver bevegelse. Slik kan man se hver enkelt forflytning, med en kort pause mellom.

I det neste utdraget starter elevene med å programmere den tredje figuren i oppgave 2, slik Per skisserer fra problemnedbrytningen (utdrag 303). Elevene starter med å lage en kode som tegner den første av de tre strekene i figuren. Løsningen av denne oppgaven strekker seg over relativt lang tid. Utdragene det vises til er komprimert, og inneholder bare instruksjonene som er relevant for den endelige algoritmen.

432	Per	Nå må den resette, vent, men se her, jeg har glemt å bytte den her til 0. Ser du da er den der, og så kan vi ta	
433		(tester koden)	
		(endrer koden flere ganger)	

458	Ola	Sånn	
459	Per	Ja okei	
		(fortsetter med å endre koden)	
476		(tester koden)	

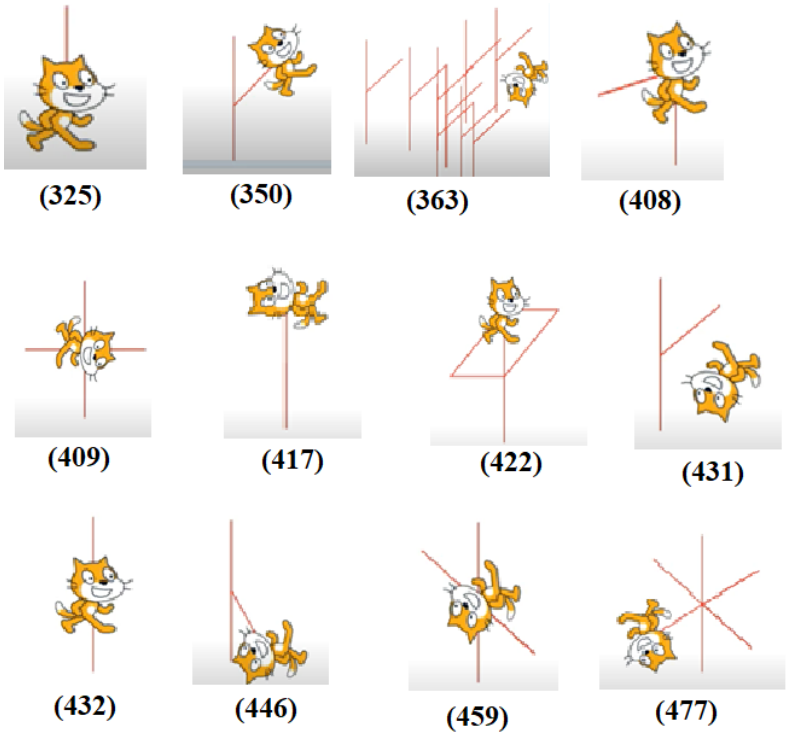
477	Ola og Per	Ja! Yeey!	

Koden i utdrag 432 tegner den første streken som vises på bildet i 433. Det Per ser ut til å mene med å resette er å bruke klossen «gå til x: 0 y: 0». Altså at Felix går til koordinatene (0,0). Koden som vises på bildet i utdrag 458 tegner figuren i 459. Koden avsluttes også her med å resette når Felix går til (0,0). Elevene gjør en liten feil underveis når de bruker «snu 40 grader», men legger ikke merke til det. Det riktige ville vært å snu 30 grader. Koden i utdrag 476 tegner hele figuren i 477. Selv om det er en liten feil i koden, kan figuren ved første øyekast se riktig ut. Elevene lager deretter en lang kode med instruksjoner for å løse problemet. Designet av algoritmen preges av prøving og feiling. Tall inne i klossene, og sammensettingen av ulike klosser, forandres flere ganger. Endringene underveis i arbeidet beskrives nærmere under kategorien *feilsøking*.

Feilsøking

Shute et al. (2017) beskriver at feilsøking er nødvendig for å teste om programmet er riktig og effektivt. Og at muligheten for å feilsøke er bakgrunnen for hvorfor programmering ofte brukes for å utvikle ferdigheter innenfor AT. Elevparets feilsøking baserer seg i stor grad på prøving og feiling.

Utgangspunktet for å se om elevene har programmert riktig er hvordan figuren tegnes i Scratch, når de kjører programmet. Det er derfor utseendet til figuren ser ut til å være avgjørende for når feilsøkingen til elevene avsluttes. Dersom figuren ser ut som elevene ønsker, ser de seg ferdig med oppgaven, og går videre til neste. Figuren under viser en sammensetning av skjermbilder, dette fra når elevene forsøkte å programmere den tredje figuren i oppgave 2. Figuren viser mangfoldet av forsøk, som til slutt leder frem til den ønskede figuren, som vises i utdrag 477.



Den sammensatte figuren viser at guttene tester programmet flere ganger, ettersom nye forslag kommer frem, og endringer må gjøres. Under kategoriene *problemnedbryting* og *algoritmer* ble det vist til den ønskede fremgangsmåten. Den var å dele figuren i tre streker. Gjennomføringen ble noe mer kaotisk enn den planlagte fremgangsmåten, og besto av flere forsøk med feilsøking for å få den ønskede koden. Som til slutt tegnet hele figuren.

4.2.2 Elevpar 2 – Ane og Mia

I datamaterialet til Ane og Mia identifiseres komponentene Shute et al. (2017) beskriver som *abstraksjon, algoritmer* og *feilsøking*. Videre presenteres eksempler på kategoriene.

Abstraksjon

Elevene bruker relativt lang tid på å komme i gang med oppgavene, noe som kan ha ulike forklaringer. Flere eksempler fra datamaterialet kan tyde på at elevene sliter med å finne ut av hva oppgavene spør etter, og hvordan de skal angripe problemet. Det første utdraget er hentet fra oppstarten med oppgave 1.

1	Ane	(leser oppgaven høyt.)
2	F	Forsto dere oppgaven?
3	Ane	Nei, jeg vet ikke
4	F	Så det første dere skal lage er et kvadrat
5	Mia	Ja, men hva hvis vi ikke …
6	Ane	… skal vi tegne det her?

Ane spør i utdrag 6 om de skal tegne kvadratet, kanskje fordi det er hennes vanlige måte å løse geometrioppgaver på. Videre vises det til et utdrag fra oppstarten med oppgave 2. Ane spør om de må gjøre oppgaven, Mia sier videre at «vi forstår ikke den her».

287	Ane	Må vi gjøre den her?
288	Mia	Vi forstår ikke den

Det neste eksemplet er et utdrag fra løsningen av oppgave 2. Elevene velger å gå bort fra oppgaven, og lager noe annet, som de ikke vet hva er.

382	F	Hvilken oppgave var dere på?
383	F	Hva var det dere prøvde å lage?
384	Mia	Det vet vi ikke
385	Ane	Ingen ting, vi prøvde bare å lage noe

Ane og Mia ser ut til å ha problemer med å hente ut viktig informasjon fra oppgaveteksten, slik at de kan løse problemet. De klarer ikke å forstå hvordan figurene skal lages uten hjelp. Arbeidsmåten deres preges av å prøve seg frem, men uten å forsøke å finne sammenhenger eller mønstre som kan brukes videre.

Algoritmer

Under *abstraksjon* ble elevenes utfordringer med å forstå oppgaven presentert. Likevel var elevene, med litt hjelp, i stand til å lage instruksjoner for å tegne flere av figurene. Det

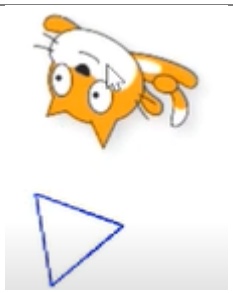
kan se ut som det finnes et fellestrekk i hvordan Ane og Mia lager algoritmer, for å løse oppgaven. De trykker flere ganger på blokken med klosser, dersom koden ikke lager hele figuren. De neste eksemplene viser hvordan elevene lager ufullstendige algoritmer, både for rektanglet, trekanten og sekskanten, og trykker flere ganger på koden. Det første eksemplet fra datamaterialet er for rektanglet.

216	Ane	30, 90, 30, 90, for å være sikker, gjør vi	
217	Ane	Okei, se	
218	Ane	(trykker flere ganger på koden)	
219	Ane	Fortsett!	
220	Ane	Okei, kanskje nå? (flytter Felix)	
221	Mia	Sånn? Omg vi har et rektangel. Nå må vi lage kvadratet igjen	

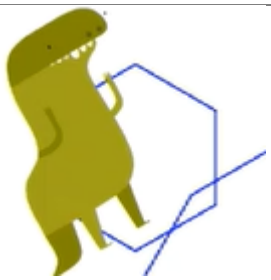
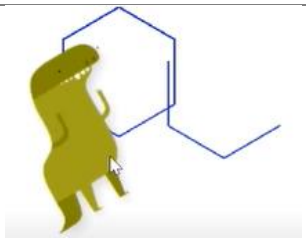
Den første gangen elevene trykker på koden tegner Felix halve rektanglet, som bildet i 217 viser. Når koden trykkes en gang til tegnes den andre halvdelen av rektanglet (se bilde i utdrag 218). Til sammen utgjør dette hele kvadratet.

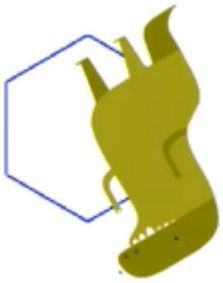
Det neste eksemplet er fra den likesidede trekanten. Også her kommer det frem at elevene lager en ufullstendig algoritme. Den tegner hele trekanten, men først når den trykkes på tre ganger.

265	Ane	120 grader snu, kanskje?	
266	Mia	Sikkert, jeg vet ikke	
267		(trykker en gang på koden)	

268		(trykker 3 ganger på koden, og flytter Felix)	
269	F	Hva har dere tegnet nå?	
270	Ane	Ja!	
271	Mia	Likesidet trekant	

Bildet i utdrag 267 viser hvordan algoritmen de har laget, 265, tegner en tredjedel av trekanten. På samme måte som for rektanglet trykker elevene flere ganger på koden, slik at den tegner en likesidet trekant. Dette vises i utdrag 268. I motsetning til de to foregående eksemplene, benyttes imidlertid gjenta-klossen for å lage sekskanten. Elevene bruker koden som er oppgitt i oppgave 3.

451		(endre på koden som er oppgitt i oppgave 3)	
452		(trykker to ganger på koden)	
453	Ane	1,2,3,4,5,6	
454	Mia	We did it	
…			
480	F	Gikk den hele sekskanten?	
481	Mia	Nei	
482	Ane	Den gikk halve	
483	F	Halve ja, hva må dere endre på koden for at den skal gå hele da?	
484	Ane	Der kanskje, nei, der	

485	F	Hva må dere endre?	
486	Mia	3-tallet til 6	
487	f	Hvilken figur ble det?	
488	Ane	Sekskant!	

Elevene benyttet seg altså av gjenta-klossen i algoritmen for å lage sekskanten. Koden som vises på bildet i utdrag 451 tegner halve sekskanten. For elevparet ble arbeidsmåten med å lage ufullstendige algoritmer, og trykke flere ganger på koden, suksessfull for flere av figurene. Bruken av gjenta-klossen gjør at man slipper å skrive unødvendig lange koder. Som i sin tur gjør programmeringen mer effektiv. Det er viktig å påpeke at prosessen med å komme frem til kodene ofte var tidkrevende, og preget prøving og feiling. Det siste eksemplet viser også hvordan elevene løste feil i programmeringen, noe som presenteres nærmere under *feilsøking*.

Feilsøking

Prøving og feiling ble en naturlig del av feilsøkingen til elevparet. Elevene kjører programmet flere ganger, for å undersøke om det funker eller ikke. Flere av eksemplene fra datamaterialet viser at elevene starter med å prøve seg frem, for å deretter endre på koden dersom det blir feil. Å prøve seg frem beskrives som et bevisst valg i intervjuet.

238 F: Hvordan gikk dere frem for å løse de ulike oppgave her da?

239 Ane: Vi prøvde oss frem

240 F: Var det et bevisst valg?

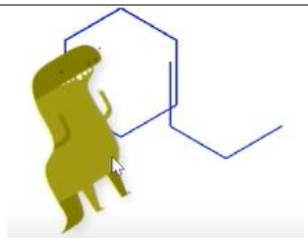
241 Ane: Ja!

242 Mia: Mm

En arbeidsmåte som hovedsakelig baserer seg på prøving og feiling viser seg å være tidkrevende. Elevparet klarte likevel å lage noen av de mer utfordrende figurene ved denne arbeidsmåten, som for eksempel sekskanten.

Det neste eksemplet tar utgangspunkt i utdragene 451 – 488, som det ble vist til under elevparets *algoritmer*. Elevene brukte koden de fikk oppgitt i oppgave 3 for å løse oppgave 2. Koden som tegnet en tolvkant ble dermed brukt for å lage sekskanten. Elevparet bruker *feilsøking* når de forsøker å endre en kode, for å tegne den ønskede figuren. Koden som er avbildet i utdrag 451 viser at elevparet har endret det som opprinnelig var «gjenta 12

ganger» til «gjenta 3 ganger». «Gå 30 steg» ble endret til «gå 60 steg», og «snu 30 grader» til «snu 60 grader». Videre ble koden fra 451 endret i forsøket på å lage en sekskant. Bildet i utdrag 482 viser både halv og hel sekskant. Da de trykket to ganger på koden, fikk elevene fram hele sekskanten.

451		(elevene lager denne koden)	
…			
482	Ane	Den gikk halve	

Hele prosessen med å endre koden, som opprinnelig tegnet en tolvkant, til å lage en sekskant, kategoriseres som *feilsøking.* Elevene tar utgangspunkt i en kode som de vet tegner en geometrisk figur. De forsøker deretter å bruke denne for å endre koden, slik at de får tegnet den geometriske figuren de ønsker.

4.3 Oppsummering av funn fra analysen

Funn fra datamaterialet viser at arbeidet med programmering i geometri har ført til matematisk resonnering blant alle elevene i studien. Ola og Per sin matematiske resonnering tyder på at programmeringsøkten kan være en arena for å utforske geometriske figurer. Ola og Per resonnerer med utgangspunkt i visuelle kjennetegn, egenskaper, og det som beskrives som abstrakt resonnement. Datamaterialet til Ane og Mia viser at bruken av matematiske resonnementer baserer seg utelukkende på *visuelle kjennetegn* under programmeringen.

Det som blir tydelig i funnene mine, er at kunnskap om egenskapene til figurene viser seg å være viktig for å kunne programmere effektivt. Funnene mine tyder videre på at programmering i Scratch kan støtte en tidlig utvikling av abstrakte tanker rundt geometriske figurer, spesielt sirkelen. Utvikling av abstrakte tanker og resonnementer på 6.trinn ville kanskje ikke vært mulig på samme måte, uten verktøy som Scratch for å lage figurene.

Prøving og feiling benyttes innenfor flere av de ulike kategoriene som beskrives av Shute et al. (2017). Men er spesielt sentralt innenfor elevenes design av *algoritmer* og *feilsøking.* Noe som ser ut til å skille arbeidsmåtene til de ulike elevene er abstraksjon. Ane og Mia klarer ikke å identifisere mønstre i problemet, og metoden som dermed kan brukes for å løse lignende problemer, noe Ola og Per gjør. Noe som også skiller parenes arbeidsmåter, er at det bare er Ola og Per som benytter problemnedbrytning.

5 Drøfting

I dette kapitlet skal jeg, ved hjelp av det teoretiske rammeverket, drøfte hvordan mine funn fra datamaterialet kan være med på å besvare problemstillingen: *«Hva kjennetegner et utvalg 6. klassingers matematiske resonnering, og deres arbeidsmåter i programmering av geometriske figurer?»* Funnene fra analysekapitlet diskuteres opp mot det teoretiske rammeverket, og sees også i sammenheng med tidligere forskning. I delkapittel 5.1 drøftes elevenes matematiske resonnering. Videre drøftes elevenes arbeidsmåter i delkapittel 5.2.

5.1 Elevenes matematiske resonnering

I dette delkapittelet vil jeg drøfte det første forskningsspørsmålet: *Hvilken type matematiske resonnementer bruker elevene i møte med programmeringsoppgaver i geometri?*

Funnene fra programmeringsøkten og intervjuet viser at arbeidet i Scratch har ført til matematisk resonnering. Dette med utgangspunkt i de geometriske figurenes visuelle kjennetegn, egenskaper, og de mer abstrakte tankene om figurene. Eksemplene som ble presentert i analysen viser at ulike elever benytter seg av ulike typer resonnementer, også innenfor samme oppgave. Alle elevene benytter resonnementer som tilhører kategorien *visuelle kjennetegn* under programmeringen. Det ser ut som om elevene som benytter seg av resonnementer med utgangspunkt i *egenskaper,* klarer å programmere de geometriske figurene på en mer effektiv og korrekt måte.

Kaput (1992) beskriver at Logo har hatt en positiv effekt på læring hos barneskoleelever. Dette innenfor læring av geometri, klassifisering av figurer, estimering av vinkler og lengder, samt forståelse av vinkelstørrelser og retning. Calao et al. (2015) og Brown et al. (2013) konkluderer i sine studier med at programmering i Scratch er en effektiv måte for å øve på problemløsning og resonnering i matematikk. På bakgrunn av at mine funn samsvarer med disse resultatene, mener jeg å kunne si at måten elevene arbeidet med Scratch på var en god tilnærming til matematisk resonnering, som også støtter læring innenfor geometri. Med utgangspunkt i funnene som er presentert, skal jeg drøfte de ulike matematiske resonnementene jeg har funnet i datamaterialet. Videre blir elevenes resonnementer brukt til å argumentere for deres antatte van Hiele-nivå.

5.1.1 Visuelle kjennetegn

I eksemplene fra Ane og Mia, presentert under *visuelle kjennetegn,* fant jeg to trekk som viser hvordan elevene bruker denne kategorien av resonnementer. Elevene benytter seg av *visuelle kjennetegn* i resonnementet, når de skal beskrive hvordan figuren ser ut. Både før og etter de lager et program for figuren tar beskrivelsen utgangspunkt i hvordan figuren ser ut. Et annen relevant trekk i elevenes resonnering er bruken av *størrelse, dette* for å beskrive hva som skiller rektanglet og kvadratet. Resonnementet til begge elevene handler om at størrelsen er avgjørende for å skille figurene.

I sine resonnementer bruker Mia bare kategorien visuelle kjennetegn, noe som viser en lav forståelse, og kjennetegner ifølge Lehrer et al. (1998) en elev på van Hiele-nivå 0. Dersom man ser på enkelte av resonnementene til Mia innenfor denne kategorien, er de

ikke korrekte, for eksempel i utsagn 151, vist til i delkapittel 4.1.2. Ut ifra påstanden om at figuren må telles som et kvadrat, kan det tyde på at Mia ikke har utviklet en fullstendig forståelse for at figuren må være lukket, for å defineres som et kvadrat. Eksemplet mener jeg tydelig viser at eleven baserer sitt resonnement på den visuelle fremstillingen. På en annen side viser Mia et tydelig forsøk på å beskrive rektanglet generelt, når hun i utdrag 240 sier at «et rektangel er at den er lang og kort». Mia forsøker å lage en generell beskrivelse for rektanglet. Selv om beskrivelsen er upresis, kan det tyde på at eleven nærmer seg en forståelse av egenskapene til akkurat denne figuren.

Ane benytter seg, i likhet med Mia, av visuelle kjennetegn i sine resonnementer under programmeringen. Noe som er interessant er at Ane flere ganger benytter seg av *egenskaper* for å beskrive figurene under intervjuet, mens den samme eleven konsekvent benyttet *visuelle egenskaper* under programmeringen. En mulig forklaring på dette er at jeg stilte spørsmål under intervjuet, noen ganger kanskje også ledende spørsmål. Dette bidro kanskje til at eleven var i stand til å resonnere, med utgangspunkt i egenskapene til figuren. En annen mulig forklaring er at elevene er vant til å tegne figurene i arbeid med geometri, uten at fokuset nødvendigvis er rettet på egenskapene til figuren. Noe som støtter den siste forklaringen, er at det i delkapittel 4.2.2 vises til at det første spørsmålet Ane stilte under programmeringsøkten var: «Skal vi tegne det her?» (utsagn 6). Kanskje vil videre arbeid med Scratch føre til det samme som Clements og Battista (1992) fant i arbeid med Logo. Det vil si at arbeid med programmering i geometri, kan støtte en overgang fra van Hiele-nivå 0 til 1, der man fokuserer på figurenes egenskaper.

I sine resonnementer bruker også Ola og Per *visuelle kjennetegn,* med fokus på *størrelse* for å beskrive rektanglet. Ola beskriver rektanglet som en lang firkant, mens Per sitt resonnement skiller seg fra de tre andre elevene. Han uttrykker at det er et kvadrat, bare at to av sidene er lengre enn de øverste. Det ser på samme måte som for Mia ut til at Per ønsker å beskrive figuren generelt, men at han kanskje ikke har den nødvendige kunnskapen som kreves for å lage en presis definisjon av rektanglet.

Flere studier viser at arbeid med Scratch utvikler matematiske ferdigheter innenfor vinkler, orientering og spatial forståelse, og at det kan skje på relativt kort tid (Fessakis et al., 2013; Calder, 2010). I arbeidet med Scratch ga elevene datamaskinen instruksjoner for hvordan figurene skulle tegnes. Noe som baserer seg på kunnskap om egenskapene til figuren. Elevene som benytter seg av visuelle kjennetegn ble nødt til å prøve seg frem for å programmere figurene. Det viser at det også var mulig å resonnere seg frem til egenskapene til figurene.

5.1.2 Egenskaper

Funnene i denne studien tyder på at det er viktig å ha en forståelse for figurenes egenskaper for å kunne programmere effektivt. Å ha forståelse i geometri baserer seg på en mer abstrakt tanke av figurene. Resonnementer basert på egenskapene til en figur er på et høyere nivå enn den visuelle fremstillingen (Lehrer, 1998).

Intervjuet var en setting der Ane resonnerte på et høyere van Hiele-nivå, enn under programmeringen. Med utgangspunkt i tidligere forskning ser dette funnet ut til å skille seg fra det man på forhånd kunne forvente. Clements og Battista (1992) viser til at elever som i utgangspunktet bare er i stand til å identifisere rektanglet visuelt, noe som samsvarer med nivå 0, ved hjelp av Logo blir oppmuntret til å konstruere en definisjonsform for et rektangel, som kjennetegner tenking på nivå 1. Kanskje skyldes elevenes manglende fokus på egenskaper at selve programmeringen ble en større utfordring for dette elevparet? På

en annen side kan det også være at programmeringsøkten bidro til læring av figurenes egenskaper. Siden datamaterialet kun ble samlet inn fra en økt, fikk ikke elevene muligheten til å vise utviklingen sin før intervjuet.

I motsetning til guttenes visuelle beskrivelse av rektanglet som «en lang firkant» (Ola i utsagn 96), og «en kvadrat, bare at to av sidene er lengre enn de øverste» (Per i utsagn 97) i delkapittel 4.1.1, bruker Ola og Per lengden på sidene og vinklene når figuren lages i Scratch. Dette kan tyde på at Scratch i deres tilfelle bidrar til å legge vekt på egenskapene til figuren når den programmeres. Dette samsvarer med funn fra Clements og Battista (1992). Elevenes visuelle beskrivelse kategoriseres som et resonnement med visuelle kjennetegn, samsvarende med nivå 0. Men ved hjelp av Scratch konstruerer elevene en sekvens med kommandoer, for å tegne rektanglet ut ifra figurens egenskaper. Noe som ifølge Clements og Battista (1992) samsvarer med nivå 1 tenkning. Scratch ser i dette tilfellet ut til å oppmuntre elevene til å resonnere på et høyere nivå, enn når de skal beskrive figurene.

Dersom man på forhånd av programmeringen ikke har kunnskap om figurenes egenskaper, vil arbeid med Scratch legge til rette for at elevene oppdager viktige egenskaper. Scratch ser derfor ut til å være egnet til å arbeide med kompetansemål etter 6. trinn i fagfornyelsen. Ett av kompetansemålene beskriver at elevene skal «bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønster». Man klarer for eksempel ikke å lage et kvadrat, uten at alle vinklene er 90 grader, og alle sidene er like lange.

5.1.3 Abstrakt resonnering

Funn fra datamaterialet viser at programmering i Scratch kan være et verktøy som egner seg for å utvikle en forståelse for de mer abstrakte delene av geometrien. Programmering bør behandles som andre verktøy innenfor matematikk, det vil si brukes dersom det er hensiktsmessig, ikke bare for egenverdien (Foerster, 2016). Et hensiktsmessig område kan se ut til å være innenfor abstrakt resonnering om sirkelen. Det kan være krevende å arbeide med sirkelens abstrakte definisjon i ordinær geometriundervisning på 6. trinn, men arbeidet i Scratch bidro til en meningsfull kontekst. Dermed kan man undersøke og utforske egenskaper, som kanskje ikke ville vært mulig uten dette verktøyet. Scratch ga blant annet elevene mulighet til å utforske mangekanter, hva som skiller sirkelen fra mangekanter, og til å finne ut at en sirkel hverken har hjørner eller sider.

I datamaterialet kommer det, i flere eksempler, frem at Ola og Per beskriver figurer som ligner på sirkler som perfekte. Hva som menes med perfekt kommer ikke tydelig frem. Det kan være egenskapene til figuren, men det kan også referere til måten sirkelen ble laget på. Altså at vi laget sirkelen perfekt. I delkapittel 4.1.1 ble det vist at både Ola og Per beskriver flere av figurene de har laget som perfekte. For eksempel «Vi klarte det perfekt!» (Ola i utsagn 632), og «den der var helt perfekt jo» (Per i utsagn 649). Begge elevene uttrykker likevel at man ikke kan lage en perfekt sirkel i Scratch. Jeg tolker elevenes beskrivelse av perfekte figurer til å handle om egenskapene til figurene. Noe jeg begrunner med elevenes utsagn om at man alltid kan gjøre sidene kortere, og rotasjonen mindre, for å få en mer perfekt sirkel. Elevenes varierende beskrivelse av figuren kan tyde på at det, innledningsvis i løsningen, oppstår en konflikt mellom den visuelle fremstillingen, og den begrepsmessige forståelsen. Fischbein (1993) viser også til konflikter som kan oppstå mellom den visuelle fremstillingen og begrepsmessige forståelsen, og beskriver det som at de to systemene ikke har dannet et figuralt begrep.

Videre i arbeidet ser begge elevene ut til å forstå at man alltid kan rotere et mindre antall grader, og gjøre lengden på sidene kortere, dersom man gjentar instruksjonen flere ganger. Et eksempel er når Ola i utdrag 646 sier «dersom du tar 0.5, vi må bare doble det her til 720!». Her står 0.5 for både antall grader rotasjon og antall steg, altså «gå 0.5 steg og roter 0.5 grader». 720 står for antall ganger instruksjonen skal gjentas. Like etterpå, i 647, sier Ola «nei, det går ikke an å lage en perfekt sirkel». Per sier videre i utsagn 653: «Ja! Eller det går an å ta 0.1». Altså «gå 0.1 steg og roter 0.1 grader». Når han videre får spørsmål om det da blir en perfekt sirkel, svarer han i utsagn 661: «Ja. Nei! Det går an å ta 0,01». Elevenes oppdagelse av at man alltid kan gjøre rotasjonen og antall steg mindre, ved å øke antall ganger man gjentar instruksjonen, er et interessant funn. Det viser at elevene har forstått at man aldri vil kunne lage en «perfekt sirkel». Underliggende for det meste av geometri er spatial resonnering, evnen til å kunne se, undersøke og reflektere rundt spatiale objekter (Battista, 2007). Når man skal resonnere matematisk bruker man den matematiske sirkelen. Den har ingen farge, er ikke en materie eller masse, den er altså perfekt (Fischbein, 1993). Ola og Per ser ut til å ha forstått at en slik sirkel vil man hverken kunne tegne i boka, eller lage i Scratch.

5.1.4 Hvilket van Hiele-nivå kjennetegner elevene

Dersom man skal rangere de tre ulike kategoriene for resonnementer, ut ifra hvilken geometrisk forståelse de viser til, kan resonnementer med utgangspunkt i *visuelle kjennetegn* indikere en forståelse på et lavt nivå. *Egenskaper* på et middels nivå, og *abstrakt resonnement* på et høyt nivå. Lehrer (1998) hevder i sin studie at elever som utelukkende benytter seg av resonnementer med utgangspunkt i visuelle kjennetegn, gjerne befinner seg på nivå 0 av van Hiele-nivåene. Elever som tar i bruk viktige egenskaper som er betydelige for figuren benytter et resonnement innenfor kategorien *egenskaper*. Dette er på et høyere nivå, som passer godt med det van Hiele (1959/1984) beskriver som nivå 1. *Abstrakte resonnementer* i arbeid med geometriske figurer kan indikere en forståelse som er på et høyere nivå enn egenskaper. Dette fordi eleven må forstå viktigheten av definisjoner, uten å kunne relatere det til en visuell fremstilling. Det er tanken om den perfekte figuren som defineres. Eleven må derfor ha en forståelse for viktigheten av nøyaktige definisjoner, noe som kjennetegner det van Hiele (1959/1984) beskriver som nivå 2.

Tabell 2 viser en oversikt over hvilke resonnementer hver elev benyttet seg av under programmeringen og intervjuet. Elevenes antatte van Hiele-nivå tar utgangspunkt i kategoriene av resonnementer som ble benyttet.

	Elevpar 1		**Elevpar 2**	
	Ola	**Per**	**Ane**	**Mia**
Visuelle kjennetegn	JA	JA	JA	JA
Egenskaper	JA	DELVIS	DELVIS	NEI
Abstrakt	DELVIS	DELVIS	NEI	NEI
Elevenes antatte van Hile-nivå	Mellom 1 og 2	Mellom 0 og 1	Mellom 0 og 1	0

Tabell 2. Oversikt over matematiske resonnementer elevene benyttet seg av.

Jeg opplever det, som Burger og Shaughnessy (1986), utfordrende å plassere elevene innenfor et nivå. Spesielt når en elev benytter resonnement som beskriver et nivå, samtidig som eleven viser forståelse for enkelte kjennetegn på et høyere nivå i noen sammenhenger. For eksempel støtter Per flere av sine resonnementer, med utgangspunkt i egenskaper, på forklaringen til Ola. Det gjør det vanskelig å skille ut hva Per egentlig har forstått selv. Ane benytter seg av visuelle kjennetegn i programmeringen, men benytter seg av egenskaper på noen av forklaringene i intervjuet. Spørsmålet blir om dette gjør Ane til en elev på nivå 1.

Noe som derimot fremstår som tydelig i datamaterialet, er at alle elevene i studien benytter seg av matematisk resonnering med utgangspunkt i visuelle kjennetegn. Jeg tolker det derfor som at alle elevene i denne studien er på minimum nivå 0. Ingen av elevene defineres til å være på det Clements og Battista (1992) omtaler som et pre-nivå. Årsaken til det er at alle viser at de kan skille mellom ulike geometriske figurer med rette linjer, som for eksempel å skille trekanter og firkanter.

Selv om samarbeidet mellom elevene ikke har vært hovedfokuset i denne studien, velger jeg å påpeke noe som jeg mener kan ha hatt innvirkning på funnene som presenteres. Og følgelig også den videre drøftingen: Elevene som samarbeidet under programmeringen ser, ut ifra tabell 2, til å være på ulike van Hiele-nivå. Det kan føre til at de ifølge van Hiele (1959/1984) ikke snakker samme språk. Hvert nivå har sitt eget språk, med symboler og et system for sammenhenger mellom symbolene. En beskrivelse som er riktig på et nivå, kan virke feil på et annet nivå. To personer som resonnerer på forskjellige nivåer har problem med å forstå hverandre, fordi ingen klarer å følge tankeprosessen til den andre (van Hiele, 1959/1984). Et eksempel på dette kan være at rektangel kan ha forskjellig betydning for to elever på forskjellig nivå, dersom den ene tenker på det visuelle utseendet, og den andre tenker på det som en bærer av egenskapene (Clements & Battista, 1992). På en annen side forklarer Foerster (2016) i sin studie at samarbeidet mellom elevene på en datamaskin viste seg å være fruktbart. Dette fordi elevene ble nødt til å diskutere programmene med hverandre. Scratch bidro med et felles matematisk språk. Det er mulig arbeid i Scratch gjør det enklere for elever på forskjellige van Hiele-nivå å samarbeide. Det kan skyldes at det kreves flere kunnskaper enn geometrisk forståelse, for å kunne lage figurene.

5.2 Elevenes arbeidsmåter med programmering

I dette delkapittelet vil jeg drøfte det andre forskningsspørsmålet: *Hvilke arbeidsmåter innenfor algoritmisk tenkning benytter elevene i løsningen av programmeringsoppgaver i matematikk?*

Elevene i studien har liten eller ingen erfaring med programmering, og benytter seg derfor kanskje ikke fullstendig av kategoriene innenfor AT, slik det ordinært presenteres av blant annet informatikeren Wing (2006). Rammeverket til Shute et al. (2017) retter seg mot elever fra 1.- 12. trinn. Det betyr at komponentene jeg benyttet meg av i analysen skal kunne identifiseres i programmeringsarbeid, helt ned til førsteklassen. Det vil derfor være naturlig å anta at ulike komponenter innenfor AT kan benyttes på ulike nivå, og over tid utvikles til mer bevisste arbeidsmåter. Atmatzidou og Demetriadis (2016) påpeker i sin studie at det tar tid å utvikle ferdigheter innenfor AT.

Mine funn viser at programmeringen til elevene i hovedsak baserer seg på prøving og feiling, blant annet innenfor kategoriene *algoritmer* og *feilsøking*. Hazzan et al. (2015) viser også til funn av at elevenes strategier domineres av prøving og feiling når de lager programmer. Elever som kjenner egenskapene til figurene, og benytter seg av dem, ser ut til å lage mer effektive og korrekte algoritmer. Funnene om at egenskapene til figurene er viktig for å kunne programmere, ser ut til å samsvare med tidligere forskning Clements og Battista (1992), som viser til at det også var tilfellet i arbeid med Logo.

5.2.1 Hva kjennetegner elevenes arbeidsmåter?

Abstraksjon

Kategorien abstraksjon baserer seg på evnen elevene har til å trekke ut relevant informasjon av et problem, for deretter å avdekke trender eller mønster i problemet og/eller løsningen. Dette kan så brukes for å løse andre problemer. Hvis elevene ikke har tilstrekkelige kunnskaper til å forstå hva oppgaven ber dem om, har det liten betydning at de eventuelt har kunnskapene til å lage et fungerende program.

Datamaterialet viser til flere eksempler der Ane og Mia har utfordringer med å forstå hvordan programmeringsoppgavene skal angripes. Eksempelvis når Ane uttrykker «Ingen ting, vi prøvde bare å lage noe» (utdrag 385), videre når Mia uttrykker «Vi forstår ikke den her» (utdrag 288). Eksemplene fra begge elevene er ikke engangstilfeller, men preger oppstarten både på oppgave 1 og 2. Utfordringen med å forstå oppgaven er også observert i andre studier (Muller, 2005; Robins, Rountree & Rountree, 2010). For at elevene skal forstå hva oppgaven ber dem om, må de ha tilstrekkelig kunnskap til å forstå hva slags program de blir bedt om å lage. Siden oppgavene består av matematiske problemer, må elevene også ha matematikkunnskaper til å forstå oppgaven. Det er vanskelig å si om det er matematikkunnskapen eller programmeringskunnskapen, eventuelt begge deler, som gjør det utfordrende for jentene å forstå oppgaven. På den ene siden viser funn som presenteres i delkapittel 5.1.1 at elevene har matematikkunnskapene som kreves for å beskrive figurene med utgangspunkt i visuelle kjennetegn. Det gjør elevene i stand til å tegne figurene for hånd. På den andre siden ble det vist at elevene ikke benyttet seg av egenskaper i sine resonnementer, noe som er vesentlig for å kunne programmere figurene i Scratch. Jeg vil på bakgrunn av dette argumentere for at det er en kombinasjon av mangler i begge kunnskapene, matematikk og programmering, som gjør det utfordrende å tekke ut essensen av problemet.

I analysekapitlet kommer det frem at Ola og Per ved flere anledninger klarer å bruke løsningen på en oppgave til å identifisere et mønster, som deretter kan brukes i møte med andre problemer. Ola forklarer at man kan endre litt på koden som tegner kvadratet for å tegne et rektangel, noe som tydelig viser at eleven har lagt merke til et mønster i hvordan figurene programmeres. Wing (2006) beskriver abstraksjon som den viktigste delen av AT. Jeg vil her argumentere for det samme. Dersom man ikke forstår problemet, vil det være vanskelig å lage et fungerende program som løser det. Videre ser det ut til å være avgjørende å finne mønster og sammenhenger mellom de ulike figurene, for å kunne programmere på en effektiv måte.

Problemnedbrytning

I arbeidet med å tegne den tredje figuren i oppgave 2 mente Per at problemet kunne deles i tre. Elevene angrep deretter problemet med det Shute et al. (2017) beskriver som problemnedbrytning. Dette er det eneste eksemplet i datamaterialet hvor arbeidsmåten

tydelig identifiseres. Det kan skyldes at Ola og Per ikke oppfattet de andre oppgavene som så store, eller kompliserte, at det var nødvendig å dele problemet i mindre deler.

Ane og Mia viser ikke noen form for bevissthet rundt å bryte ned oppgaven i mindre deler, hverken i intervjuet eller programmeringsøkten. Det er ofte sånn at studenter i introduksjonskurs ikke får undervisning i hvordan de skal bryter ned større problemer (Keen & Mammen, 2015). Når vi her ser at elevene hverken nevner noen form for problemnedbrytning, eller anvender det i løsningen av oppgavene, indikerer det at elevene ikke har fått tilstrekkelig undervisning om dette. Alle elevene i studien har liten erfaring innen programmering, så det vil være naturlig å anta at de ikke har fått tilstrekkelig undervisning. Det kan imidlertid også skyldes at problemene ikke oppfattes som så store/komplekse at det er nødvendig å dele problemet i mindre deler.

Algoritmer

Løkker er et viktig begrep i programmering, det nevnes også i kompetansemål i fagfornyelsen (Utdanningsdirektoratet, 2019). I Scratch kan elevene bruke løkker ved å benytte seg av gjenta-klossen. I matematikk kan forståelsen av gjenta-klossen knyttes til additiv og multiplikativ tenking. Ola og Per lager en kode for trekanten ved å bruke gjenta-klossen, noe som gjør programmeringen mer effektiv. Fra å sette sammen fire «sett» av klosser for å lage kvadratet, har de sett effektiviteten av å sette én gjentablokk med tallet tre rundt ett enkelt «sett» av klosser, for å lage trekanten. Dette samsvarer med forskning av Wilson og Moffat (2010) og Sáez-López, Román-González og Vázquez-Cano (2016), som viser at elevene på bare noen uker med Scratch lærte seg de grunnleggende konseptene ved programmering, som rekkefølge, gjentakelse og utvelgelse av riktige klosser.

I analysen ble det vist at Ane og Mia ved flere anledninger lager ufullstendige instruksjoner som tegner hele figuren, dersom man kjører programmet flere ganger. Elevene ser ikke ut til å benytte seg av gjenta-klossen selvstendig, noe som kan ha ulike forklaringer. På den ene siden kan det skyldes at elevene har så liten erfaring med programmering at de ikke var klare over fordelene med å bruke den. På den andre siden kan det være at kodene elevene lager gir riktig figur når man kjører dem flere ganger. Noe som også kan sies å være en effektiv løsning på problemene. Flere studier viser til at det tar tid for elevene å lære seg AT ferdigheter (Atmatzidou & Demetraidis, 2016; Wohl et al., 2015). En introduksjons- og programmeringsøkt, til sammen 4 skoletimer, kan se ut til å være knapt for å lage effektive algoritmer for flere av elevene.

Feilsøking

Noe som kjennetegner feilsøkingen til elevene er at de endrer instruksjonene i koden, dersom den ønskede figuren ikke tegnes. Man kan derfor si at det ofte er prosessen ved å endre algoritmen som er elevenes feilsøking. Elevene endrer tall på klossene, blant annet for å rotere mer/mindre, antall steg, samt antall gjentakelser.

I programmering beskriver Papert (1980) at «debugging» en viktig del av prosessen når man skriver instruksjoner til datamaskinen. Å se etter feil, rette dem opp, og på den måten finne en bedre løsning på problemet, er en sentral arbeidsmåte blant elevene i denne studien. I delkapittel 4.1.1 vises det til at guttene endrer lengden på sidene i kvadratet fordi man ikke kan se hele figuren. Scratch kan på mange måter sammenlignes med Logo, og gir elevene mulighet til å prøve seg frem, uten å komme med en ladet tilbakemelding. Papert mener Logo er et sted der elever ikke blir dømt, men et sted der det er lov til å ta feil (Papert, 1980). Ane og Mia beskriver det som et bevisst valg å prøve seg frem for å

løse oppgavene. Dersom instruksjonene som blir gitt til datamaskinen ikke tegner den ønskede figuren, forsøker de å endre på tallene eller klossene koden består av. Underveis testes programmet, dette for å se om man nærmer seg det ønskede resultatet.

5.2.2 Prøving og feiling som arbeidsmåte

Elevenes utfordringer med å lage flere av figurene på en effektiv måte har blitt presentert i denne studien. Funnene mine tyder på at elevene benytter seg av varierende arbeidsmåter. Noe som likevel ser ut til å være et fellestrekk blant alle elevene, er en arbeidsmåte som baserer seg på prøving og feiling. Arbeidsmåten ser ofte ut til å benyttes av elevene dersom man enten: (1) ikke har en klar plan for å angripe oppgaven, eller (2) elevene får en figur som ser nesten riktig ut.

Med bakgrunn i mine funn, og studien til Hazzan et al. (2015), kan det tyde på at elevenes bruk av prøv- og feil-strategier er et resultat av elevenes manglende kunnskap innenfor gode arbeidsmåter i programmering. Mangelen på kunnskap kan skyldes at elevene har relativt liten erfaring med programmering, og at introduksjonsøkten ikke la nok fokus på at elevene utviklet gode arbeidsmåter. Hazzan et al. (2015) mener at introduksjonskursene fokuserer for lite på å lære problemløsningsstrategier. Det fører til at elevene utvikler egne, ineffektive strategier. På en annen side kan man anta at dette er kunnskap som utvikles over tid, og at prøving og feiling er en naturlig og utforskende fremgangsmåte for å bli kjent med programmering. Flere andre studier viser til at det tar tid å utvikle ferdigheter innenfor AT (Atmatzidou & Demetraidis, 2016; Wohl et al., 2015), noe denne studien også tyder på.

På samme måte som Papert (1980) beskriver Logo som et sted der elever ikke blir dømt, kan man si at Scratch også er et sted der det er lov til å ta feil. Den lave terskelen for å kunne programmere feil, og deretter rette det opp, kan bidra til et miljø der det er enklere for elevene å prøve seg frem for å finne svaret. Ved å finne feil og endre disse, ved såkalt «debugging», kan de prøve på nytt, og se om endringen har ført til at figuren tegnes riktig (Papert, 1980).

6 Avslutning

I denne oppgaven har jeg med innsamlet datamateriale og relevant teori, drøftet og belyst problemstillingen: *«Hva kjennetegner et utvalg 6. klassingers matematiske resonnering, og deres arbeidsmåter i programmering av geometriske figurer?».* Dette kapittelet presenterer konklusjonen, en vurdering av kvaliteten på studien og forslag til videre forskning.

6.1 Konklusjon

I datamaterialet har jeg funnet eksempler som viser at elevene gjennom arbeid med programmering har benyttet seg av matematiske resonnementer, med utgangspunkt i både visuelle kjennetegn, egenskaper og abstrakte resonnement. Scratch har vist seg å være et nyttig verktøy for arbeid med geometri. Spesielt nyttig var de innebygde mulighetene for å arbeide med vinkler og lengder, samt muligheten til å få ting til å skje gjentagende ganger. Dette ga elevene mulighet til å utforske mønstre i geometrien. Flere av elevene resonnerte abstrakt om sirkelens egenskaper, noe som ikke nødvendigvis er like enkelt med penn og papir. Elever på ulike van Hiele-nivåer ser ut til å samarbeide godt når de arbeider med Scratch. Det skiller seg ut fra tidligere forskning innenfor geometri, uten programmering.

Studien illustrerer kjente utfordringer, som at elever i stor grad benytter seg av prøving og feiling i programmering. Men den viser samtidig hvordan flere av elevene allerede etter 4 skoletimer benyttet seg av effektive arbeidsmåter, for å løse problemene. Det denne studien, og tidligere forskning, viser, er at elevene trenger tid for å utvikle bevisste og effektive arbeidsmåter innen programmering. Tid er en mangelvare i skolen. Noe som gjør at det er viktig å planlegge programmeringsøktene godt, og ha kompetente lærere. Lærerne skal være med på å utvikle neste generasjons arbeidstakere, og gi dem ferdigheter som blir viktige i fremtidens arbeidsmarked.

Et sentralt spørsmål når man innfører noe nytt i skolen, er hva som må prioriteres bort for å gi plass til det nye. Jeg mener denne masteroppgaven viser at det er mulig å bruke programmering som er verktøy der det er hensiktsmessig, på samme måte som andre eksisterende verktøy. Programmering i matematikk trenger ikke å bety at man må ta bort noe annet, for å få tid. Studien min viser at elevene arbeider med flere av de nye kompetansemålene fra fagfornyelsen innen både programmering og geometri. Dette er mulig allerede etter en introduksjonsøkt på 120 min.

Å undervise i programmering handler ikke bare om at elever skal lære seg å forstå datamaskiner, eller utvikle en bedre teknologisk kompetanse. Det handler også om å gi elevene gode verktøy for å løse oppgaver og problemer. Med studien har jeg fått nyttige erfaringer, når det gjelder hvordan jeg som fremtidig lærer kan benytte meg av programmering i matematikkundervisning. Jeg har utviklet to undervisningsopplegg, der studien min viser at elevene arbeidet med flere kompetansemål i matematikk fra fagfornyelsen på 6. trinn. Både innenfor programmering og geometri. Jeg vil derfor argumentere for at undervisningsoppleggene jeg har laget, egner seg for å introdusere programmering i matematikk på en meningsfull måte.

6.2 Vurdering av kvalitet på studien

Jeg har underveis i studien tatt metodiske valg, som påvirket funnene mine. Jeg har undersøkt et utvalg av fire elever på 6. trinn. Utvalget mitt er ikke stort nok til å kunne generalisere funnene til å gjelde andre elever. Det er imidlertid heller ikke hensikten med en kasusstudie. Studiens bidrag er en detaljert beskrivelse av hvordan elevene arbeidet med programmering av ulike geometrioppgaver. Jeg mener at studien har gitt verdifull informasjon om hvilke arbeidsmåter norske elever benytter, og hvordan de resonnerer i programmeringen av geometriske figurer. Jeg er klar over at dette ikke nødvendigvis er representativt, hverken for disse elevenes arbeidsmåter og resonnementer i andre situasjoner, eller for andre elever.

Jeg valgte å designe både introduksjonsøkten og selve programmeringsøkten for datainnsamlingen selv. Jeg er delvis inspirert av andre undervisningsopplegg. Det ble gjort på denne måten på bakgrunn av en mangel av gode eksisterende opplegg, som kunne passet for undersøkelsen min. En vellykket pilotundersøkelse bidro positivt. Dermed var jeg på forhånd av datainnsamlingen trygg på at det planlagte opplegget var tilfredsstillende, for å kunne undersøke problemområdet. Forskningsspørsmålene endret seg underveis gjennom store deler av studien, mens problemområdet forble det samme.

Studien viser hvordan man kan benytte Shute et al. (2017) sin litteraturstudie for å undersøke sentrale komponenter innenfor AT. Og bruke det for å beskrive hvilke arbeidsmåter elevene brukte. Studien viser også hvordan Lehrer (1998) sitt rammeverk for matematisk resonnering kan brukes. Dette for å beskrive elevers resonnement når figurene programmeres, uten at fokuset nødvendigvis er på å skille figurer fra hverandre, slik det ordinære rammeverket presenterer.

6.3 Videre forskning

Fagfornyelsen viser at både programmering og AT blir sentrale i den norske matematikkundervisningen. Det ble gjennomført en rekke studier før det ble bestemt at programmering skulle integreres i norsk skole, og hvordan dette på best mulig måte skulle gjennomføres. Rapporten til Ludvigsenutvalget om fremtidens skole (NOU 2015:8), og forsøket med å innføre valgfaget programmering på ungdomskolen, har begge vært sentrale steg på veien mot en norsk skole med programmering. Rapporten «Teknologi og programmering for alle» av Sanne et al. (2016), anbefalte i 2016 at det opprettes et nytt obligatorisk fag i grunnskolen. Dette faget skulle omfatte teknologi og programmering. Det skulle være et praktisk fag, hvor eleven fikk mulighet til å tilegne seg grunnleggende, teknologisk kompetanse. Etter deres vurdering er det særlig tre elementer som er sentrale for en vellykket implementering av teknologi og programmering, som eget fag i grunnskolen: lærerkompetanse, dedikerte timer, samt ressurser og engasjement fra skoleeier og skoleledelse.

Selv om vi nå vet at programmering ikke innføres som et nytt fag, inkluderes det i flere eksisterende fag. I forbindelse med innføringen av den nye læreplanen vil det være interessant å undersøke norske læreres kompetanse, spesielt relatert til programmering i matematikk. Tidligere forskning viser at innføringskurs i programmering ofte legger for lite vekt på problemløsningsstrategier, noe som gjør at elevene i stor grad benytter seg av prøving og feiling (Hazzan et al., 2015). En interessant studie vil være å undersøke hvordan ulike lærere gjennomfører den første økten i programmering med elevene, og hvordan ulike tilnærminger påvirker elevenes arbeidsmåter.

Jeg viste i delkapittel 5.1.4 til hvilket van Hiele-nivå elevene kjennetegnes av. Et eksperiment som også kunne vært interessant å gjennomføre, etter at programmering har blitt integrert i kompetansemål i matematikk, hadde vært en før- og etter-test av van Hiele-nivå, i en eksperiment- og kontrollgruppe. Da kunne man undersøkt om arbeid med programmering i Scratch hadde påvirket eksperimentgruppens geometriske forståelse, relativt til nivåene i en test av van Hiele-nivå. Dersom elevene i eksperimentgruppen hadde hatt en økning i van Hiele-nivå som var større enn kontrollgruppen, som for eksempel arbeidet med geometriske figurer i ordinær undervisning, kunne det tyde på at det er hensiktsmessig at alle elever på 6. trinn arbeider med programmering innenfor geometri, for å øke den geometriske forståelsen.

Det er relativt lite forskning innenfor programmering i matematikk med norske elever. Jeg ønsker å oppmuntre lærerstudenter i matematikkdidaktikk til å fortsette å studere dette fagområdet. Slik kan man få bedre innsikt i hvordan programmering kan brukes på en meningsfull måte, i matematikkundervisningen i norsk skole.

Litteraturliste

2015:8, N. *Fremtidens skole. Fornyelser av fag og kompetanser*. Oslo. Hentet fra https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/?ch=1&q=

Anderson, N. D. (2016). A call for computational thinking in undergraduate psychology. *Psychology Learning & Teaching, 15*(3), 226-234.

Atmatzidou, S. & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661-670.

Barcelos, T. S., Muñoz-Soto, R., Villarroel, R., Merino, E. & Silveira, I. F. (2018). Mathematics Learning through Computational Thinking Activities: A Systematic Literature Review. *J. UCS, 24*(7), 815-845.

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Acm Inroads, 2*(1), 48-54.

Battista, M. T. (2007). The development of geometric and spatial thinking. *Second handbook of research on mathematics teaching and learning, 2*, 843-908.

Benton, L., Hoyles, C., Kalas, I. & Noss, R. (2016). Building mathematical knowledge with programming: insights from the ScratchMaths project. Suksapattana Foundation.

Botten, G. (1999). *Meningsfylt matematikk* Bergen: Caspar forlag.

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology, 3*(2), 77-101.

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (s. 25).

Brown, Q., Mongan, W., Kusic, D., Garbarine, E., Fromm, E. & Fontecchio, A. (2013). Computer aided instruction as a vehicle for problem solving: Scratch programming environment in the middle years classroom. *Retrieved September, 22*(6.1), 1.

Burger, W. F. & Shaughnessy, J. M. (1986). Characterizing the van Hiele levels of development in geometry. *Journal for research in mathematics education*, 31-48.

Burton, D. M. (2010). *The history of mathematics: An introduction* (bd. 7). New York: McGraw-Hill.

Calao, L. A., Moreno-León, J., Correa, H. E. & Robles, G. (2015). Developing mathematical thinking with scratch. I *Design for teaching and learning in a networked world* (s. 17-27). Springer.

Calder, N. (2010). Using scratch: an integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom, 15*(4), 9-14.

Christoffersen, L. & Johannessen, A. (2012). *Forskningsmetode for lærerutdanningene* Abstrakt.

Clements, D. H. & Battista, M. T. (1992). Geometry and spatial reasoning. *Handbook of research on mathematics teaching and learning*, 420-464.

Cohen, L., Manion, L. & Morrison, K. (2017). *Research methods in education* routledge.

Creswell, J. W. & Poth, C. N. (1998). *Qualitative inquiry and research design: Choosing among five approaches* Sage publications.

Fessakis, G., Gouli, E. & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97.
Fischbein, E. (1993). The theory of figural concepts. *Educational studies in mathematics, 24*(2), 139-162.
Foerster, K.-T. (2016). Integrating programming into the mathematics curriculum: Combining scratch and geometry in grades 6 and 7. *Proceedings of the 17th annual conference on information technology education* (s. 91-96).
Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education.
Gjøvik, Ø. & Torkildsen, H. A. (2019). Algoritmisk tekning *Tangenten – tidsskrift for matematikkundervisning, 30(3)*, 31-37.
Grover, S. & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42*(1), 38-43.
Guba, E. G. (1981). Criteria for assessing the trustworthiness of naturalistic inquiries. *Ectj, 29*(2), 75-91.
Gutiérrez, A., Jaime, A. & Fortuny, J. M. (1991). An alternative paradigm to evaluate the acquisition of the van Hiele levels. *Journal for research in mathematics education, 22*(3), 237-251.
Hazzan, O., Lapidot, T. & Ragonis, N. (2015). *Guide to teaching computer science: An activity-based approach* Springer.
Jeannotte, D. & Kieran, C. (2017). A conceptual model of mathematical reasoning for school mathematics. *Educational studies in mathematics, 96*(1), 1-16.
Kaput, J. J. (1992). Technology and mathematics education. I D. A. Grouws & National Council of Teachers of Mathematics. (Red.), *Handbook of research on mathematics teaching and learning* (s. 515-556). New York: Macmillan Publishing Company.
Keen, A. & Mammen, K. (2015). Program decomposition and complexity in CS1. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (s. 48-53).
Kvale, S. & Brinkmann, S. (2009). *Interviews: Learning the craft of qualitative research interviewing* Sage.
Lehrer, R. (1998). Longitudinal study of children's reasoning about space and geometry. I *Designing learning environments for developing understanding of geometry and space* (s. 151-182). Routledge.
Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61.
Muller, O. (2005). Pattern oriented instruction and the enhancement of analogical reasoning. *Proceedings of the first international workshop on Computing education research* (s. 57-67).
Nilssen, V. L. (2012). *Analyse i kvalitative studier: den skrivende forskeren* Universitetsforlaget.
Opplæringslova. (1998). Lov om grunnskolen og den videregåande opplæringa Hentet fra https://lovdata.no/dokument/NL/lov/1998-07-17-61/KAPITTEL_1#KAPITTEL_1
Papert, S. (1980). *Mindstorms : children, computers, and powerful ideas*. New York: Basic Books.
Postholm, M. B. (2005). *Kvalitativ metode. En innføring med fokus på fenomenologi, etnografi og kasusstudier*. Oslo: Universitetsforlaget.

Project, T. B. (2014). Computational Thinking. Hentet fra http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for All. *Communications of the Acm, 52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Robins, A., Rountree, J. & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education, 13*(2), 137-172.

Sáez-López, J.-M., Román-González, M. & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129-141.

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., … Voll, L. O. (2016). Teknologi og programmering for alle-En faggjennomgang med forslag til endringer i grunnopplæringen-august 2016.

Sevik, K. m. f. (2016). Programmering i skolen. I S. f. I. i. utdanningen (Red.). Utdanningsdirektoratet. Hentet fra https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Shute, V. J. (1991). Who is likely to acquire programming skills? *Journal of educational Computing research, 7*(1), 1-24.

Shute, V. J., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Stake, R. E. (1995). *The art of case study research* Sage.

Usiskin, Z. (1982). Van Hiele Levels and Achievement in Secondary School Geometry. CDASSG Project.

Utdanningsdirektoratet. (2019). *Forslag - Læreplan i matematikk fellesfag 1.–10. trinn* Hentet fra https://www.udir.no/globalassets/filer/lareplan/fagfornyelsen/lareplanutkast/mat1-05---lareplan-i-matematikk-fellesfag-1.-10.-trinn.pdf?fbclid=IwAR2FZFCaUMFUX6GL7TcFuWEZMC8p6H7w1rJjtTdRz3oOU2EypUq4OTTlM2g

Van de Walle, J., Karp, K. & Bay-Williams, J. (2015). Elementary and middle school mathematics: teaching developmentally (ninth edition). *Essex: Pearson*.

van Hiele, P. M. (1959/1984). A Child's Thought and Geometry. I D. Fuys (Red.), *EnglishTranslation of Selected Writings of Dina van Hiele-Geldof and Pierre M. van Hiele* (s. 247-255).

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127-147.

Wilson, A. & Moffat, D. C. (2010). Evaluating Scratch to Introduce Younger Schoolchildren to Programming. *PPIG* (s. 7).

Wing, J. M. (2006). Computational thinking. *Communications of the Acm, 49*(3), 33-35. https://doi.org/Doi 10.1145/1118178.1118215

Wohl, B., Porter, B. & Clinch, S. (2015). Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing. *Proceedings of the Workshop in Primary and Secondary Computing Education* (s. 55-60).

Yackel, E. & Hanna, G. (2003). Reasoning and proof. I J. Kilpatrick, W. G. Martin & D. Schifter (Red.), *A research companion to principles and standards for school mathematics* (s. 227-236). Reston: NCTM.

Yin, R. K. (2009). *Case Study Research: Design and Methods* SAGE Publications.

Vedlegg

Vedlegg 1: Introduksjonsøkten

Vedlegg 2: Programmeringsøkten

Vedlegg 3: Intervjuguide

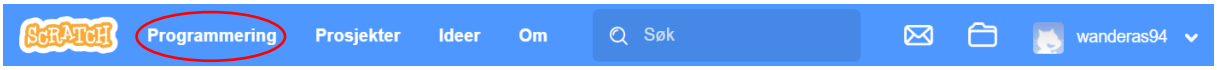
Vedlegg 4: Informasjonsskriv og samtykke

PROGRAMMERING I MATEMATIKK

VELKOMMEN TIL SCRATCH

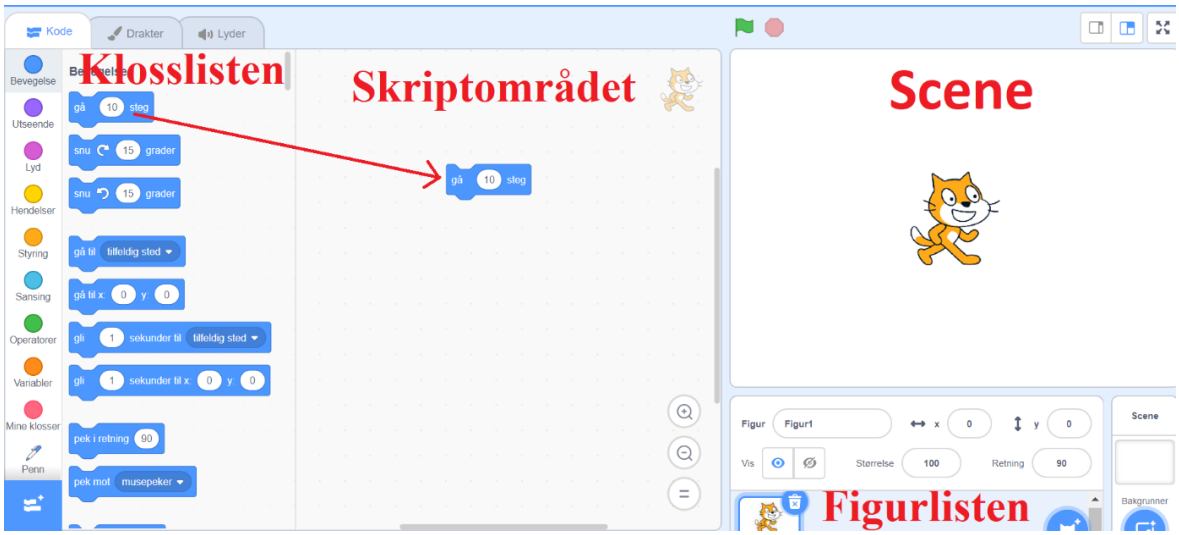
Dette er et introduksjonshefte til programmet Scratch. Heftet er laget slik at det presenteres en ny funksjon og deretter kommer det oppgaver slik at du får testet denne funksjonen. Det er lurt å følge rekkefølgen og prøve alle funksjonene. Det første dere må gjøre er å lage en bruker:

- Gå til nettsiden: https://scratch.mit.edu/
- Lag deg en bruker
- Start et nytt prosjekt ved å trykke på *programmering* på toppen av siden



- Sjekk at språket er satt til «Norsk bokmål». Dette kan du endre ved å trykke på jordkloden øverst til venstre.

- Nå er du klar til å programmere. Scratch-katten Felix er allerede plassert midt på scenen.
- Ulike klosser har ulike farger, og alle bevegelsesklossene mørkeblå. For å programmere Felix til å gå 10 steg, tar vi tak i «gå 10 steg» klossen fra klosslisten og slipper klossen på skriptområdet. Du kan når utføre handlingen ved å trykke på knappen med musepekeren.



OPPGAVE 1

a. Det første du skal gjøre er å endre bakgrunn på scenen.

b. Du skal deretter programmere Felix til å gå 100 steg, snu 90 grader og gå 100 steg.

FUNKSJON – START PROGRAM

- Under Hendelser på klosslisten finner du klossen «når det grønne flagget klikkes», denne kan brukes for å starte programmet du har laget.

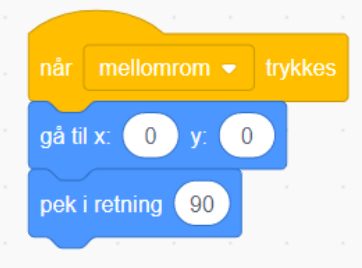
OPPGAVE 2

a. Hvor mange ganger må Felix gjøre det fra oppgave 1 før han ender på samme plass som han startet?

b. Programmer Felix til å gjøre det når du trykker på det grønne flagget.

FUNKSJON – START MIDT PÅ SCENEN

- Med denne blokken med klosser starter Felix midt på scenen dersom du trykker på mellomrom.
- Dersom Felix peker i retning 90, vil det si mot høyre. Prøv å endre retningen for å se hva som skjer.
- Hvorfor tror du figuren starter midt på scenen når x og y er satt til 0?



OPPGAVE 3

Du skal nå programmere Felix til å:

a. Starte midt på scenen når du trykker på en valgfri tast

b. Si noe i 3 sekunder

c. Gå 50 steg, snu 90 grader, vent 1 sek, gå 50 steg, snu 90 grader, vent 1 sek, gå 50 steg, snu 90 grader, vent 1 sek, gå 50 steg, snu 90 gradet, vent 1 sek. Hvor ender Felix?

FUNKSJON – GJENTA KLOSSER

- Noen ganger kan kodene bli så lange at vi ønsker å programmere det på en kortere og enklere måte. Da kan vi bruke klosser som heter «gjenta». Der kan man bestemme hvor mange ganger noe skal gjenta seg eller gjenta noe for alltid.

OPPGAVE 4

Bruk «gjenta» klossen for å gjøre koden i oppgave 3c kortere.

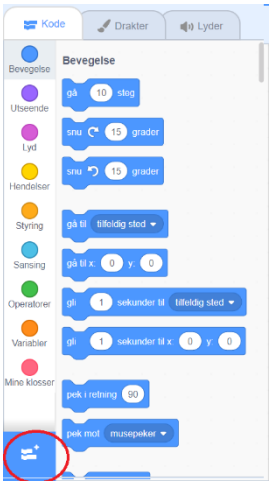
- Hvor mange ganger må du gjenta koden?
- Hva skjer dersom du bruker klossen «gjenta for alltid»?

FUNKSJON - PENN

- Dersom vi vil tegne hvor Felix har gått kan vi bruke en tilleggsfunksjon som heter Penn. Den finner vi dersom vi trykker på knappen ned til venstre, se på bildet.

- Bruk «Penn på» for å tegne og «slett alt» for å fjerne det du har tegnet med pennen.

- Bruk programmet du laget i oppgave 4 og legg til pennen for å tegne hvordan Felix beveger seg.

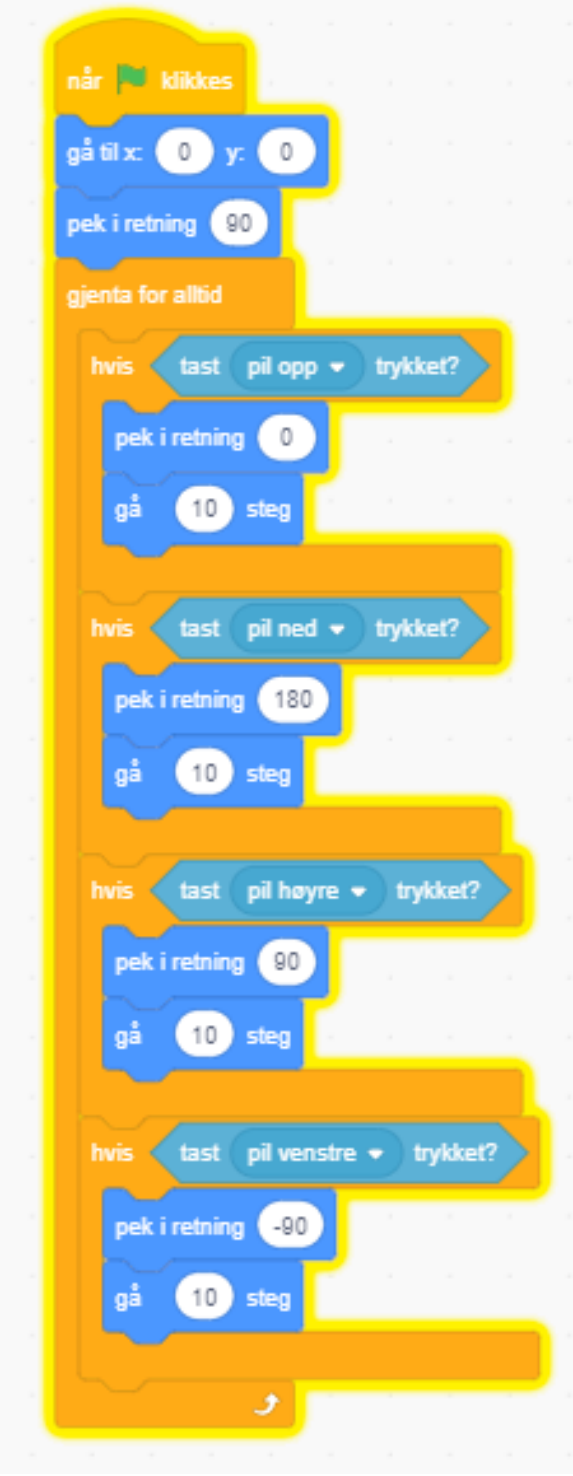


OPPGAVE 5

Programmer Felix til å tegne en valgfri tegning. Her kan du gjerne eksperimente med å bruke ulik bredde og farger på pennen.

LAG DITT EGET SPILL ELLER PROGRAM

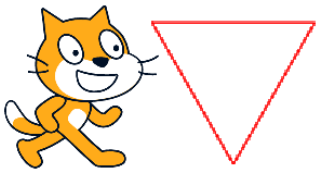
Bildet til høyre viser et forslag til hvordan du kan starte for å lage et spill i Scratch. Du kan bruke denne som utgangspunkt for å videreutvikle ditt eget spill eller program. Hva tror du denne koden gjør?



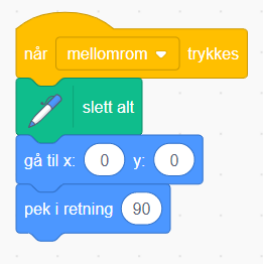
Skriv navnet til begge på gruppen her:

PROGRAMMERING I MATEMATIKK

GEOMETRI I SCRATCH



- Du skal nå bruke de ulike funksjonene vi lærte forrige gang for å programmere geometriske figurer.
- For å kunne se figurene må vi bruke pennen når vi programmerer.
- Et lurt hjelpemiddel å ha når man programmerer figurer er å kunne slette alt som er tegnet med pennen når man trykker på mellomrom, som vist på bilde til høyre.



OPPGAVE 1 – PROGRAMMER FIGURER

I denne oppgaven skal du også fylle inn svar i tabellene på arket. Du skal nå programmere Felix til å tegne:

- Kvadrat
- Rektangel
- Likesidet trekant

a. Beskriv egenskapene til de tre figurene i tabellen under og diskuter hva som er forskjellen og likhetene deres.

Kvadrat	Rektangel	Likesidet trekant

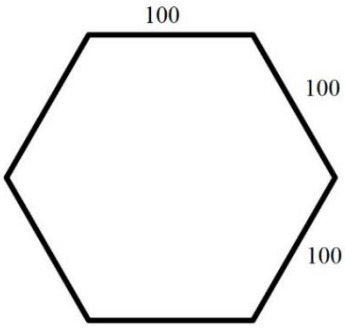
b. Hva er forskjellen og likheter mellom selve kodene dere har laget til de ulike figurene?

Forskjeller	Likheter

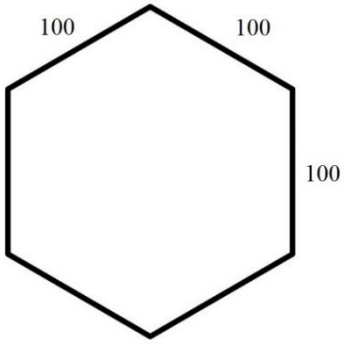
OPPGAVE 2 – BRUK FÆRRE KLOSSER

Programmere Felix til å tegne figurene med så få klosser som mulig:

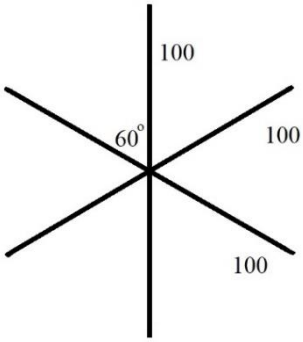
1.



2.



3.



a. Hvilke figurer er dette?
b. Forklar sammenhengen mellom de tre ulike figurene. Er det noen forskjeller og likheter?
c. Diskuter og forklar hvorfor det er akkurat den måten du har gjort det på som bruker så få klosser som mulig.

OPPGAVE 3 – FEILSØKING AV KODE

Når man lager ulike programmer kan det fort bli en feil i koden, og da må det ofte en del detektivarbeid til. Dette kalles å feilsøke. Oppgaven din er å endre koden på bildet slik at den tegner en figur som er veldig lik en sirkel, som vises på det andre bildet.

a. Hvilken figur tegner koden slik den er nå?
b. Endre koden slik at den tegner en figur som er så lik en sirkel som du klarer. Prøv i Scratch.

OPPGAVE 4 – ÅPEN OPPGAVE

Fortsett på spillet du startet på forrige uke. Du skal nå forsøke å bruke ulike geometriske figurer i spillet.

Her kan du være kreativ og benytte deg av de ulike klossene og figurene vi tidligere har programmert!

Intervjuguide

1. Hva synes dere om at alle skal programmere i skolen fra neste år (2020)?

2. Har dere prøvd programmering før?
    - Har du brukt Scratch eller noen andre programmer/språk før?
    - I hvilken sammenheng brukte du det?
    - Hvor ofte har du gjort det?

3. Bruker dere kunnskap dere har fra før av når dere løser oppgaver i programmering?
    - Synes du geometrioppgavene ble vanskeligere eller enklere ved bruk av programmering? Hvorfor?
    - Prøvde du å lage noe som ikke fungerte?
    - Hva gjorde du for å endre det?

4. Hva lærte du om programmering i dag?

5. Hvilke egenskaper har en likesidet trekant?
    - Hvordan lærte du det?
    - Hvor lange er sidene?
    - Hvor stor er vinklene?
    - Kan du formulere en definisjon for likesidet trekant?

6. Hva er forskjellen på et kvadrat og et rektangel?
    - Hvor lange er sidene?
    - Hvor stor er vinklene?
    - Kan du formulere en definisjon for kvadrat og rektangel?
    - Er alle kvadrat egentlig rektangler? Hvorfor eller hvorfor ikke?

7. Hva er sammenhengen mellom de tre figurene i «Oppgave 2»?
    - Hvilke figurer er det som er programmert?
    - Hva er forskjellen på figur 1og 2?
    - Hvilken figur får du dersom du setter sammen 2 og 3? (setter 2 inn i 3)
    - Prøv å formulere en definisjon for regulære sekskanter?

8. Hvilke egenskaper har en sirkel?
    - Var det du laget i Scratch en sirkel? Hvorfor/hvorfor ikke?
    - Kan du formulere en definisjon for sirkel?

9. Hvordan gikk dere frem for å løse oppgavene?
    - Hva var det første dere gjorde når dere skulle løse oppgaven?
    - Brukte dere en bestemt fremgangsmåte eller strategi?
    - Finnes det flere ulike måter å programmere den samme figuren på?
    - Prøver du vanligvis å finne fremgangsmåter selv eller gjør du slik læreren viser i timen?
    - Prøve dere å tegne noen av figurene på arket først?

10. Hvilken fremgangsmåte brukte du for å løse oppgave 2?
    - Var det en bevisst strategi å gjøre det slik?
    - Endret du fremgangsmåte underveis?
    - Måtte du prøve flere ganger før du klarte det?

11. Hva er det første du gjør når du får en oppgave?
    - Prøver du ulike måter å løse oppgaven på dersom du ikke klarer å løse den med en gang?
    - Hva gjør du dersom du ikke klarer å løse den med en gang?

12. Hva gjør du når du har funnet ut at svaret er riktig?
    - Tenker du om svaret er logisk?
    - Setter du prøve på svaret?
    - Prøver du også å løse det på andre måter?

Vil du delta i forskningsprosjektet

"Programmering i matematikk"?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å undersøke hvordan elever arbeider med geometri i programmering. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
I den nye læreplanen som gjelder fra 2020 vil programmering være en sentral del innenfor matematikkfaget. Dette prosjektet er en masteroppgave i matematikkdidaktikk som går ut på å bruke programmering for å konstruere ulike geometriske figurer.

> Kompetansemål etter 6. trinn i ny læreplan (gjelder fra 2020): *bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønster.*

Forskningsspørsmålet jeg skal undersøke er: *«Hvilke strategier benytter elever på 6. trinn seg av i arbeid med programmering av geometriske figurer?»*

Hvem er ansvarlig for forskningsprosjektet?
NTNU er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?

Jeg ønsker å undersøke hvordan elever på 6. trinn arbeider med programmering for å utforske geometriske figurer, jeg trengte derfor et utvalg elever og kontaktet din matematikklærer for å få gjennomføre undersøkelsen min i deres klasse.

Hva innebærer det for deg å delta?

Dersom du velger å delta i dette prosjektet, innebærer det at jeg kan gjennomføre intervju, lyd- og skjermopptak av utvalgte elever i klassen. Personopplysninger om elev og skole vil bli anonymisert. Dersom foreldre ønsker å se intervjuguide etc. på forhånd kan du ta kontakt med meg.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykke tilbake uten å oppgi noen grunn. Alle opplysninger om deg vil da bli anonymisert. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

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. Navnet og kontaktopplysningene dine vil bli erstattet med en kode som lagres på egen navneliste adskilt fra øvrige data, datamaterialet vil bli kryptert og oppbevart innelåst.

Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Prosjektet skal etter planen avsluttes 16.05.20. Personopplysninger og lydopptak blir slettet og det anonymiserte datamateriale vil kun være tilgjengelig for meg og veileder for prosjektet.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:
- innsyn i hvilke personopplysninger som er registrert om deg,
- å få rettet personopplysninger om deg,
- få slettet personopplysninger om deg,
- få utlevert en kopi av dine personopplysninger (dataportabilitet), og
- å sende klage til personvernombudet eller Datatilsynet om behandlingen av dine personopplysninger.

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

På oppdrag fra NTNU har NSD – Norsk senter for forskningsdata AS vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Hvor kan jeg finne ut mer?
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:
- Ulrik Haugen Wanderås (Ulrikhw@ntnu.no)
- *NTNU* ved Iveta Kohanova (Iveta.kohanova@ntnu.no)
- NSD – Norsk senter for forskningsdata AS, på epost (personverntjenester@nsd.no) eller telefon: 55 58 21 17.

Med vennlig hilsen

Iveta Kohanova	Ulrik Haugen Wanderås
Prosjektansvarlig (Forsker/veileder)	*Student*

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet *«Programmering i matematikk»,* og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i *intervju*
- ☐ å delta i *lydopptak*
- ☐ å delta i *skjemopptak*

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet, *16.05.2020.*

(Signert av prosjektdeltaker, dato)