Programmering i matematikkfaget

En kasusstudie av elevers arbeid med programmeringsspråket Scratch.

CHRISTIAN ORE

VEILEDER
Anders Skarpeteig Fidje

Universitetet i Agder, 2022
Fakultet for teknologi og realfag
Institutt for matematiske fag

Forord

Denne masteroppgaven markerer siste ledd i fullførelsen av min erfaringsbaserte mastergrad i matematikkdidaktikk ved Universitetet i Agder. De siste tre årene har jeg studert på deltid, parallelt med jobb som ungdomsskolelærer. Det har vært krevende, men jeg har fått utvikle meg selv, både akademisk og som praktiserende lærer. Jeg håper masteroppgaven min kan være til inspirasjon for andre i arbeidet med å ta i bruk programmering i matematikkundervisningen.

Takk til Anders Skarpeteig Fidje for en humørfylt og konstruktiv veiledning. Du har vært en trygg sparringspartner, og holdt meg på stø kurs gjennom hele prosessen.

Takk til alle elevene som har vært villig til å bli observert gjennom lyd- og skjermopptak, og med det bidratt til datamaterialet som danner grunnlag for denne studien.

Takk til Kjell og Ole Magnus, ledelsen ved Lista ungdomsskole, for støtte og tilrettelegging av arbeidsmengde i perioder der studien har krevd det.

Takk til Petter Salvesen for korrekturlesing av den engelske versjonen av sammendraget. Jeg lover å huske på apostrofen bak genetivs-s i fremtiden.

Takk til min kjære kone, Camilla. Uten din støtte hadde ikke dette vært mulig. Vi har en hektisk hverdag med to små, og krevende jobber. Du har vist forståelse gjennom hele prosessen, og lagt til rette for at jeg har kunne fullføre mastergraden.

Christian Ore
Farsund, 15. mai 2022

Sammendrag

Tema for denne masteroppgaven er programmering i matematikkfaget. Programmeringen har fått betydelig plass i faget etter at Fagfornyelsen trådte i kraft som ny læreplan høsten 2020. Dette medfører faglige og didaktiske utfordringer for hvordan programmering kan implementeres i matematikkundervisningen på en hensiktsmessig måte.

Oppgaven beskriver en kasusstudie rettet mot elever sitt arbeid med programmering i matematikk. Datamaterialet analyseres med en deduktiv tilnærming av relevant teori og rammeverk. Instrumentell genesis og læreplanens kjerneelementer er benyttet som utgangspunkt i forskningen. Studien tar for seg to forskningsspørsmål.

1. **Hvordan kan elevenes arbeid med programmering forstås i lys av instrumentell genesis?**
2. **På hvilken måte kan kjerneelementene komme til syne gjennom elevers arbeid med programmering?**

Det er utviklet et oppgavehefte med seks oppgaver. Datagrunnlaget for studien baserer seg på elevenes arbeid med to av oppgavene i heftet. Skjerm- og lydopptak er benyttet som datainnsamlingsmetode. Studiens analyse- og drøftingsdel synliggjør prosessene elever gjennomgår i arbeid med oppgavene. Funn som diskuteres er blant annet:

- Hvordan programmet og elevene påvirkes av hverandre.
- Konsekvensen av ubalanse mellom elevenes matematikk- og programmeringsforståelse.
- Muligheter og begrensninger i programmeringsspråket.

Det konkluderes med at instrumentell genesis er et hensiktsmessig rammeverk for å forstå elevenes arbeid med programmering. Komponentene i skjemautviklingen fungerer som et godt verktøy for å systematisere og forstå prosessene i elevenes arbeid.

Kjerneelementene er i stor grad representert i elevenes arbeid med programmering. Dette støtter opp om at programmering er et hensiktsmessig verktøy til bruk i matematikkfaget og at elevene arbeider på en måte som er i tråd med læreplanen.

Abstract

The theme for this master's thesis is programming in mathematics education. Programming has become a significant part of mathematics education in the new curriculum of the Norwegian primary and secondary school. This adds new academic and didactic challenges of how programming can be implemented in mathematics teaching in an appropriate manner.

This thesis describes a case study aimed at pupils' work with programming in mathematics. Collected data is analyzed with a deductive approach based on instrumental genesis and the core elements from the new curriculum. The study addresses two research questions.

1. How can pupils' work with programming be understood in the light of instrumental genesis?

2. In what way can the core elements become apparent through pupils' work with programming?

A set of exercises has been developed for this case study. The analysis is based on data from pupils' work with two of these exercises. Screen and audio recording is the method used to collect data. The analysis and discussion of the study seeks knowledge of the process pupils encounter during their work on the exercises. Some of the findings discussed in this thesis include:

- How the program and the pupils are affected by each other.
- The consequence of an imbalance between pupils' understanding of mathematics and programming.
- Constraints and possibilies of the programming language.

This thesis concludes that instrumental genesis is an appropriate framework for understanding how pupils approach programming. The components of utilization scheme functions as an appropriate tool for systematizing and understanding the processes in students work.

The core elements are largely represented in the pupils' work with programming. This supports that programming is an appropriate tool for use in mathematics education.

Innhold

1 Innledning

1.1 Bakgrunn for valg av tema

Denne masteroppgaven har som mål å undersøke elevers arbeid med programmering i matematikkfaget. Bakgrunnen for dette arbeidet er fullførelse av en erfaringsbasert masterutdanning innen matematikkdidaktikk. Studiet er en del av lærerspesialistordningen og er gjennomført på deltid, parallelt med jobb som realfagslærer ved en ungdomsskole. Utgangspunktet for valg av tema er ny læreplan fra høsten 2020 (Utdanningsdirektoratet, 2020). Programmering ble da innført som en obligatorisk del av matematikkfaget. Fra 4.-10. trinn, samt videregående skole, er programmering nå en del av elevenes kompetansemål. Frem mot dagens læreplan har det vært flere nasjonale utredninger (NOU 2013:2, 2013; NOU 2015:8, 2015) som har pekt på behovet for at programmering burde ha en plass i skolen, enten som eget fag eller som en del av eksisterende fag. Argumenter for programmering i skolen knyttes gjerne til nødvendige ferdigheter i dagens, og fremtidens, samfunn (Sevik, 2016). I læreplanen beskriver Utdanningsdirektoratet (2020) programmering som et verktøy for å utforske og løse matematiske problem.

1.2 Forskningsspørsmål

For lærere i ungdomsskolen fører dette med seg faglige og didaktiske utfordringer for hvordan programmering kan implementeres i matematikkundervisningen på en hensiktsmessig måte. I denne studien vil det undersøkes hvordan elevene jobber med arbeidsoppgaver som kombinerer programmering og matematikk. Datamatrialet fra dette arbeidet vil analyseres opp mot et etablert teoretisk rammeverk. Instrumentell genesis er valgt som rammeverk for dette arbeidet. Det kan bidra til å synliggjør prosessene i elevenes arbeid med programmering og hvordan elevene og programmeringsspråket påvirkes av hverandre.

Parallelt med dette vil resultat fra observasjonene også analyseres med utgangspunkt i kjerneelementene i læreplanen. Dette baseres på et ønske om å se at, og på hvilken måte, læreplanens overordnede intensjoner oppfylles gjennom arbeid med programmering. En

tydelig tilstedeværelse av kjerneelementene vil være en indikasjon på, og et argument for, at programmering er et hensiktsmessig verktøy til bruk i matematikkfaget.
Det er med dette utgangspunktet utarbeidet to forskningsspørsmål for denne studien:

1. **Hvordan kan elevenes arbeid med programmering forstås i lys av instrumentell genesis?**
2. **På hvilken måte kan kjerneelementene komme til syne gjennom elevers arbeid med programmering?**

1.3 Struktur av oppgaven

Oppgaven vil videre presenteres gjennom fem kapitler.
I kapittel 2 presenteres teori og rammeverk som er relevant for denne studien. Begreper og oversettelser brukt i videre presentasjon av studien vil beskrives i dette kapitlet. Kjerneelementene presenteres med utgangspunkt i læreplanens definisjoner. Videre vil programmering belyses og et utvalg programmeringsspråk trekkes frem. Avslutningsvis vil instrumentell genesis presenteres med utgangspunkt i instrumentell tilnærming.

I kapittel 3 beskrives forskningsmetoden brukt i denne studien. Dette kapitlet beskriver videre grunnlaget for valg av respondenter, fremgangsmåte for datainnsamling, argumentasjon for at kvaliteten på forskningen er ivaretatt og etiske betraktninger. Kapitlet avsluttes med en presentasjon av hvordan instrumentell genesis og kjerneelementene er operasjonalisert for bruk i denne studien.

I kapittel 4 presenteres resultatene fra observasjonene. Resultatene analyseres med utgangspunkt i operasjonaliseringen presentert i kapittel 3.

I kapittel 5 diskuteres utvalgte funn som er avdekket gjennom analysearbeidet. Studien vil her sees i sammenheng med annen relevant forskning.

Kapittel 6 avrunder presentasjonen med en oppsummering og konklusjon, samt en vurdering av pedagogiske og didaktiske implikasjoner som har kommet frem under denne forskningen.

2 Teoretisk forankring

2.1 Kjerneelementene

Læreplanen er styrende for hvordan lærere skal legge opp undervisningen i faget. Temaet for denne studien er et resultat av læreplanens inkludering av programmering som verktøy i matematikkundervisningen. Utdanningsdirektoratet (2020) innførte ny læreplan, LK20, i norsk skole høsten 2020. LK20 markerer en kursendring for hvordan undervisningen i skolen skal praktiseres. Kjerneelementer innføres i alle fag og beskriver de overordnete kompetansene elevene skal tilegne seg på en mer detaljert og eksplisitt måte enn i forrige læreplan (Utdanningsdirektoratet, 2013). I matematikkfaget innføres kjerneelementene;

- utforskning og problemløsning
- modellering og anvendelse
- resonnering og argumentasjon
- representasjoner og kommunikasjon
- abstraksjon og generalisering
- matematiske kunnskapsområder

Det vil videre utdypes hva Utdanningsdirektoratet (2020) legger i disse begrepene.

Utforskning og problemløsning

Utforsking handler om at elevene skal utfordres på å lete etter mønster, finne sammenhenger og diskutere seg frem til en felles forståelse, der strategier og fremgangsmåter skal vektlegges. *Problemløsning* handler om at elevene utvikler metoder for å løse utfordringer de ikke kjenner fra før. Utdanningsdirektoratet (2020) introduserer i denne sammenheng algoritmisk tenkning som nytt begrep i læreplanen der elevene skal lære strategier ved å bryte problemer ned i delproblemer som løses systematisk. I forbindelse med LK20 har Utdanningsdirektoratet (2019) publisert en artikkel der de utdyper hva som legges til grunn for algoritmisk tenkning, og hvordan dette skal forstås.

Algoritmisk tenkning

Algoritmisk tenkning handler om at elevene lærer seg strategier og fremgangsmåter for å kunne løse komplekse utfordringer ved å bryte dem ned til håndterlige deler (Utdanningsdirektoratet, 2019).

Algoritmisk tenkning er den norske oversettelsen av det engelske begreper computational thinking. Wing (2006) beskriver computational thinking som evnen til å «(…) reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.» (Wing, 2006, s. 33). Hun beskriver videre computational thinking som en grunnleggende kompetanse og mener den bør likestilles med lesing, skriving og aritmetikk i undervisningen.

Modellering og anvendelse

Modellering i matematikk handler om å beskrive virkeligheten gjennom et matematisk språk. Utdanningsdirektoratet (2020) legger til grunn at elever skal ha innsikt i hvordan modeller kan brukes til å beskrive aspekter ved dagliglivet, arbeidslivet og samfunnet ellers.

Anvendelse beskrives av Utdanningsdirektoratet (2020) ved at elevene skal få innsikt i hvordan de kan bruke matematikk i ulike situasjoner.

Resonnering og argumentasjon

Utdanningsdirektoratet (2020) definerer *resonnering* som evnen til å kunne følge, vurdere og forstå matematiske tankerekker. Dette innebærer at elever skal ha en forståelse av at matematiske regler og resultater ikke er tilfeldige, men forholder seg til faste rammer.

Argumenter beskrives av Utdanningsdirektoratet (2020) ved at elever skal kunne begrunne gyldigheten av fremgangsmåter, resonnementer og løsninger.

Representasjoner og kommunikasjon

Representasjoner i matematikk innebærer måter matematikken kan uttrykkes på. «Representasjoner kan være konkrete, kontekstuelle, visuelle, verbale og symbolske.» (Utdanningsdirektoratet, 2020, s. 3).

Kommunikasjon beskrives av Utdanningsdirektoratet (2020) som elevenes evne til å bruke matematisk språk i samtaler, argumentasjon og resonnering, og at elevene må få mulighet til

å bruke matematiske representasjoner i ulike sammenhenger gjennom egne erfaringer og samtaler.

Abstraksjon og generalisering

Abstraksjon innebære at elevene utvikler en formalisering av tanker, strategier og matematisk språk. Utdanningsdirektoratet (2020) utdyper at denne utviklingen går fra konkrete beskrivelser til formelt symbolspråk og formelle resonnementer.
Elevene skal gjennom *generalisering* oppdage sammenhenger og strukturer i arbeidet med faget. «Det vil si at elevene kan utforske tall, utregninger og figurer for å finne sammenhenger og deretter formalisere ved å bruke algebra og hensiktsmessige representasjoner.» (Utdanningsdirektoratet, 2020, s. 3)

Matematiske kunnskapsområder

De matematiske kunnskapsområdene består av *tall og tallforståelse*, *algebra*, *funksjoner*, *geometri*, *statistikk* og *sannsynlighet*. Utdanningsdirektoratet (2020) argumenterer videre for viktigheten av at elevene i grunnskolen utvikler kompetanse på disse områdene. De matematiske kunnskapsområdene som i LK20 defineres som et kjerneelement, er i all hovedsak en videreføring av det som forrige læreplan beskrev som *hovedområde* (Utdanningsdirektoratet, 2013).

2.2 Programmering

LK20 introduserte programmering som arbeidsmetode og verktøy i matematikkfaget (Utdanningsdirektoratet, 2020). Fra 4. til 10. trinn er det innført konkrete kompetansemål som legger føringer for at elevene skal jobbe med programmering. Fra å ikke være nevnt i forrige læreplan (Utdanningsdirektoratet, 2013) er begrepet programmering nå spesifikt nevnt ni ganger i LK20 for matematikkfaget. Programmering har i mange år vært en mulig innfallsvinkel i matematikkundervisningen. Med ny læreplan er det nå blitt en obligatorisk del som har fått et relativt stort fotavtrykk i læreplanen. Det er gått fra å være noe enkelte, spesielt interesserte, lærere har brukt som et didaktisk verktøy til å bli noe alle matematikklærere må forholde seg til, og ta i bruk, i undervisningen.

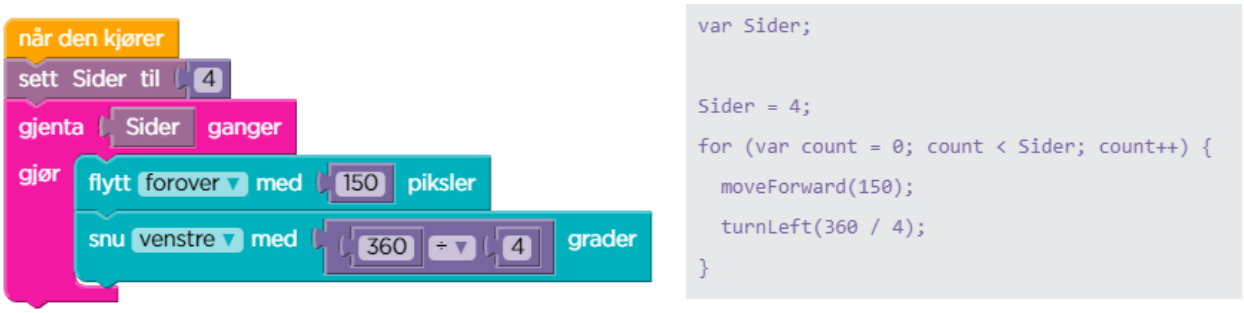
I perioden mellom utgivelsen av forrige læreplan og LK20 har flere utvalg pek på viktigheten av å øke den digitale kompetansen. Digitutvalget (NOU 2013:2, 2013) konkluderte med at innsatsen for å lære barn og unge digitale ferdigheter må endres og styrkes. De mente det måtte legges til rett for at barn skal få en dypere forståelse, som strekker seg utover det å bare være en digital forbruker.

> For å sikre digital verdiskaping i fremtiden er vi nødt til å legge til rette for at barn og unge ikke kun er i stand til å bruke, men også skape digitalt innhold og digitale tjenester. I dag har barn få muligheter til å lære seg programmering og programvareutvikling. (NOU 2013:2, 2013, s. 10).

Digitutvalget foreslo videre å innføre programmering som valgfag i ungdomsskolen, noe som også ble en realitet noen år senere (NOU 2013:2, 2013). Sanneutvalget (Sanne et al., 2016) foreslo i sin rapport at teknologi og programmering burde opprettes som et eget, og obligatorisk, fag i grunnskolen. Sevik (2016) presenterer *Programmering for skolen*, et notat fra Senteret for IKT i utdanningen. I dette dokumentet foreslås det at programmering får en plass i norsk skole. Det foreslås muligheter for hvordan dette kan gjøres; som eget IKT-fag, integreres i eksisterende fag og som fagovergripende kompetanse i flere fag. I dag er programmering en del av undervisningen gjennom eget valgfag og som en integrering i de eksisterende fagene matematikk, naturfag, musikk og kunst & håndverk.

Programmeringsspråk

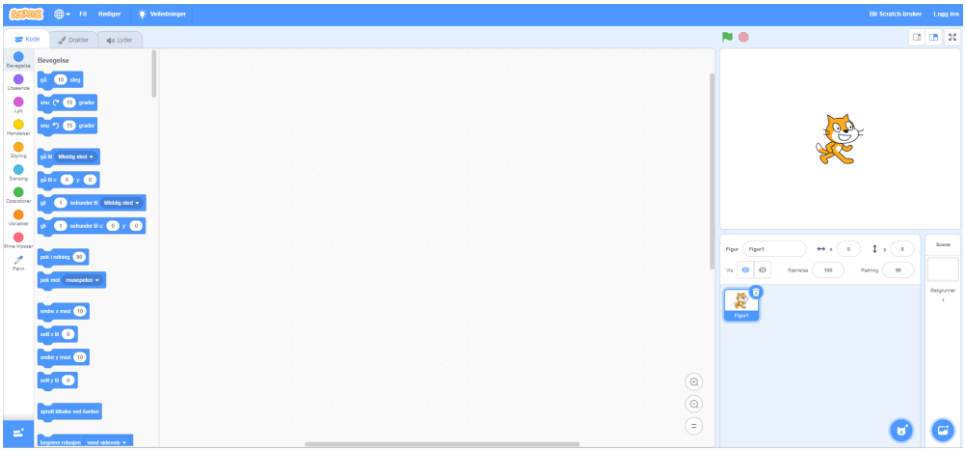
Det har blitt utviklet en rekke programmeringsspråk. Hovedsakelig deles de inn i tekst- og blokkbaserte språk. I tekstbaserte programmeringsspråk skrives alle kodene som tekstlinjer. Eksempler på slike programmeringsspråk er Python og JavaScript. I blokkbaserte programmeringsspråk er kodene skjult bak grafiske blokker. Disse blokkene settes i en rekkefølge ut fra hvilke oppgaver en ønsker at programmet skal utføre. Eksempler på blokkbaserte programmeringsspråk er Scratch, code.org og micro:bit. Figur 1 viser to program med samme innhold, skrevet ved blokkprogrammering i code.org og ved tekstprogrammering i JavaScript.



Figur 1: Skjermdump av koder fra code.org (venstre) og JavaScript (høyre).

Tekstbaserte programmeringsspråk anses å ha en høyere terskel for bruk enn blokkprogrammering grunnet strenge krav til syntaks i kodene (Flø, 2021). LK20 legger heller ikke føringer for at det må brukes tekstprogrammering i ungdomsskolen. Dette bidrar til at blokkprogrammering kan være en hensiktsmessig inngang til programmeringen for elever i ungdomsskolen.

I forbindelse med denne studien har elevene jobbet med programmeringsspråket Scratch (https://scratch.mit.edu). Dette er et gratisprogram der elevene programmerer direkte i nettleseren.



Figur 2: Skjermdump fra Scratch

Begrepsavklaring

Begrepene *programmering* og *koding* er relevante når en skal beskrive tema for denne studien. I denne oppgaven defineres, og brukes, begrepet koding for de konkrete handlingene der elevene setter sammen blokkene, og definerer blokkenes verdier. Programmering er et begrep som i større grad beskriver hele prosessen med å lage et program, både konkret og mentalt, for at kodene skal fungere sammen.

2.3 Instrumentell tilnærming

LK20 sin innføring av programmering leder undervisningspraksisen inn på et område det er relativt lite forskning på. Denne studien ønsker derfor å se nærmere på samspillet mellom programmering og matematikk når elevene jobber med programmering i matematikkfaget. For å legge til rette for et hensiktsmessig utgangspunkt for analysen vil datamaterialet sees i sammenheng med et eksisterende rammeverk, instrumentell tilnærming. Verillon og Rabardel (1995) beskriver denne tilnærmingen der et individ (subjekt) lærere å bruke et verktøy (artefakt) gjennom å utforske dets muligheter og begrensninger. Verktøy kan være både fysiske og konkrete, som en hammer, regneark og kalkulator, men også mentale, som språk og matematiske formler. Målet er at subjektet, gjennom å undersøke de muligheter og begrensninger som ligger i verktøyet, skal utvikle det til å bli et instrument. De beskriver skillet mellom verktøy og et instrument på denne måten.

> But it is important to stress the difference between two concepts, the artifact, as a man-made material object, and the instrument, as a psychological construct. The point is that no instrument exists in itself. A machine or a technical system does not immediately constitute a tool for the subject. (Verillon & Rabardel, 1995, s. 84)

Instrumentet er utviklet fra et verktøy gjennom dannelse og utvikling av anvedelsesskjema (*utilization schemes*) (Verillon & Rabardel, 1995). Anvedelsesskjema er mentale skjemaer som utvikles gjennom en gjensidig påvirkning mellom verktøyet og subjektet. Subjektet lærer verktøyet og kjenne, samtidig som verktøyets iboende egenskaper vil påvirke subjektets utvikling. Gueudet og Trouche (2009) beskriver denne prosessen som en likning:

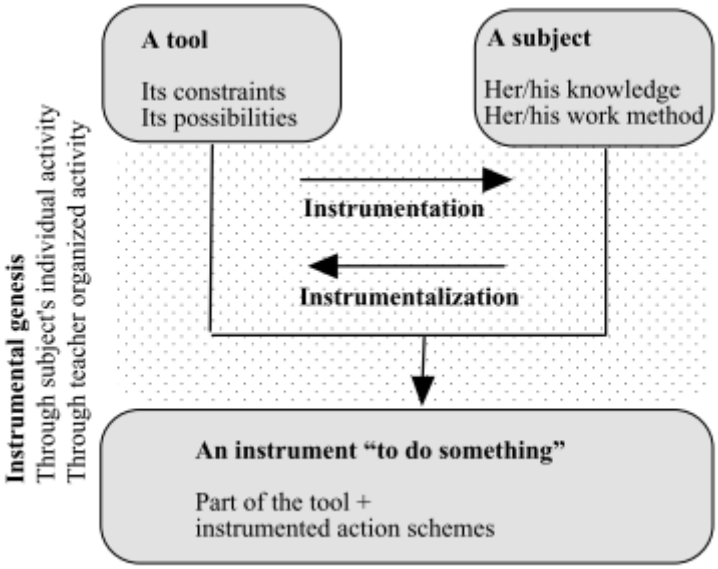
> *Instrument = artefakt + Anvedelseskjema* (Gueudet & Trouche, 2009, s. 204)

Idéen om instrumentell tilnærming er gjeldende i alle deler av livet og samfunnet. I denne studien er det brukt som rammeverk for å utforske elevers bruk av programmering i matematikkfaget. Gjennom prøving og feiling utforskes programmeringsspråket slik at elevene danner skjemaer for hvordan de i økende grad kan bruke programmering som et verktøy. Dette skjer ved å utforske de muligheter og begrensninger som ligger i programmeringsspråket.

Instrumentell tilnærming deles inn i to deler; instrumentell orkestrering og instrumentell genesis. Instrumentell orkestrering omfatter hvordan lærere kan bruke, og legge til rette for elevenes bruk av, verktøy i undervisningen. (Drijvers & Trouche, 2008) I denne studien undersøkes elevsiden av læringsprosessene og instrumentell genesis vil derfor være det rammeverket som benyttes.

2.3.1 Instrumentell genesis

Instrumentell genesis omfatter prosessen der elevene (subjetet) omdanner verktøy, ellet et artefakt, til et instrument. Trouche (2004) synliggjør disse prosessene skjematisk i sin artikkel.



Figur 3: Skjematisk fremstilling av instrumentell genesis (Trouche, 2004).

Dette rammeverket hjelper til med å tolke og beskrive prosessene i elevenes arbeid og læring i møte med teknologi. Verktøyet i denne studien er programmeringsspråket Scratch. Trouche (2005) beskriver et verktøy, eller et artefakt, som et objekt som kan utføre en handling. I arbeidet med et verktøy vil elevene utvikle en forståelse av dets begrensninger og muligheter, og gradvis kunne gjøre det til et instrument. Denne prosessen er tosidig ved at verktøyet påvirker subjektet og subjektet påvirker verktøyet. Disse prosessene kalles instrumentering (*instrumentation*) og instrumentalisering (*instrumentalization*). (Drijvers & Trouche, 2008; Gueudet & Trouche, 2009; Rabardel, 2002; Trouche, 2005). «Instrumental genesis is a process (therefore needs time) and has two components, the first one

(instrumentalization) directed toward the artifact, the second one (instrumentation) directed toward the subject.» (Trouche, 2005, s. 145)

Ved instrumentalisering bruker eleven sin forståelse og kunnskap til å påvirke verktøyet. Instrumentalisering handler dermed om hvordan eleven anvender sin fagkunnskap i bruk av verktøyet, og til å utvikle det (Gueudet & Trouche, 2009). Rabardel (2002) beskriver denne prosessen i sin bok som: «Instrumentalization can be defined as process in which the subject enriches the artifact's properties.» (Rabardel, 2002, s. 106). I arbeid med programmeringsspråket Scratch vil instrumentaliseringen komme til syne gjennom prosessene der elevene bruker sin matematikkforståelse til å utvikle forståelse av programmet.

Instrumentering er prosesser knyttet til elevens mentale læring og utvikling gjennom arbeidet med verktøyet. «(…) instrumentation process: the artifact shapes the thinking of the user». (Drijvers & Trouche, 2008, s. 369) I denne studien vil instrumentering handle om hvordan programmering påvirker og bidrar til utvikling av elevenes matematiske forståelse.

Det er utfordrende å observere utviklingen av mentale skjema direkte. Observasjoner blir begrenset til valg og teknikker elevene gjør bruk av i arbeid med verktøyet, og hvordan de beskriver eget arbeid, enten verbalt eller skriftlig. (Drijvers & Trouche, 2008)

Buteau, Gueudet, Muller, Mgombelo og Sacristán (2019) beskriver hvordan skjemaene kan brytes ned i fire komponenter (min oversettelse). Gjennom resten av dokumentet benyttes oversatt versjon av begrepene til Buteau et al. (2019).

1. Mål for aktiviteten (*Goal of the activity*)

   I arbeidet med verktøyet dannes det mål. Målene kan brytes ned til delmål og forventninger til *hva* som kommer til å skje. Det kan være mål gitt av andre i form av oppgavetekst eller egne mål som dannes underveis i arbeidet.

2. Handlingsregler (*Rules of action*)

   Handlingsregler viser til hvilke teknikker og fremgangsmåte som brukes i arbeid med verktøyet. Denne komponenten av skjemautviklingen er knyttet til *hvordan* elevene bruker verktøyet og er dermed et observerbart element.

3. Operasjonelle invarianter (*Operational invariants*)

   Handlingsreglene gjøres med utgangspunkt i et sett med operasjonelle invarianter (Buteau et al., 2019). Operasjonelle invarianter beskriver de mentale prosessene subjektene (elevene) legger til grunn for valg av handlingsregler. Disse har dermed av en mental karakter og ikke like lett å observere. Operasjonelle invarianter deles inn i teoremforståelse (therorem-in-action) og konseptforståelse (concept-in-action). Teoremforståelse kommer til syne når subjektet bruker sin faglige forståelse til grunn for valg av handlingsregel. I denne studien vil teoremforståelse komme til syne der elevene velger handlingsregler basert på sin matematikkforståelse. Konseptforståelse kommer til syne når subjektets handlingsregler drives av en forståelse av de muligheter og begrensninger som ligger i verktøyet, som i denne studien er programmeringsspråket Scratch.

4. Muligheter for slutninger? (*Possibilities of inferences*)

   Gjennom arbeidet med å utvikle handlingsregler og operasjonelle vil subjektet (eleven) gjøre erfaringer som kan overføres til nye situasjoner. «The inferences permit an adaptation to this new situation, and can lead to the emergence of new operational invariants and new rules of action.» (Buteau et al., 2019, s. 1027)

I et forsøk på å besvare første forskningsspørsmål vil observasjonene i denne analyseres med utgangspunkt i disse komponentene.

P-skjema og m-skjema

På bakgrunn av forskningsarbeid med instrumentell genesis utvikler Buteau, Muller, Mgombelo, Sacristán og Dreise (2020) en nivådeling av elevers arbeid med programmering. I denne sammenheng introduserer de begrepene p-skjerma og m-skjema. De tar

utgangspunkt i Assude (2007) sine fire nivåer av instrumentell integrasjon i undervisning (*initiation, exploration, reinforcement og symbiosis)* . Hennes rammeverk er rettet mot læreres bruk av instrumenter i undervisningen og dokumentell genesis. Buteau et al. (2020) tar utgangspunkt i nivådelingen til Assude (2007) og tilpasser den til elevenes læring og instrumentell genesis. «We propose that the instrumental integration model can also be relevant to identify instrumental stages of students' engagement of using programming for complete mathematical investigation projects.» (Buteau et al., 2020, s. 369).
P-skjema indikerer at skjemautviklingen hovedsakelig skjer innen programmering, og m-skjema innen matematikkforståelse. De fire nivåene Buteau et al. (2020) sin modell vil presenteres under.

1. *(guided) programming exercises*.
   Tilsvarer *instrumental initiation* i Assude (2007) sitt rammeverk. På dette nivået er det hovedsakelig utvikling av p-skjema gjennom utforsking av programmeringsspråkets funksjoner og muligheter. Programmering for programmeringens skyld, uten matematisk forankring.

2. *(guided) programming-based, mathematics problems.*
   Instrumental exploration i Assude (2007) sitt rammeverk. Utforskingen av programmet skjer i en matematisk kontekst. Det er fremdeles stor grad av p-skjema, men også noe p+m-skjema, der programmeringsforståelse og matematikkforståelse utvikles sammen.

3. *(guided) authentic mathematical investigation projects.*
   Tilsvarer *instrumental reinforcement* i Assude (2007) sitt rammeverk. Elevene utvikler i større grad p+m-skjema. M-skjema utvikles også i noen grad på dette nivået, for eksempel gjennom tolking av programmets matematiske utdata eller svar.

4. (*self-undertaken) authentic mathematical investigation projects.*
   Instrumental symbiosis i Assude (2007) sitt rammeverk. På dette nivået har elevene utviklet en god forståelse for programmering og dets muligheter og begrensninger. De har gjort programmering til et instrument for matematikk og bruker dette på en

hensiktsmessig måte i arbeidet med å utforske matematikk og løse matematiske utfordringer.

Hensikten med å introdusere p-skjema og m-skjema i denne oppgaven er at det kan bidra til å beskrive operasjonelle invarianter i hovedrammeverket. Dette gjøres ved at p-skjema relateres til konseptforståelse og m-skjema relateres til teoremforståelse.

Utfordringer med instrumentell genesis

Instrumentell genesis er et rammeverk som har vært mye brukt innen forskning på bruk av teknologi i undervisning de siste tjue årene. Se for eksempel forskningen til (Artigue, 2002; Borg, Fahlgren & Ruthven, 2020; Buteau et al., 2019; Gueudet et al., 2022; Trouche, 2004, 2005). Funnen i disse studiene er i stor grad rettet mot isolerte hendelser i form av kasusstudier. Instrumentell genesis bidrar med et begrepsapparat som oppleves relevant og hensiktsmessig, og bidrar til at dette rammeverket fungerer selv om det er stor forskjell på hvilke verktøy som er gjenstand i forskningene. Dette til tross, ingen modeller og rammeverk er perfekte. Ruthven (2013) sammenlignet en rekke studier som benyttet seg av instrumentell tilnærming som rammeverk. Han konkluderte med at forskerne i stor grad er selektive ved hvilke deler av tilnærmingen og begrepsapparatet de benytter seg av. Denne kritikken er ikke rettet mot rammeverket i seg selv, men hvordan forskere velger å bruke det. For å synliggjøre måten begrepsapparatet benyttes i denne studien vil det i metodekapitlet beskrives hvordan begrepene er tolket og operasjonalisert. I denne studien er det hovedsakelig komponentene i de mentale skjemaene som er benyttet i analysearbeidet. Komponenten som beskriver muligheter for slutninger (*Possibilities of inferences*) vil i denne studien være mindre fremtredende og vektlagt enn de andre komponentene.

En annen utfordring ved å bruke instrumentell genesis som rammeverk er at mange av prosessene skjer på det mentale plan og er dermed utfordrende å observere. Dette kan til dels løses ved å bruke hensiktsmessige former for observasjon. I denne studien benyttes skjerm- og lydopptak slik at prosessen kan observeres og analyseres gjentatte ganger. I utviklingen av oppgavene til denne studien er diskusjon og kommunikasjon lagt inn som et

sentralt element. Dette er gjort for å øke muligheten for å forstå hvordan elevene reflekterer i arbeid med oppgavene og på den måten synliggjøre de mentale prosessene.

Ruthven (2014) argumenterer i en annen artikkel for at instrumentell genesis er et hensiktsmessig rammeverk. «Although some aspects of its conceptual apparatus are rather convoluted, the broad thrust of the instrumental approach has proved valuable in highlighting these processes (…).» (Ruthven, 2014, s. 7). Med utgangspunkt i dette anses det som hensiktsmessig å gjennomføre denne studien i lys av instrumentell genesis som rammeverk og begrepsapparat.

3 Metode

3.1 Forskningsmetode

Forskningsspørsmålene er utgangspunktet for denne studien. Grunnlaget for å besvare disse er observasjoner av elevers diskusjon i arbeid med programmering i matematikkfaget. Forskningsspørsmålene er avgjørende for valg av forskningsmetode. Det er to hovedkategorier for forskningsmetode; kvalitativ- og kvantitativ metode. Kvantitativ forskning tar utgangspunkt i resultater som kan beskrives med tall og statistikk, og en mulighet for generalisering. Kvantitativ forskningsmetode tar i større grad for seg data som uttrykkes med ord og handlinger.

Denne studien tar utgangspunkt i kvalitativ forskningsmetode og et ønske om å se nærmere på elevenes ord og handlinger i arbeid med programmering. Innen kvalitativ forskning vil denne studien kategoriseres som en kasusstudie. Yin (2009) beskriver blant annet to egenskaper som definerer kasusstudie og som vil bli belyst under.

1. Forskning kan ta form som en kasusstudie er når forskeren forsøker å besvare **hvordan** og/eller **hvorfor** noe skjer (Yin, 2009). I denne studien undersøkes elevers arbeid med programmering. Hvordan de løser oppgavene og hva som ligger bak denne fremgangsmåten vil være relevante spørsmål i analysearbeidet.
2. Ved kasusstudie har **forskeren liten påvirkningsmulighet** på hvordan respondentene handler (Yin, 2009). Elevenes arbeid, som er grunnlaget for denne studien, tar den retningen elevene velger. Designet på oppgaven er gitt, men utover dette har observatør liten mulighet til å styre eller påvirke elevenes valg. Dette er heller ikke av interesse siden det er nettopp deres handlinger gjennom arbeidet med programmering som skal belyses.

Fordelen med denne forskningsmetoden er at det gir mulighet for en dypere forståelse av de mentale prosessene som skjer i elevers arbeider med programmering. Det er prosesser som ikke vil kunne belyses på samme måte ved en kvantitativ tilnærming.

Kasusstudie som forskningsmetode er ikke uproblematisk. En utfordring med denne forskningsmetoden er at det er stor frihet til fremgangsmåte og muligheter for forskeren til å styre retningen på konklusjonen.

> Perhaps the greatest concern has been over the lack of rigor of case study research. Too many times, the case study investigator has been sloppy, has not followed systematic procedures, or has allowed equivocal evidence or biased views to influence the direction of the findings and conclusions.» (Yin, 2009, s. 34).

I denne studien er dette forsøkt minimert ved å velge en observasjonsmetode der det er liten mulighet for å påvirke elevene. Et annet argument for at observatør skal har en tilbaketrukket rolle er situasjonene og diskusjonen skal bli så naturlige og autentiske som mulig. Det er lagt opp til at presentasjon av datamateriell, bruk av teoretisk rammeverk og analyse skal være så transparent som mulig.

En annen utfordring Yin (2009) trekker frem ved denne forskningsmetoden er at resultatene i liten grad kan generaliseres til å være gjeldende for et større utvalg(Yin, 2009). I denne studien anses dette som uproblematisk. Målet er ikke å generalisere elevenes arbeid, men å se på hvordan det teoretiske rammeverket kan benyttes til å tolke elevenes arbeid med programmering i naturlige læringssituasjoner. «scientific facts are rarely based on single experiments; they are usually based on a multiple set of experiments that have replicated the same phenomenon under different conditions.» (Yin, 2009, s. 34).

Resultatene fra denne studien vil kunne være en del av en større sammenheng med annen relevant forskning, og dermed bidra til å utvikle og generalisere teorien og hvordan den kan brukes. «case study, like the experiment, does not represent a “sample,” and in doing a case study, your goal will be to expand and generalize theories (analytic generalization) and not to enumerate frequencies (statistical generalization).» (Yin, 2009, s. 34). Programmering i matematikkundervisningen er et relativt nytt forskningsfelt. De første studiene inne et nytt forskningsfelt fortoner seg ofte som kasusstudier der man går i dybden på ulike case for å identifisere aspekter som opptrer innenfor denne tematikken.

3.2 Kontekst og valg av respondenter

Forskningen er gjennomført på egen skole og med elever i egen klasse på 9. trinn. Samtlige elever i klassen fikk tilbudet om å delta i forskningen. Seks elever ble tilfeldig valgt ut blant

de som takket ja. Elevene ble delt i to grupper med tre elever i hver. Ved gjennomføring av oppgave 6 var det på grunn av sykefravær kun to elever per gruppe.
Elevene i denne klassen har lite erfaring med programmering fra tidligere. Dette er bakgrunnen for at blokk-programmeringsspråket Scratch er benyttet i denne studien. Dette er et programmeringsspråk med relativ lav inngangsterskel.

3.3 Utforming av oppgavene i Scratch

I forbindelse med denne studien ble det laget et hefte med seks oppgaver, se vedlegg 2. Klassen jobber med oppgave 1-3 for å danne seg en grunnleggende forståelse for programmering og et fundament å jobbe videre fra. Oppgave 4-6 er ment som gjenstand for observasjon og grunnlag for denne studien. Samtlige oppgaver har en matematisk kontekst og vil på bakgrunn dette kunne plasseres på nivå 2, og i noen tilfeller nivå 3, i Assude (2007) sin nivådeling. Oppgave 4 er designet for at elevene skal utfordres i bruk av flere variabler ved beregning i programmet. De løser denne oppgaven ved å gjøre en remiks av et eksisterende program der de blant annet utfordres på tallforståelse og algebra. Oppgave 5 inneholder et ferdiglaget program som simulerer to terningkast. Summen av øynene på terningene legges sammen og programmet beregner sannsynligheten for at gitte summer inntreffer. Denne oppgaven innebærer i liten grad å skrive eller endre på programmets koder, men legger opp til at elevene skal bruke programmet til å utføre matematiske operasjoner i form av simuleringer. Ved å endre antall simuleringer er idéen bak denne oppgaven at elevene skal se nytten av programmet og oppdage matematiske elementer som store talls lov. I oppgave 6 utfordres elevene i større grad til å bygge programmet selv, men med enkelte hint for å hjelpe dem på vei. Denne oppgaven er rettet mot geometri ved å utfordre elevene til å kode et program som tegner geometriske figurer.

3.4 Datainnsamling

Elevene som observeres sitter på eget rom under arbeid med oppgave 4-6. Resterende elever i klassen jobber med de samme oppgavene til samme tid, men da i klasserommet. Datainnsamlingen er gjennomført ved at det har vært skjerm- og lydopptak av elevenes PC. Opptaksverktøyet som er benyttet er Screencastify. Etter hver økt med observasjon er opptaket gjennomgått, og det er gjennomført et strategisk utvalg for transkripsjon. Elevene

snakker en dialekt som i stor grad skiller seg fra bokmål. For lesbarheten til denne studien er transkripsjonen skrevet på bokmål.
Det ble etter gjennomføring av oppgave 5 oppdaget tekniske utfordringer som har gjort observasjonene ubrukelige. Datagrunnlaget som er brukt i denne studien baserer seg derfor kun på observasjoner fra elevenes arbeid med oppgave 4 og 6. Det er dermed fire transkripsjonsdokumenter. Disse ligger vedlagt som vedlegg 3 til 6.

Transkripsjonsnøkkel

Transkripsjonene inneholder tre kolonner. Første kolonne beskriver oppgavenummer, gruppenummer og sitatnummer. F. eks vil O4G2-15 henvise til at sitatet er fra gruppe 2 (G2) sitt arbeid med oppgave 4 (O4) og sitatet nummer 15 i transkripsjonen.
Andre kolonne beskriver hvilken elev som står bak utsagnet. Elevene betegnes *elev* etterfulgt av et nummer. Dette er for å ivareta anonymitet. Hvis det ikke står oppført noe elevnummer blir det beskrevet noe som skjer på skjermen, men som ikke uttrykkes verbalt av elevene. Dette skjer eksempelvis når de tester programmet. Denne beskrivelsen vil stå i kursiv.
Siste kolonne inneholder selve sitatet eller en observasjon fra skjermopptaket. Non-verbale observasjoner vil være skrevet mellom to klammer hvis de er en del av et elevutsagn.

3.5 Forskningens kvalitet

Forskningens kvalitet bestemmes ut i fra hvordan kunnskapen er produsert (Postholm, Jacobsen & Søbstad, 2018). I dette delkapitlet vil denne studiens pålitelighet og gyldighet belyses.

2.5.1 Reliabilitet - Pålitelighet

Reliabilitet handler om i hvilken grad studien kan etterprøves. Graden av etterprøvbarhet i kvalitativ forskning er lavere enn ved en kvantifisert forskningsmetode. Dette skyldes at det er en rekke elementer ved kvalitativ forskning som vanskelig lar seg gjenskape. Postholm et al. (2018) fremhever at kvalitative studier vil være vanskelige å replikere siden forskere, forskningsfelt og mennesker som deltar i forskningen er forskjellige og tar med sine subjektive og individuelle teorier inn i forskningen. Oppgavene benyttet i denne studien er objektive og kan med enkelthet brukes av andre forskere. Rammeverket som ligger til grunn

for analysen er i denne studien er operasjonalisert med detaljert beskrivelse av hvordan disse brukes i arbeid med observasjonene. Elevene, læringssituasjonen og læringskultur er elementer ved denne studien det er mer utfordrende å gjenskape identisk.
Postholm et al. (2018) trekker frem at forskeren gjennom sitt arbeid må reflektere over sin egen påvirkning på forskningen, og at forskningsprosessen må gjøres synlig slik at andre kan reflektere over den. I denne studien er det skjerm- og lydopptak bruk som observasjonsmetode. Fordelen med denne metoden er at observatør i liten grad har mulighet til å påvirke resultatet. Dette er også bakgrunnen for valg av observasjonsmetode. Det anses som spesielt viktig i denne studien med tanke på at observatør også er læreren til disse elevene og kjenner de godt, både faglig og sosialt.

2.5.2 Validitet - gyldighet

Postholm et al. (2018) beskriver en todeling av gyldighetsbegrepet; indre og ytre gyldighet. Indre gyldighet tar for seg om det er samsvar mellom studiens analyse og de begreper og teorier som benyttes. «Forskeren må påse at det finnes grunnlag for analysene og tolkningene i beskrivelsene i datamaterialet, og at det er sammenheng mellom beskrivelsene og analysene og de tolkninger som blir gjort.» (Postholm et al., 2018, s. 230). For å opprettholde indre gyldighet er det vedlagt transkripsjon fra observasjonene. Studiens analysedel bygger i sin helhet på disse transkripsjonene. Leseren har dermed full innsikt i det datagrunnlaget som analysen tar utgangspunkt i. Transkripsjonene, i lys av operasjonaliseringen av rammeverket, danner analysegrunnlaget i denne studien. Denne transparente fremstillingen åpner for at andre forskere kan gjøre sine egne analyser basert på det samme datamaterialet.

Ytre gyldighet handler om i hvilken grad funn kan overføres til andre kontekster, altså om funnene kan generaliseres. I kvalitativ forskningsmetode knyttes dette opp mot hvor overførbar forskningen er for den som leser. (Postholm et al., 2018, s. 21). Måten rammeverket brukes i analysen av datamaterialet bidrar til at leseren kan trekke paralleller til liknende situasjoner. Dette bidrar ikke til at resultatene kan generaliseres, men måten rammeverket og begrepsapparatet brukes i analysen kan generaliseres til å være gjeldende utover denne studien.

3.6 Etikk

I forkant av forskningen er det søkt NSD om tillatelse til å gjennomføre observasjoner av elevene. De har vurdert at behandlingen av personvernopplysningene tilfredsstiller regelverket for personvern.

Denne studien er gjennomført med elever fra egen klasse. Dette betyr at observatør har en dobbeltrolle som lærer og forsker, og krever noen etiske og metodiske betraktninger. Elevene kan oppleve usikkerhet om det vil få konsekvenser for dem om de velger å ikke delta på studien. Det har blitt informert, både muntlig og i informasjonsdelen til samtykkeerklæringen, at dette ikke vil ha noen form for negative konsekvenser eller av arbeidet med oppgavene i denne studien vil være en del av fagvurderingen. De har også fått både skriftlig og muntlig informasjon om at et eventuelt samtykke kan trekkes tilbake på et hvilket som helst tidspunkt.

For å ivareta de elevene som ikke har gitt samtykke til lydopptak har observasjonsgruppene jobbet på egne grupperom. Dette fører til at ikke det gjøres utilsiktet opptak av andre elever. Som lærer har en mye kunnskap om elevene, både faglig og sosialt. For at dette skal ha minst mulig påvirkning på rollen som observatør jobber elevene alene med oppgavene på grupperommet, med skjerm- og lydopptak som observasjonsmetode. I løpet av arbeidsøktene besøkes elevene et par ganger for å se til at alt er ok. Bortsett fra dette er observatør fysisk fraværende. Dette er også bakgrunnen til at det ikke er benyttet andre observasjonsmetoder i studien, selv om dette kunne vært interessant i analysearbeidet.
I transkriberingen er elevers navn anonymisert slik at det ikke er mulig å spore elevutsagn tilbake til den enkelte. Filer med skjerm- og lydopptak slettes når denne oppgaven er bestått.

3.7 Analyseprosessen

Analysearbeidet har en deduktiv tilnærming til datamaterialet. Elevenes arbeid i programmering belyses med utgangspunkt i instrumentell genesis og kjerneelementene i

LK20. Instrumentell genesis og kjerneelementene vil i dette delkapitlet operasjonaliseres for bruk i denne studien.

3.7.1 Operasjonalisering av instrumentell genesis

I forskningsspørsmål 1 vil elevenes arbeid analyseres gjennom instrumentell genesis.

- **Hvordan kan elevenes arbeid med programmering forstås i lys av instrumentell genesis?**

For å øke graden av gjennomsiktighet er dette rammeverket operasjonalisert slik at både forsker og leser har klart for seg på hvilken måte rammeverket benyttes i analysearbeidet. Tilnærmingen til instrumentell genesis i denne kasusstudien vil være å se på komponentene som utgjør skjemaene i rammeverket. I prosessen med å avdekke disse komponentene er det brukt en deduktiv tilnærming der observasjoner av elevenes arbeid knyttes opp mot det teoretiske rammeverket. Under vil det utdypes hvordan disse komponentene brukes i analysearbeidet.

Aktivitetsmål *(Goals of the activity)*

I analyseprosessen har synet på aktivitetsmål vært at dette i utgangspunktet defineres gjennom oppgavenes utforming, og at elevene har som et grunnleggende mål å løse oppgaven. I analysearbeidet vil det forsøkes å se etter utsagn og handlinger som underbygger at elevene lager nye delmål etter hvert som arbeidet skrider frem. Et eksempel på dette kan være at de støter på utfordringer i deler av programmet som de isolert sett må løse for å komme videre i arbeidet.

I prosessen med å identifisere aktivitetsmål vil det være relevant å se på *hva* elevene gjør i prosessen med å løse oppgavene. Eksempler på dette kan være at de i arbeidet med å lage et program møter utfordringer med kodene i løkka. De kan da lage delmål som handler om å få løkka til å fungere, før de returnerer til hovedmålet som er å få hele programmet til å fungere.

Handlingsregler *(Rules of action)*

Handlingsregler avdekkes ved å undersøke *hvordan* elevene handler for å løse utfordringene de møter, altså se på valg av fremgangsmåte. Eksempel på en handlingsregler kan være måten elevene utnytter muligheten til å teste programmet sitt eller hvordan de integrerer variabler i kodene sine.

Operasjonelle invarianter *(Operational invariants)*
Aktivitetsmål og handlingsregler er stort sett konkrete og observerbare komponenter i den instrumentelle genesisen. Operasjonelle invarianter styres i større grad av mentale prosesser og er dermed ikke like lett å identifisere. En måte å observere disse vil være om elevene på en eller annen måte uttrykker *hvorfor* de handler slik de gjør i arbeid med programmeringen. For eksempel at de sier «*Jeg gjør det slik* (beskriver handlingsregel) *fordi* … (operasjonell invariant)». Det er ikke en selvfølge at elevene finner det naturlig å beskrive sin egen arbeidsprosess så detaljert. Hvis det ikke kommer utsagn som beskriver de operasjonelle invarianter vil det også være en mulighet å komme med antagelser og hypoteser basert på handlingsreglene elevene bruker og de utsagnene som foreligger. Teoremforståelse (*theorems-in-action*) vil i denne studien knyttes opp mot elevenes matematiske forståelse og på hvilken måte denne påvirker handlingsreglene. Konseptforståelse (*consepts-in-action*) knyttes opp til elevenes forståelse av de tekniske aspektene ved programmeringen.

Muligheter for slutninger *(Possibilities of inferences)*
Denne delen av instrumentell genesis kommer til syne om elevene utvikler en forståelse i arbeidet som lar seg overføre eller generaliseres til nye situasjoner. Med tanke på at dette er en relativt liten studie, med begrenset datamateriell over en kort tidsperiode, kan det bli utfordrende å avdekke denne komponenten av skjemautvikling.

3.7.2 Operasjonalisering av kjerneelementene

I forskningsspørsmål 2 vil elevenes arbeid knyttes opp mot kjerneelementene i læreplanen.

- **På hvilken måte kan kjerneelementene komme til syne gjennom elevers arbeid med programmering?**

Operasjonaliseringen beskrevet i tabellen er hentet fra Utdanningsdirektoratet (2020) sin beskrivelse av kjerneelementene. I analyseprosessen har sekvenser fra elevenes arbeid blitt undersøkt og sett i sammenheng med beskrivelsene under. I de tilfelle kjerneelementene har vært representert i resultatene har disse blitt knyttet sammen. I forbindelse med denne studien er beskrivelsen av kjerneelementene i LK20 brutt ned til kulepunkter. Denne konkretiseringen er gjort for at kjerneelementene skal være mer anvendelige, og lettere gjenkjennbare, i analysearbeidet.

Kjerneelementer	**Elevene:**
Utforskning og problemløsning	- leter etter mønstre og sammenhenger - diskuterer seg fram til en felles forståelse - Jobber med problemer de ikke kjenner fra før - Bryter problem ned i delproblem - vurderer gyldigheten på løsningen sin
Modellering og anvendelse	- lager modeller som beskriver noe fra virkeligheten på en matematisk måte - vurdere om modellene er gyldige - bruker sin matematikkforståelse til å løse oppgaver
Resonnering og argumentasjon	- utformer egne resonnementer - forstår medelevers resonnement - begrunner egne fremgangsmåter
Representasjoner og kommunikasjon	- uttrykker matematiske begreper, sammenhenger og problem - bruker matematisk språk når de diskuterer - bruker ulike representasjoner
Abstraksjon og generalisering	- bruker formelt språk og resonnement i arbeid med oppgaven - oppdager sammenhenger og strukturer - formaliserer sammenhenger ved å bruke algebra og hensiktsmessige representasjoner

Matematiske kunnskapsområder	- jobber med elementer fra de matematiske kunnskapsområdene
	  ○ tall og tallforståelse
	  ○ algebra
	  ○ funksjoner
	  ○ geometri
	  ○ statistikk
	  ○ sannsynlighet

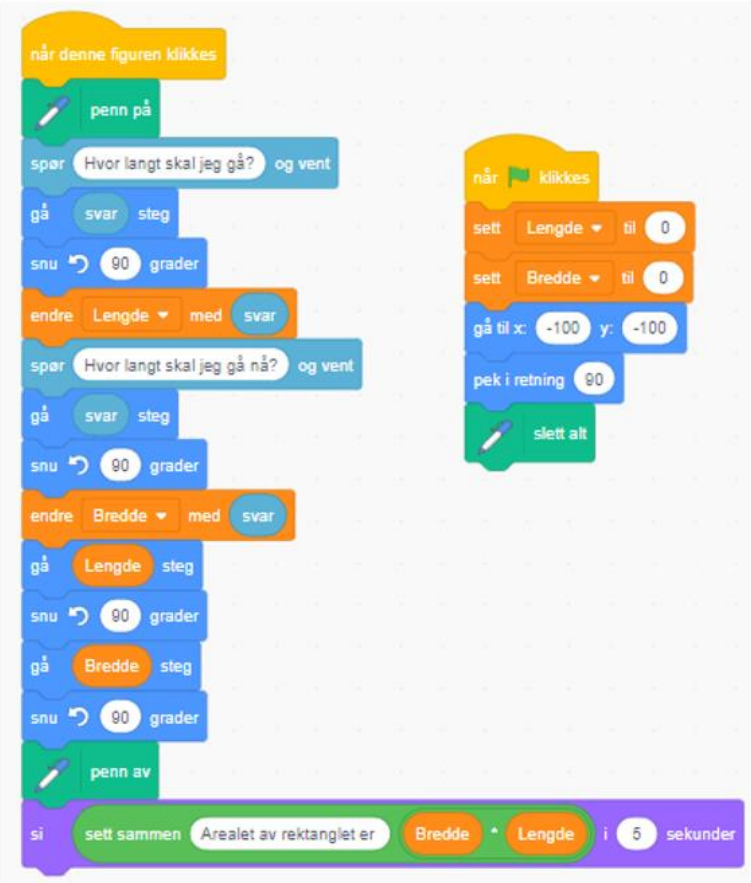
Tabell 1: Operasjonalisering av kjerneelementene.

4 Resultater og analyse

I dette delkapitlet vil resultat og analyse fra observasjonene presenteres. Observasjonene tar for seg elevenes arbeid med oppgave 4 og 6. Oppgavene blir presentert etterfulgt av en gjennomgang av observasjoner. Resultatene til videre bli analysert i lys av de to forskningsspørsmålene. I analysen rettet mot instrumentell genesis vil gruppene presenteres individuelt. Analyse av hvordan kjerneelementene kommer til syne i arbeidet gjøres samlet for begge gruppene.

4.1 Oppgave 4

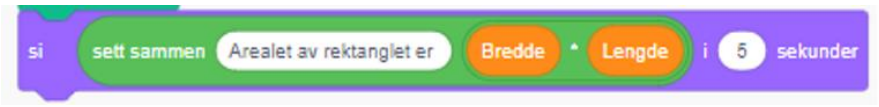
I denne oppgaven ble elevene presentert for et ferdiglaget program som beregner areal av et rektangel. Se vedlegg 2 for full oppgavebeskrivelse.



Figur 4: Skjermdump av kodene gitt i oppgave 4.

Bakgrunnen for å ha med denne oppgaven i studien var et ønske om at elevene skulle utfordres på beregninger der de må forholde seg til flere variabler og regnearter. De ble presentert for et program som beregnet areal av et rektangel ut fra de verdiene som blir

oppgitt som lengde og bredde. Elevene ble blant annet utfordret på å endre operatørblokkene slik at programmet beregner omkretsen i stedet for areal. Disse beregningene gjøres i operatørblokka nederst i programmet. Figuren 5 viser utgangspunktet elevene startet med.

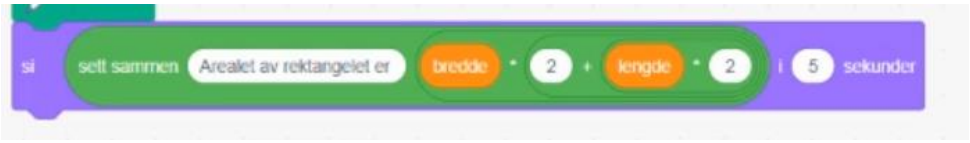


Figur 5: Skjermdump av kode i operatørblokka i oppgave 4.

Begge gruppene møtte utfordringer i arbeidet med denne oppgaven, og endte opp med å løse de på forskjellige måter. Observasjoner fra elevenes arbeid beskrives under. Videre vil disse sees i lys av forskningsspørsmålene.

4.1.1 Gruppe 1 – Observasjoner

Utgangspunktet for elevene var et program som beregner areal av et rektangel. Hoveddelen av oppgaven bestod i å gjøre en endring, eller remix, slik at programmet beregnet omkrets av rektanglet i stedet for arealet. Observasjonene som er tatt med i denne presentasjonen konsentreres om elevenes arbeid med operatørblokkene som utfører beregningene. Elevene gjør en remix av programmet der de endrer sammensettingen av blokkene. De setter blokkene opp som vist på bildet under og gir uttrykk for at de tror dette blir riktig [O4G1 1-2].



Figur 6: Skjermdump av gruppe 1 sine koder i arbeid med oppgave 4.

O4G1-1	Elev 1	Da kan vi ta lengde gange 2 pluss bredde gange 2.
O4G1-2	Elev 2	Det blir jo riktig nå, blir det ikke det?

Elevene tester ikke programmet. Diskusjonen fortsetter og elev 1 argumentere for at dette er riktig så lenge programmet forholder seg til prioriteringsrekkefølge [O4G1 11]. Samme elev kommer også med en alternativ løsning ved at de legger inn at sidene i rektanglet adderes [O4G1 13].

O4G1-9	Elev 1	Bredde gange 2…
O4G1-10	Elev 2	Pluss lengde gange 2. Det blir jo rett.

O4G1-11	Elev 1	Ja, han må jo bare ta ganginga først, før han tar plussen, hvis han er litt smart da.
O4G1-12	Elev 2	Jeg tror han skal ta plussen først. [*uten at han argumenterer eller utdyper ytterligere*]
O4G1-13	Elev 1	Må vi ikke ta dette også? [*henviser igjen til den øvrig del av programmet med musepekeren.*] Eller skal vi bare ta fire plusser?
O4G1-14	Elev 2	Vi har jo gjort det riktig nå.
O4G1-15	Elev 1	Ja. Test. Test?
O4G1-16	Elev 2	Ja.

Elevene tester programmet med 5 og 8 som verdier på sidene i rektanglet. De konkluderer med at svaret de får må være feil [O4G1 18-22].

O4G1-18	Elev 1	Ok, nå har jeg det rett her. 5 og 8. [*dette er verdiene som oppgis om inndata i programmet.*] Der! Omkretsen av rektanglet er 96.
O4G1-19	Elev 2	Der fant vi det da.
O4G1-20	Elev 1	Men blir det rett da?
O4G1-21	Elev 3	Ja
O4G1-22	Elev 1	Blir det rett? 96 liksom. Blir ikke det litt mye?

De velger å prøve programmet en gang til. Denne gangen bruker de andre verdier på sidene. De argumenterer for at det er lettere å vurdere svaret hvis verdiene de oppgir er lave tall og at de dermed vet hva svaret skal bli før programmet gjør sine beregninger [O4G1 24-27].

O4G1-24	Elev 1	Eller vi prøver en gang til der vi tar noen små tall. Hvis det er 1 så, hvis alle sidene er 1 og 2 så blir det jo, eh 6.
O4G1-25	Elev 2	Hvordan da? Det skal jo bli 4.
O4G1-26	Elev 1	Nei, fordi bredden er 2 og da blir det 4. Og så lenge er 1 og da blir det 2.
O4G1-27	Elev 1	2 + 4 er 6.

Ny test av programmet gir fremdeles ikke elevene det svaret de forventer. De kommer ikke frem til noen løsninger for hvordan de skal få programmet til å gjøre beregningene slik de ønsker. Elev 1 foreslår derfor å prøve den alternative fremgangsmåten hun foreslo tidligere i arbeidsøkta, at de legger inn addisjon mellom alle sidene i rektanglet [O4G1 39].

O4G1-39	Elev 1	Så tar vi bort disse igjen. Vi tar bort dette så tar vi heller pluss. Vi må beholde lengde og bredde tror jeg.

Se figur 7 for måte elevene satte sammen blokkene.

Figur 7: Skjermdump av gruppe 1 sitt endrede forslag til koder i oppgave 4.

Elevene fortsetter å støte på utfordringer med den alternative måten å beregne omkretsen. De bruker 1 og 2 som inndata, men programmet gir de 12 til svar [O4G1 44-45].

O4G1-44	Elev 1	Okei nå begynne vi. Vi tar noe lett. Vi tar 1 og 2 igjen.
O4G1-45	Elev 1	Nå ble det enda høyere enn i stad.

Like etter kommer læreren inn i rommet. Elevene er frustrerte over at programmet ikke gjør slik de forventer. Læreren ser gjennom kodene og konkluderer med at disse skal fungere. Programmet testes en gang til og da blir svaret slik de forventer. Elevene stusser på dette [O4G1 63]. Det kommer frem at elevene ikke har brukt den delen av programmet som nullstiller verdien til variablene [O4G1 64-65]. Når dette gjøres fungerer programmet slik det skal.

O4G1-63	Elev 1	Det var ikke det som stod i stad. Da tok jeg 1 og 2. Så ble det 2 og 4 plutselig.
O4G1-64	Lærer	Det er ikke det at du ikke nullstilte programmet da?
O4G1-65	Elev 1	Det kan være.
O4G1-66	Lærer	1 og 2. [*bruker 1 og 2 som inndata*]
O4G1-67	Lærer	Omkretsen er 6.
O4G1-68	Lærer	Hvis du gjør dette en gang til nå. 1 og 2.
O4G1-69		*Kjører programmet en gang med 1 og 2 som inndata, uten å nullstille det. Programmet oppgir da at omkretsen er 12.*
O4G1-70	Elev 1	Ja, det var nok det som skjedde i stad.

Elevene ser seg fornøyd med løsningen og undervisningsøkta avsluttes.

4.1.2 Gruppe 1 – Instrumentell genesis

I dette delkapitlet vil observasjoner fra gruppe 1 sitt arbeid med oppgave 4 belyses og analyseres med utgangspunkt i instrumentell genesis.

Når elevene jobber med denne delen av oppgaven har de et overordnet mål for aktiviteten. Dette målet er å endre de matematiske beregningene i programmet slik at det beregner omkretsen av rektanglet. Dette målet blir gitt til elevene gjennom oppgavens innhold. De forsøker å oppnå dette ved å bruke de handlingsregler de har tilegnet seg gjennom tidligere erfaring med operatørblokker og variabler.

Elevene sin handlingsregel i denne situasjonen er å arrangere blokkene slik at variablene og regneartene kommer i samme rekkefølge som de ville gjort hvis de hadde jobbet med denne oppgaven i matteboka. De begrunner dette valget med en antakelse om at programmet

forholder seg til prioriteringsrekkefølgen [O4G1 10-12]. Elevene tar høyde for at programmet prioriterer multiplisering fremfor addering og viser at de har en forståelse for rekkefølgen beregningene skal prioriteres. Basert på utsagnene [O4G1 10-12] kan det tyde på at elevene ikke har en forståelse av at blokker i programmeringsspråket Scratch fungerer på samme måte som parenteser i matematikken. De leser trolig operatørblokkene uten å ta hensyn til grupperinger og tolker beregningene som $bredde \cdot 2 + lengde \cdot 2$. Oppbyggingen av operatørblokkene fører derimot til at programmet leser kodene som $bredde \cdot (2 + (lengde \cdot 2))$. Elevene sin handlingsregel i denne situasjonen tar ikke høyde for at hver enkelt blokk i programmeringsspråket Scratch fungerer på samme måte som parenteser i matematikken. Når elevene tester programmet med 5 og 8 som sider i rektanglet får de et svar de mener ikke kan være riktig [O4G1 18-22].

De operasjonelle invarianter som ligger til grunn for elevenes handlingsregel i denne situasjonen viser at elevene baserer handlingene sine på grunnlag av en matematisk overbevisning og forståelse. De viser med det at de har en god teoremforståelse. Konseptet ved at operatørblokkene fungerer som parenteser er ikke blitt en del av elevenes operasjonelle invarianter i arbeid med beregninger. Utfordringene elevene møter skyldes at deres konseptforståelse av operatørblokkenes egenskaper fører til at de har en handlingsregel for denne typen beregninger som ikke fører frem.

En annen handlingsregel som ønskes belyst i denne analysen synliggjøres når elevene skal utføre videre testing av programmet. De velger da å benytte seg av lavere tall for lengde og bredde når de tester programmet [O4G1 24]. Elevene gir uttrykk for at det er enklere å tolke hva programmet gjør hvis de vet hva svaret skal bli. De teste programmet en gang til, men velger da inndata som skal gjøre det lettere for dem å tolke utfallet av beregningene. Elevene erfarer at valg av hensiktsmessig inndata kan forenkle arbeidet med testingen og feilsøking av program. Siden denne situasjonen er en isolert hendelse er det utfordrende å trekke slutninger om at det har skjedd en varig endring av elevenes handlingsregel, men argumentene elevene bruker kan tyde på at dette likevel er tilfelle.

Endringen av inndata forenkler testing og feilsøkingen, men det endrer ikke utfordringene elevene har ved at programmet utfører beregningene annerledes enn de forventer. I videre arbeid med oppgaven forsøker elevene å ta bort alle operatørblokkene, og sette de sammen på en ny måte der de bare benytter seg av operatørblokker med addisjon [O4G1 39].
Se figuren 7 for måten elevene endret blokkene.

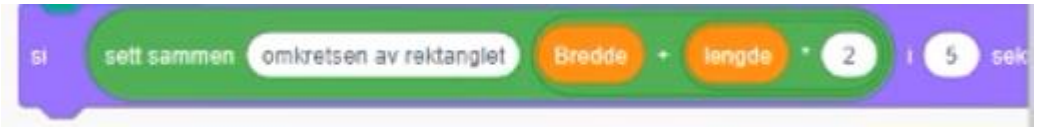
I denne situasjonen unngår de problemet med prioriteringsrekkefølge og parenteser (blokker) ved å redusere den sammensatte beregningen av multiplikasjon og addisjon til å kun bestå av addisjon. Dette er mulig siden multiplikasjon er gjentatt addisjon. Denne endringen hjelper de å løse delmålet knyttet til beregningen. Sett i lys av skjemautvikling vil denne handlingsregelen løse denne oppgaven og tilsvarende situasjoner. Ved mer avanserte beregninger vil ikke denne handlingsregelen kunne føre frem. Graden av mulige slutninger elevene kan trekke ut av arbeidet med operatørblokkene i denne oppgaven vil dermed være relativt liten.

Når kodene er endre til kun å bestå av addisjon vil programmet fungere. Elevene får likevel ikke det svaret de forventer [O4G1 44-45]. De bruker også denne gangen 1 og 2 som inndata. Dette underbygger påstanden over, om at elevene har hatt en endring i handlingsregler for bruk av inndata når de skal teste programmet. Svaret blir likevel ikke riktig. Programmet oppgir 12 som svar. Elevene forventer 6, som også er det riktige svaret. Dette bunner ikke i at kodene i programmet er feil, men at elevene ikke har vært bevisste på at verdien til variablene må nullstilles for at ikke de skal akkumuleres. Dette er en utfordring som gjelder spesifikt for programmering, og er ikke gjeldende i andre deler av matematikken. Elevene viser her manglende konseptforståelse over for denne funksjonen ved programmeringsspråket Scratch.

4.1.3 Gruppe 2 – Observasjoner

Observasjonene som er med i denne presentasjonen er knyttet til elevene i gruppe 2 sitt arbeid med operatørblokkene. I likhet med gruppe 1 møter denne gruppa utfordringer når de skal endre programmet slik at det beregner omkretsen, i stedet for arealet, av et rektangel.

Gruppe 2 starter arbeidet med å legge til en addisjonsblokk. De setter blokkene opp som vist på figuren under. De gir uttrykk for at det forventes at variablene først adderes, for så å multipliseres med 2 [O4G2-7].



Figur 8: Skjermdump av gruppe 2 sitt forslag til løsning i oppgave 4.

O4G2-7 Elev 4 Sånn? Nå er det det *[bredde]* pluss lengde gange to.

Elevene tester programmet to ganger, først med 50 og 60 som inndata, så med 100 og 200 [O4G2 8-12]. Programmet gir uriktige svar i begge tilfellene, uten at elevene gir uttrykk for at de oppfatter dette.

O4G2-8		*Tester programmet med 50 og 60 som lengde og bredde i rektanglet. Programmet gir svaret 164.*
O4G2-9	Elev 5	Prøv på nytt. [*Eleven gir ikke uttrykk for at det tror svaret er feil*] Skriv 100.
O4G2-10	Elev 4	100 [*eleven gir lengde verdien 100*]
O4G2-11	Elev 4	200 [*eleven gir bredde verdien 200*]
O4G2-12		Programmet beregner omkretsen med de verdiene til å bli 400

Først når de tester med verdiene 100 for lengde og 100 for bredde skjønner de at svaret til programmet ikke stemmer med omkretsen til rektanglet [O4G2 13-16].

O4G2-13	Elev 5	Prøv 100 to ganger
O4G2-14		Programmet beregner omkretsen med de verdiene til å bli 300
O4G2-15	Elev 4	Det blir jo ikke 300. Det var ikke 300 da.
O4G2-16	Elev 5	100 + 100 + 100 + 100. Det må jo bli 400.

Gjennom videre testing konkluderer de med at programmet kun beregner lengden til tre av sidene [O4G2 21].

O4G2-21 Elev 4 Han tar jo bare tre av sidene.

Lærer kommer inn i rommet og elevene søker forklaring på hvorfor programmet bare beregner tre sider [O4G2 25-39]. Elevene ledes inn på at det har noe med måten operatørblokkene og regneartene er satt opp. Etter noe diskusjon konkluderer elevene med at programmet kun tar med én bredde i beregningen slik kodene er [O4G2 37]. Elev 5 følger opp med å relatere dette til regnerekkefølge [O4G2 38].

O4G2-25 Elev 4 Hallo, du. Han regner bare tre av sidene. Se nå. 50. 50.

O4G2-26		*Elev skriver inn 50 som lengde og 50 som bredde.*
O4G2-27	Elev 4	Det blir 150.
O4G2-28	Elev 5	Hvorfor blir det et kvadrat da?
O4G2-29		*Elev konstaterer at det tegnes et komplett kvadrat i programmet til tross for at beregningen av omkretsen kun tilsvarer tre sider.*
O4G2-30	Lærer	Ja, hvorfor gjør han det?
O4G2-31		*Lærer lar elevene få litt tid til å tenke.*
O4G2-32	Lærer	Dere sier han skal ta bredde pluss lengde ganger 2.
O4G2-33	Elev 5	Den er 2.
O4G2-34	Lærer	Hvorfor det?
O4G2-35	Lærer	Hvorfor blir det ikke rett nå?
O4G2-36	Elev 4	Men det er bare 3 sider.
O4G2-37	Elev 4	For at det er bredde én gang og lengde 2 ganger.
O4G2-38	Elev 5	Ah, vi ganger jo alltid før pluss, og da blir det lengde ganger 2 pluss bredde.
O4G2-39	Elev 5	Ja da blir det bare tre!

Skoletimen avsluttes like etter, så det ble ikke tid til å endre kodene i programmet slik at det beregnet omkretsen for hele rektanglet.

4.1.4 Gruppe 2 – Instrumentell genesis

Elevenes mål for aktiviteten er å endre programmet slik at det beregner omkretsen av et rektangel. Handlingsregelen de benytter seg av setter opp blokkene som vist på figur 8. Elevene begrunner oppsettet ved at de mener programmer først adderer variablene for så å multiplisere med 2 [O4G2 7]. Dette indikerer at de har en forventning om at programmer utfører beregningene i den rekkefølgen de står, fra venstre mot høyre.

O4G2-7 Elev 4 Sånn? Nå er det det *[bredde]* pluss lengde gange to.

De prøver ut programmet med 100 som verdi for både lengde og bredde og forventer å få 400 til svar, men programmet beregner omkretsen til å være 300 [O4G2 13-16]. Elevene forsøker igjen med annen inndata, men opplever igjen at svaret ikke blir slik de forventer [O4G2 17-20]. De innser at handlingsreglene de har for beregningen ikke fungerer.
Elevene gir uttrykk for at de har forventning til hvordan omkretsen av et rektangel skal beregnes [O4G2 7]. De bekrefter at de har en forståelse at det er en definert rekkefølge for regneoperasjoner når det er ulike regnearter [O4G2 38]. Elevene viser at de har god teoremforståelse for hvordan rekkefølge på regnearter skal prioriteres i matematikken.

Bakgrunnen for at programmet utfører beregningen feil ligger i elevenes konseptforståelse. Elevene har ikke en god nok forståelse av hvordan regnerekkefølgen håndteres i programmeringsspråket Scratch. De operasjonelle invarianter som ligger til grunn for elevenes handlingsregel i denne situasjonen er dermed mangelfull på konseptnivå.
Elevene har gjennom arbeidet med denne oppgaven utvidet sin forståelse av hvordan programmet fungerer. Dette fører til en utvikling av operasjonelle invarianter og dermed en varig endring av handlingsregler for bruk av programmering generelt, og programmeringsspråket Scratch spesielt. Elevenes reaksjon i slutten av undervisningsøkta [O4G2 37-39] vitner om at det falt noen mentale brikker på plass for dem. De forstod at programmet følger faste regler for beregninger og ga uttrykk for at de skjønte dette med bakgrunn i sin forståelse av prioriteringsrekkefølge.

4.1.5 Oppgave 1 - Kjerneelementer

Over har vi sett på situasjoner fra arbeidsøkta med utgangspunkt i instrumentell genesis. De samme observasjonene vil nå sees i lys av kjerneelementene fra læreplanen.

Utforskning og problemløsning

Utforskning er en del av dette kjerneelementet. Det beskriver at elevene skal *lete etter mønster og sammenhenger*. Observasjonene viser at dette er noe elevene stort sett jobber med gjennom hele arbeidsøkta. Funn fra begge gruppenes arbeid synliggjør hvordan de forsøker å oppnå en felles forståelse gjennom arbeid med oppgavene. Gruppene jobber sammen for å løse utfordringene. Dette arbeidet drives frem gjennom diskusjon og meningsutveksling. Eksempler på dette kommer blant annet til syne når elevene i gruppe 1 forsøker å avdekke sammenhengen mellom oppbyggingen av operatørblokkene og svaret de fikk. I denne situasjonen *diskuterer elevene seg frem til en felles forståelse* for hvordan man beregner omkrets av et rektangel og hvordan dette kan overføres til programmet [O4G1 24-31].
Eksempler der elevene diskuterer seg frem til felles forståelse finner vi også i gruppe 2. I slutten av timene viser de forståelse for at programmet bare tar med en av rektanglets bredder i beregningen.

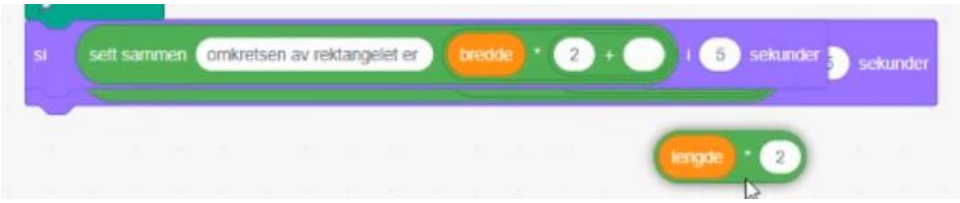
Dette kjerneelementet beskriver også at elevene s*kal vurdere gyldigheten av løsningen sin*. I gruppe 1 kommer dette til syne når de tester programmet og oppdager at programmet gir

96 til svar når det beregner omkretsen med sidene 5 og 8. De konkluderer da med at denne løsningen ikke kan være gyldig.

Lignende situasjon er også representert i funn fra gruppe 2 sitt arbeid. Elevene velger 100 som verdi på både lengde og bredde. De forventer 400 til svar, men programmet beregner det til å bli 300. De konkluderer dermed at dette svaret ikke stemmer. I det videre arbeidet med programmeringen endret gruppe 1 handlingsregler for valg av inndata. Dette begrunner de med at det vil forenkle vurderingen av gyldigheten til beregningene.

I prosessen med å avdekke hvorfor programmet ikke beregner slik elevene ønsker, forsøker de å bryte ned og isolere problemet [O4G2 31-35]. Elev 1 trekker de forskjellige operatørblokkene ut, i et forsøk på å isolere problemet. Først trekker eleven ut den innerste operatørblokka [O4G2 32].

O4G1-32 Elev 1 Ja, det er det jeg også tenker fordi denne her er jo 1 [*henvis til innerste operatørblokk*].



Figur 9: Skjermdump av første steg i endringen til elev gruppe 2 i arbeid med oppgave 4.

Videre trekker eleven ut neste blokk [O4G2 33].

O4G1-33 Elev 1 Og dette er én. [*Henviser til nest innerste operatørblokk*]

Figur 10: Skjermdump av andre steg i endringen til elev gruppe 2 i arbeid med oppgave 4.

Til slutt trekker hun ut siste operatørblokka [O4G1-35].

O4G1-35 Elev 1 Og dette er én. [*Henviser til ytterste av de tre operatørblokkene som utfører selve beregningene*]

Figur 11: Skjermdump av andre steg i endringen til elev gruppe 2 i arbeid med oppgave 4.

Denne systematiske måten å tilnærme seg problemet er i tråd med algoritmisk tenkning og problemløsning. «Algoritmisk tenkning er viktig i prosessen med å utvikle strategier og framgangsmåter for å løse problemer og innebærer å bryte ned et problem i delproblemer som kan løses systematisk.» (Utdanningsdirektoratet, 2020, s. 2)

Modellering og anvendelse

Anvendelse er en del av dette kjerneelementet og legger føringer for at elevene skal *bruke sin matematikkforståelse til å løse oppgaver*. Elevene i gruppe 1 viser matematikkforståelse når de begrunner for hvordan de organiserer operatørblokkene og argumenterer for at dette blir riktig hvis programmet følger prioriteringsrekkefølgen fra matematikken.

Resonnering og argumentasjon

Elevene skal *utforme egne resonnementer* og bruke disse til å forstå og løse faglige utfordringer. I elevenes arbeid kommer resonnementer til uttrykk gjennom diskusjonen. Elevene i gruppe 1 beskriver hvordan de mener programmet vil utføre de matematiske beregningene ved at de forventer at multiplikasjon prioriteres før addisjon. [O4G1 11-12] Elev 1 begrunner ikke sitt resonnement, men uttrykker forståelse for de matematiske reglene for prioritering av regnearter. Elev 2 argumentere for at programmet utfører addisjonen først. Dette argumentet blir heller ikke begrunnet, men det er vanskelig å tolke dette annerledes enn at det bunner i en misoppfatning. Elevene sine påstander, og senere erfaringer, vitner om at de utvikler forståelse gjennom diskusjon og arbeid med programmeringen. Dette er i tråd med kjerneelementene som beskriver at «elevene skal utforme egne resonnementer både for å forstå og for å løse problemer.» (Utdanningsdirektoratet, 2020, s. 3)

I gruppe 2 kommer også elevenes resonnement til uttrykk i prosessen med å tolke hvordan programmet utfører beregningene. De har innsett at svarene blir feil. Gjennom resonnering kommer de frem til at feilen ligger i måten programmet utfører beregningene, og at det bare tre av sidene er med i svaret. [O4G2 35-59].
Elev 5 innser at reglene for regnerekkefølge i matematikk må anvendes i programmeringen og at dette er grunnen til at programmet ikke gir riktig svar basert på verdiene til variablene.

Representasjoner og kommunikasjon

Oppgavene elevene jobber med i denne situasjonen utfordrere de på å diskutere og kommunisere matematikk med hverandre. Oppgaveteksten legger føringer for at elevene skal forklare for hverandre hva de forskjellige blokkene gjør, og hva programmet som helhet gjør. Elevene blir da utfordret til å uttrykke matematiske og programmeringstekniske sammenhenger. Dette er i tråd med kjerneelementet representasjon og kommunikasjon der kommunikasjon beskrives ved at *elever bruker matematisk språk i samtaler, argumentasjon og resonnementer.* Observasjonene beskriver kommunikasjon mellom elevene der de resonnerer og argumenterer i arbeidet med oppgavene.

Utfordringene elevene møter i disse situasjonene, kan i stor grad knyttes opp mot representasjoner. Programmeringsspråket har andre måter å representere matematikken på enn elevene er vant til. Dette resulterer i at de har utfordringer med å uttrykke de aritmetiske regneoperasjonene på en måte som løser oppgavene slik de ønsker. Det er blant annet måten programmeringsspråket Skratch representerer parenteser på som er utfordrende for elevene i situasjonene over. I programmeringsspråket fungere som nevnt blokkene som parenteser. Når elevene ikke tar høyde for dette vil programmet prioritere regnerekkefølgen annerledes enn det elevene forventer. De møter da utfordringer med representasjonene når de skal endre operatørblokken. Figur 6 og 8 viser at elevene på dette tidspunktet ikke forholder seg til måten programmet representerer parenteser på.

Dette danner grunnlag til å antyde at arbeid med programmering utfordrer elevene til å forstå, og ta i bruk, nye representasjonsformer. Dette er i tråd med læreplanens beskrivelse. «Elevene må få mulighet til å forklare og begrunne valg av representasjonsform. Elevene må

kunne oversette mellom matematiske representasjoner og dagligspråket og veksle mellom ulike representasjoner.» (Utdanningsdirektoratet, 2020, s. 3)

Abstraksjon og generalisering

Elevene skal *formalisere sammenhenger ved å bruke algebra og hensiktsmessige representasjoner*. Funn fra elevenes arbeid med oppgave 4 viser hvordan de jobber med abstraksjon gjennom bruk av variabler når de skal endre programmet til å beregne omkrets i stedet for areal. For at beregningene skal være gjeldende for alle verdier må elevene bruke variabler for verdiene til lengde og bredde. Denne generaliseringen er i tråd med kjerneelementet ved at de *bruke algebra og hensiktsmessige representasjoner* i dette arbeidet.

Programmering kan variablene defineres med navn. I arbeid med oppgavene er variablene definert som *lengde* og *bredde*. I diskusjonene bruker elevene disse definisjonene når å uttrykke sine resonnementer og argumenter på en hensiktsmessig måte [O4G1 1-2,9-10].

Matematiske kunnskapsområder

Kjerneelementene viderefører de matematiske kunnskapsområdene fra forrige læreplan. Observasjonene viser at elevene møter utfordringer ved å representere regneoperasjoner i programmeringsspråket Scratch. Dette arbeidet utfordrer elevene i arbeid med tallforståelse og algebra.

4.2 Oppgave 6

I denne arbeidsøkta var oppgavene i større grad rettet mot geometri. Elevene jobbet med en tredelt oppgave. I deloppgave 1 ble de utfordret på å lage et program som tegner geometriske figurer som trekanter, firkanter, sekskanter osv. I deloppgave 2 ble de utfordret på å lage et program som tegner en mangekant som har sider tilsvarende en verdi som oppgis som inndata. Deloppgave 3 utfordret elevene på å lage et program som tegner en figur som likner mest mulig på en figur fra oppgaven. Elevene ble da utfordret på å lage et program som tegner en Pac-man som vist på figuren under. Se vedlegg 2 for full oppgavebeskrivelse.

Figur 12: Skjermdump fra Pac-man i oppgave 6-3.

4.2.1 Gruppe 1 – Observasjon

Elevene i gruppe 1 starter med å lage et program som tegner en trekant. De tar utgangspunkt i eksemplet på oppgavearket. Se figur 13 for deres koder til dette programmet.

Figur 13: Gruppe 1 sine koder for et program som tegner en trekant.

I neste deloppgave skal de endre programmet slik at det tegner en firkant. Elevene endre operatørblokka til å bli 360/4 [O6G1 18-19]. Testing av programmer viser at de også må endre løkketelleren til 4 [O6G1 22].

O6G1-18	Elev 2	Nå skal vi lage en firkant. Det er jo mye lettere, er det ikke det?
O6G1-19	Elev 2	Da må vi fikse på den snu-blokka.
O6G1-20		*Endrer divisor i operatørblokk fra 3 til 4.*
O6G1-21		*Tester programmet med resultat at det bare tegner tre av sidene.*
O6G1-22	Elev 1	Nei, vi må jo endre løkka også til 4.

De bruker samme fremgangsmåte for å endre programmet til å tegne en sekskant [O6G1-25].

O6G1-24	Elev 2	Tegn en sekskant.
O6G1-25	Elev 1	Da kan vi jo, eh, den sånn og den sånn. [*endrer divisor og løkke til 6*]
O6G1-26		*Tester programmet og det tegnet en sekskant.*

I deloppgave 2 utfordres elevene på å lage et program som spør hvilken mangekanter som tegnes, og deretter tegner en figur som har like mange sider som oppgis som inndata. Elevene bygger videre på programmet de lagde i deloppgave 1 [O6G1 33-37].

O6G1-33	Elev 2	Hvordan gjør vi dette?
O6G1-34	Elev 1	Vent. Spør hvor mange. [*setter inn spørre-blokk og endrer spørsmålet slik at det er tilpasses oppgaven*]
O6G1-35	Elev 2	Sånn.
O6G1-36	Elev 1	Og vent. Jeg er ikke helt sikker på hva "og vent" betyr.
O6G1-37	Elev 1	Ah, jeg tror vi har det nå. Vi setter inn svar [*sett inn svar-blokk i løkka og som divisor*]

De tester programmet og konkluderer med at det fungerer slik de ønsker [O6G1 38-43].

O6G1-38	Elev 2	Vi prøver med 10.
O6G1-39	Elev 1	1,2,3,4,5,6,7,8,9,10. [*Teller at det blir en tikant*]
O6G1-40	Elev 1	Nå må vi kanskje prøve med flere da.
O6G1-41	Elev 1	15 [*setter svar til 15*]
O6G1-42	Elev 2	Oi, det ser nesten ut som en runding.
O6G1-43	Elev 1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15. Det blir rett det.

I deloppgave 3 skal de lage et program som tegner en Pac-man. Elevene koder programmet sitt med utgangspunkt i eksempel 3 på oppgavearket. De ønsker å få strekene i figuren tettere ved å øke løkketelleren [O6G1 49-50].

O6G1-49	Elev 2	Men hvordan kan vi få flere streker?
O6G1-50	Elev 1	Hva om vi gjentar 50 ganger?

Dette fører til at programmet tegner flere streker, men de legges oppå hverandre slik at resultatet blir det samme. Videre forsøker de å endre vinkelen i rotasjonsblokka. Først prøver de med 10 grader, deretter 0,1 [O6G1 54,57].

O6G1-51		*Tester programmet som kun legger strekene over hverandre.*
O6G1-52	Elev 1	Og hvis vi endrer den [*rotasjons-blokka*] til 10 grader.
O6G1-53		*Tester programmet*
O6G1-54	Elev 1	Åh! Jeg tror dette kan gå faktisk.
O6G1-55	Elev 1	Hvis vi tar 0.1 [*i rotasjonsblokka*]
O6G1-56		*Tester programmet*

Elev 1 konkluderer med at 0,1 er unødvendig tett og at dette gjør at programmet bruker lang tid på å tegne figuren. De setter rotasjonen til å være 0,9 grader for hver gang løkken kjøres.

Elevene øker løkketelleren til 100. Når de kjører programmet oppdager de at det med 0,9 i rotasjon ender opp med å lage en sektor på 90 grader [O6G1 71]. Med det utgangspunktet beregner de hvor stor sektoren blir med ulike løkketellere [O6G1 73]. De konkluderer med at telleren bør stå på 350 grader for å ligne på figuren i oppgaven.

O6G1-71	Elev 1	Er ikke det 90 grader?
O6G1-72	Elev 2	Jo
O6G1-73	Elev 1	Æh, da vi må har 200 på 180 grader, og 270 på 300. 350! Vi prøver det.
O6G1-74		*Setter inn 350 i løkka og tester programmet.*
O6G1-75	Elev 1	Det er jo brukende.

Elevene prøver seg frem med forskjellige verdier på blokka som indikerer i hvilken retning programmet tegner. De lander til slutt på 45 grader [O6G1 89-91].

O6G1-89	Elev 1	Nei, nå har jeg det. Litt så [*setter verdien til 45 grader*]
O6G1-90		*Tester programmet*
O6G1-91	Elev 1	Ja, se der.

Elevene utvider programmet med en ny løkke for å tegne øyet. Etter litt utforskning ender de opp med 55 som y-verdi for koordinatene [O6G1 117].

O6G1-117	Elev 1	Da setter vi 55 på den [*y-verdien av koordinatblokka*] så får vi den litt høyere.

Til sist tilpasser de størrelsen på øyet ved å endre på sted-blokka til 5 og sier seg fornøyd med løsningen sin [O6G1 135-136].

O6G1-135		*Endrer antall steg til 5 og tester programmet.*
O6G1-136	Elev 2	Sånn ja. Perfekt!

4.2.2 Gruppe 1 – Instrumentell genesis

I arbeid med deloppgave 1 danner elevene et mål for aktiviteten som innebærer at de skal lage et program som tegner en trekant. Elevene prøver seg frem med hvilke blokker som er nødvendige og hvilken rekkefølge de skal stå i. De henter inspirasjon fra eksempel 2 i oppgaveteksten for hvordan programmet kan bygges.

I denne prosessen synliggjøres elevenes handlingsregel for hvordan de kombinerer antall sider og størrelser på vinkler i mangekanter. Elev 2 foreslår å legge inn verdien 45, men testing av programmet viser at dette ikke gir ønsket form på mangekanten. Elev 1 knytter sin matematiske forståelse av at likesidete trekanter har vinkler på 60 grader. Dersom programmet skal tegne neste vinkelbein med en vinkel på 60 grader må programmet legge

inn en rotasjon på 120 grader. Det kommer ikke tydelig frem gjennom dialogen hvorfor han velger å gjøre dette gjennom en operatørblokk [O6G1 14] fremfor å bruke et konkret tall, men det letter arbeidet når de skal utvide programmet til å lage firkant [O6G1 20] og sekskant [O6G1 25]. Denne teoremforståelsen av vinkelstørrelse i mangekanter fungerer som en operasjonell invariant for måten elevene løser denne delen av oppgavene på.

I oppgave 2 er mål for aktiviteten at elevene skal lage et program som tegner en regulær mangekant, der antallet sider tilsvarer en inndata som programmet spør etter. Elev 1 løser dette ved å sette inn en spørre-blokk i forkant av løkka [O6G1 34]. Elev 1 viser her en konseptforståelse av at programmet må inneholde en variabel som brukeren får mulighet til å definere gjennom inndata. Måten elev 1 bruker variabelen viser også en teoremforståelse av hvordan variabler bidrar til en generalisering av matematiske uttrykk. Dette programmet har bare én variabel og denne knyttes opp mot verdien til svar-blokka. Det er dermed ikke nødvendig å opprette og definere denne med en egen variabelblokk, slik som i oppgave 4.
I oppgave 3 er mål for aktiviteten å lage et program som tegner en Pac-man. Elevene starter med å bygge programmet med utgangspunkt i eksemplet fra oppgaveteksten. Det videre arbeidet med kodingen bærer preg av stor grad av prøving og feiling. De forsøker å gjøre endringer i de forskjellige blokkene for så å teste ut hva som skjer ved å kjøre programmet. [O6G1 47, 51,53,56] Det kommer ikke frem av observasjonene at de i denne fasen gjør noen tydelige forsøk på å diskutere hva de forskjellige blokkene gjør, eller kommer med antakelser om hvordan programmet påvirkes om blokkene endres. Handlingsregelen som kommer frem i denne situasjonen er å teste hyppig av programmet og analysere endringene som skjer. Det kan virke som elevene velger å jobbe på denne måten for å lære seg hva de forskjellige blokkene gjør, og hvordan de påvirker programmet. De operasjonelle invariantene som ligger til grunn for disse handlingene er trolig en forståelse om at blokkene sin funksjon kan synliggjøres ved å gjøre små endringer, for så å observere resultatet gjennom hyppig testing av programmet.

O6G1-45	Elev 2	Da må vi ha den. Og den. Og den. Og den blå. [*eleven bygger opp programmet for stjerne slik det står i eksemplet i oppgaven*]
O6G1-46	Elev 1	Hvis vi prøver det nå.
O6G1-47		*De tester programmet*
O6G1-48	Elev 1	Vi må ha litt flere steg., [*endrer til 50*]
O6G1-49	Elev 2	Men hvordan kan vi få flere streker?

O6G1-50	Elev 1	Hva om vi gjentar 50 ganger?
O6G1-51		*Tester programmet som kun legger strekene over hverandre.*
O6G1-52	Elev 1	Og hvis vi endrer den [*rotasjons-blokka*] til 10 grader.
O6G1-53		*Tester programmet*
O6G1-54	Elev 1	Åh! Jeg tror dette kan gå faktisk.
O6G1-55	Elev 1	Hvis vi tar 0.1 [*i rotasjonsblokka*]
O6G1-56		*Tester programmet*
O6G1-57	Elev 1	Da må vi endre her. [a*ntall repetisjoner i løkka. Setter verdien til 100000*] Sånn!

4.2.3 Gruppe 2 – Observasjon

Gruppe 2 starter med å lage et program som tegner en trekant. I likhet med gruppe 1, benytter de seg også av en operatørblokk med divisjon, 360/3. Når de skal lage firkant endrer de divisoren i operatørblokka og løkketelleren til 4 [O6G2 31-34].

O6G2-31	Elev 4	Vi lager heller firkant.
O6G2-32	Elev 4	da er det fire. [*Endrer divisor til 4*]
O6G2-33	Elev 4	Er det fire der også? [*Henviser til løkka*]
O6G2-34	Elev 5	Jo

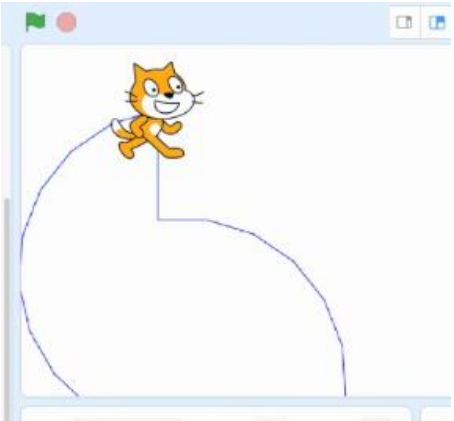
Samme fremgangsmåte bruker de når de skal tegne sekskant [O6G2 39].

O6G2-39	Elev 4	Ja, da må vi vel bare bytte på disse tallene. 6 og 6. [Endrer divisor og løkketeller til 6]

På spørsmål om hvilke andre mangekanter de skal lage blir de enige om å lage en 21-kant. De ender løkketeller og divisoren til 21. Tegningen programmet produserer blir ikke slik de forventet [O6G2 44].

O6G2-44	Elev 4	Det funket ikke. Hvorfor funket ikke det?

Se figur 14 for skjermbilde av programmet.



Figur 14: Skjermdump av visningsvinduet fra gruppe 2 sitt arbeid med mangekanter.

Elevene undres over hvorfor ikke programmet tegner en mangekant. De forsøker med lavere tall [O6G2 52-53].

O6G2-52	Elev 4	Vi prøver med 12. [*Endrer divisor og løkketeller til 12 og steg tilbake til 50*]
O6G2-53	Elev 4	Det ble rett. Hvorfor fungerte det ikke med 21?

Programmet fungere som ønsket med 12 som inndata. De prøver deretter med å tegne en 15-kant, men visningen av denne blir også mislykket [O6G2 54-55].

O6G2-54		*Endrer divisor og løkketeller til 15 og kjører programmet.*
O6G2-55	Elev 4	Der gikk den i skjeis. Hvorfor gjorde den det?

Videre undersøkelser viser at den største mangekanten programmet klarer å lage er 13-kant [O6G2 62-64]. Elevene slår seg til ro med dette og går videre på neste deloppgave.

O6G2-62	Elev 4	Jeg må bare sjekke det siden 13 fungerte.
O6G2-63		*Tester programmet*
O6G2-64	Elev 4	Ja! Det fungerer opp til 13- kant, ikke lengre.

I deloppgave 2 diskuterer elevene seg frem til at de må ha med en spørre-blokk for å lage et program som løser oppgaven [O6G2 71].

O6G2-71	Elev 4	Da må vi ha spørreblokka.

Elevene setter inn spørreblokka i kodene. Etter hvert oppdager de at de også må bytte ut løkketeller og divisor med svar-blokka slik at det programmet tar med variabelen i beregningene [O6G2 92-97].

O6G2-92	Elev 5	Vi må bruke svar-blokka tror jeg.
O6G2-93	Elev 4	Ja, sånn ja! Du er smart! [*Endrer divisor og løkketeller til svar-blokk*]
O6G2-94	Elev 4	Si et tall
O6G2-95	Elev 5	5
O6G2-96		*Tester med 5 som input*
O6G2-97	Elev 4	Jada!

I deloppgave 3 tar elevene utgangspunkt i eksempel 3 i oppgaveteksten og bygger videre på disse kodene. De konkluderer med at programmet deres gjenta løkka flere ganger [O6G2 121].

O6G2-121	Elev 4	Litt rar gulfarge, men det går jo fint. Så må det gjentar mange flere ganger. Vi prøver 100.

I likhet med det som skjedde for gruppe 1 endrer ikke dette resultatet av figuren siden strekene tegnes oppå hverandre i de samme punktene [O6G2 124-125].

O6G2-124	Elev 4	Men det blir jo ikke gjentatt 100 ganger.
O6G2-125	Elev 4	Åja, de kommer på samme plass. Hæ? Nå skjønner jeg ikke. Skjønner meg ikke på Pac-man.

Gjennom videre arbeid med oppgaven forstår elevene at de må endre rotasjonen for å få linjene i figuren tettere [O6G2 129].

O6G2-129	Elev 4	Åh, det har noe med snu 36 grader som har noe å si for hvor nær den skal være. Snu 1 grad heller. Da blir den mye tettere.

På dette tidspunktet har elevene dobbelt opp med noen av blokkene i løkka. De innser at dette trolig ikke er nødvendig. De foreslår å fjerne de løkkene som er overflødige [O6G2 168-169]. Dette gjøre at programmet tegner en sektor som bare har halvparten så stor vinkel som tidligere. Elevene løser dette med å øke løkketelleren. Da oppdager de sammenhengen mellom vinkelen ved rotasjonen og løkketelleren, og at 360 roteringer med 1 grad blir en sirkel [O6G2 175].

O6G2-168	Elev 4	Trenger vi egentlig denne delen? [*Henviser til at den opprinnelige løkka har dobbelt opp med noen blokker*]
O6G2-169	Elev 4	Skal vi prøve uten den? Bare for å se hvordan det er?
O6G2-170		*Tester programmet*
O6G2-171	Elev 4	Tror bare han lager det samme og at de blokkene er unødvendige. Nei, det gjorde den ikke. [*Programmet tegner halvparten så stor sektor som tidligere*]
O6G2-172	Elev 4	Da må jo bare løkka gjentas flere ganger. Det er jo bare det.
O6G2-173	Elev 4	Vi endrer den til 260 ganger, i stedet for 160.
O6G2-174	Elev 4	Nei, litt til. 300
O6G2-175	Elev 4	Åh! 360 blir vel en sirkel. [*Det går opp for eleven at antall løkker korrelerer med grader*]

Elevene lager egen løkke for at programmet skal tegne øyet [O6G2 187]. De justerer størrelsen på øyet til 10 steg og sier seg fornøyd med besvarelsen når de ser resultatet til programmet.

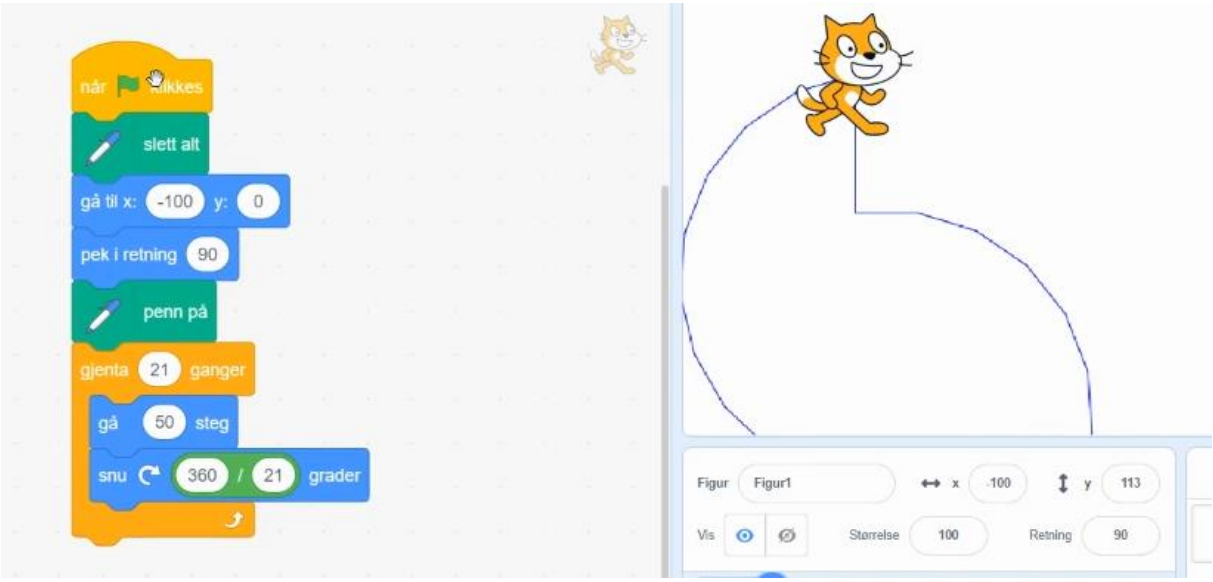
O6G2-187	Elev 4	Oi, den ble liten. Vi prøver heller med 10 steg.
O6G2-188	Elev 4	Er ikke det litt stort da?

O6G2-189	Elev 5	Neida.
O6G2-190	Elev 4	Dette ble fint.
O6G2-191	Elev 5	Bra jobba!

4.2.4 Gruppe 2 – Instrumentell genesis

I oppgave 1 c) har elevene som mål for aktiviteten å endre et program fra å tegne firkanter til å tegne sekskant. Elevenes handlingsregel kommer til syne ved at de endrer verdiene i løkketelleren fra 4 til 6 [O6G2 39]. Dette medfører at løkken gjentas seks ganger slik at det lages seks sider. Videre endrer de også divisoren i operatørblokk fra 4 til 6 [O6G2 39]. Dette medfører at programmet roterer 60 grader (360/6=60) slik av vinkelen mellom vinkelbeinene blir 120 grader, som tilsvarer vinklene i sekskanten. Elevene viser her en teoremforståelse for forholdet mellom antall sider i en mangekant og vinklene mellom dem. De viser også konseptforståelse ved at de kjenner til hvordan løkker og operatørblokker fungerer i programmeringsspråket og hvordan disse kan brukes til å utføre matematiske beregninger.

Når elevene jobber videre med neste deloppgave støter de på et problem. De blir utfordret på hvilke andre mangekanter de kan lage med utgangspunkt i programmet sitt. De foreslår å lage en 21-kant [O6G2 41-42]. Målet blir da å endre programmet slik at det tegner en 21-kant. De endrer divisoren og løkketelleren til 21. Programmet testes, men resultatet blir ikke slik elevene forventet. Figuren under viser resultatet da de testet programmet med 21 som inndata.



Figur 15: Gruppe 2 sine koder og visningsvindu fra arbeidet med mangekanter.

De fulgte samme handlingsregel som forrige oppgave. Elevene stusser over at det programmet ikke fungerer slik de forventet [O6G2 44-49].

O6G2-44	Elev 4	Det funket ikke. Hvorfor funket ikke det?
O6G2-45	Elev 5	Har det noe med stegene hans å gjøre?
O6G2-46	Elev 4	Du ser jo inne der så er det mange små hakk. Men det ble jo ikke rett.
O6G2-47		*Prøver programmet flere ganger, med samme resultat.*
O6G2-48	Elev 4	Hva skjer a? Hvorfor fungerer ikke dette?
O6G2-49	Elev 5	Vet ikke

Elevene velger da å endre programmet til å lage en mangekant med færre sider. De lager en 12-kant [O6G2 52]. Denne gangen fungerer programmet slik de ønsker. Grunnen til at programmet mislykkes i å tegne 21-kanten er ikke at det er noe feil med kodene. Problemet oppstår i det den tegnede figuren kommer utenfor visningsvinduet (vinduet katten kan bevege seg i). Dette er ikke en utfordring elevene har vært borti ved tidligere oppgaver, så de har ikke kjennskap til denne begrensingen ved programmeringsspråket. Hadde vinduet vært så stort at hele figuren kunne tegnes uten å komme utfor ville figuren blitt en regulær 21-kant. Det kommer ikke frem av observasjonene at de forstår at det er denne begrensningen ved programmeringsspråket Scratch som er utfordringen. Elevene utforsker videre for å få en forståelse av hvor stor mangekant programmet klarer å tegne. De med at det klarer å tegne figurer med opptil 13 sider [O6G2 64]. Elevene aksepterer at programmet ikke klarer å tegne mangekanter over 13-kant uten å diskutere dette ytterligere. De opplever at deres handlingsregel for å tegne større mangekanter ikke fører frem. Det ligger en manglende konseptforståelse til grunn for denne utfordringen.

I deloppgave 3 danner elevene et delmål i arbeid med å tegne Pac-man. De forsøker å gjøre sin figur tettere, altså at det blir mindre vinkel mellom strekene i figuren. De gjør endringer i flere av blokkene for så å teste programmet for å analysere hvordan endringene påvirker resultatet. Etter noe utforskning oppdager Elev 4 at det er en sammenheng mellom gradene i rotasjonsblokka og tettheten til strekene [O6G2 129]. Denne forståelsen bygger de videre på i arbeidet med programmet, og knytter det opp mot gradene til en sirkel. De bruker da sin matematiske forståelse til å beregne sirkelsektorens størrelse fremfor å prøve seg frem med

forskjellige verdier på løkketellerne. Valg av handlingsregel begrunnes ved en økt teoremforståelse av programmeringsspråket [O6G2 175-176].

4.2.5 Oppgave 6 – Kjerneelementene

I de følgene avsnittene vil det beskrives hvordan kjerneelementene kommer til syne når gruppene jobbet med oppgave 6.

Utforskning og problemløsning

En del av dette kjerneelementet legger føringer for at elevene *lete etter mønster og sammenhenger*. Funn viser at dette er noe elevene stort sett gjør gjennom hele arbeidsøkta. Et konkret eksempel på dette kan observeres når elevene i gruppe 1 jobber med deloppgave 3, tegning av Pac-man. I arbeidet med å lage en sammenhengende sirkel konkluderer de med av løkken må gjentas mange ganger og rotasjonsgraden må være liten. Rotasjon er satt til 0,9 grader. De prøver med 100 løkker. Testing av programmet viser en sektor som tilsvarer 90 grader [O6G1 69-72]. Elevene ser da en sammenheng mellom gradene og antall ganger løkka kjører. Elev 1 beregner da at sektoren blir 180 grader om løkke repeteres 200 ganger og 270 grader om den repeteres 300 ganger. I likhet med gruppe 1 oppdager elevene i gruppe 2 egenskaper ved figurens rotasjon ved at de knytter sektorstørrelsen opp mot løkketeller og rotasjonsgrad [O6G2 174-175].

En annen del av dette kjerneelementet er at elevene skal *diskutere seg frem til en felles forståelse*. Dette er også noe som er dekkende for store deler av arbeidsøkta, med bakgrunn i at elevene jobber sammen om oppgavene. Konkrete eksempler på at dette kommer frem, kan observeres når elevene jobber med å plassere retningen på munnen til Pac-man i deloppgave 3.

Elevene i gruppe 1 har forstått at *snu*-blokka har påvirkning på hvor plasseringen til munnen blir, altså hvilken del av sektoren som ikke fargelegges. De diskuterer seg frem til en felles forståelse av hvordan rotasjonen påvirker dette [O6G1 76-89]. Elevene konkluderer til slutt med at det den må være på omtrent 45 grader for å tilsvare figuren i oppgaven.

Konkrete eksempler på at dette kommer frem i gruppe 2 sitt arbeid kan observeres når de skal tegne 21-kanten. Elevene diskuterer, og prøver seg frem, for å avdekke hvorfor ikke programmet tegner figuren slik de forventer. I denne situasjonen konkluderer de med en felles forståelse som er uriktig. De mener programmet bare fungerer opp til 13-kanter [O6G2 64].

Modellering og anvendelse

I dette kjerneelementet skal elevene blant annet *bruke sin matematiske forståelse til å løse problemer*. For elevene i gruppe 2 kommer dette frem i eksemplet nevnt over, der de beregner størrelsen på sirkelsektoren [O6G2 174-175]. Når elevene først klarer å relatere programmeringen opp mot sin matematiske forståelse av gradene i en sirkel, bruker de denne forståelsen til å løse oppgaven.

Dette kjerneelementet kommer også til syne gjennom arbeidsøkten til gruppe 1. Eksempelvis når Elev 1 bruker sin forståelse av vinkelsum til å definere hvordan operatørblokkene skal beregne vinklene i figuren [O6G1 14].

Resonnering og argumentasjon

Elevene skal *utforme egne resonnementer*. Det er begrenset i hvor stor grad elevene uttrykker sine resonnementer i arbeid med disse oppgavene. De resonnerer trolig mye gjennom tankene, men finner det ikke naturlig å uttrykke det verbalt. I arbeidet med å få størrelsen på Pac-man sin munn i deloppgave 3 utformet Elev 1 et resonnement for hvordan antall repetisjoner i løkka påvirker størrelsen på sektoren [O6G1 73].

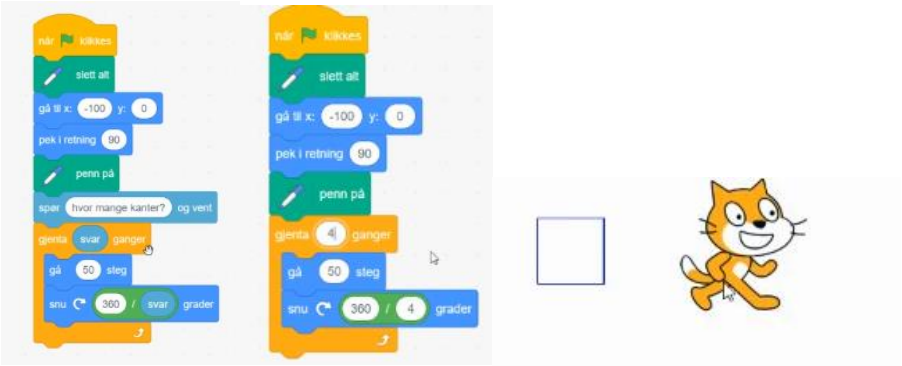
Elevene i gruppe 2 utformer egne resonnement i arbeidet med deloppgave 1a), der de skal lage et program som tegner en trekant. De henter inspirasjon fra et av eksemplene og kopierer inn operatørblokka som inneholder beregningen 360/5 (siden dette programmet tegner en femkant). Elev 4 innlemme dette i dere program direkte, uten å gjøre endringer. Elev 5 gjør refleksjoner rundt dette og foreslår at de skal endre divisoren til 3 [O6G2 22]. Hun har trolig resonnert seg fremt til forholdet mellom antall kanter og divisoren i operatørblokka. Elev 4 fanger ikke opp hva elev 5 mener med en gang, men forstår det når hun knytter det opp mot figuren programmet tegner [O6G2 25].

O6G2-22	Elev 5	Blir det ikke 3?
O6G2-23	Elev 4	360 delt på 5 grader. Jeg vet ikke helt.
O6G2-24		*Forsøker å kjøre programmet.*
O6G2-25	Elev 4	Han tegnet bare en halv. Han må snu… Åja, det var det du mente med delt på 3!
O6G2-26	Elev 5	Ja

Representasjon og kommunikasjon

I kjerneelementet representasjon og kommunikasjon skal elevene *uttrykke matematiske begreper, sammenhenger og problem*. Matematiske sammenhenger blir uttrykt i eksemplet som er beskrevet over, når elev 1 uttrykker sammenhengen mellom vinkelsum og de enkelte vinklene [O6G1 14] og elev 4 uttrykker sammenhengen mellom vinkelsum og de enkelte vinklene [O6G2 173-174].

Kjerneelementet legger også føringer for at elevene skal *bruke ulike representasjoner* når de jobber med matematikk. I arbeidet med disse oppgavene blir mangekanter representert som tegnede figurer. De samme figurene ble samtidig representert gjennom koder med aritmetiske eller algebraiske beregninger. Bildet under er skjermdump fra arbeidet til gruppe 2 og synliggjør ulike representasjonsformer.



Figur 16: Skjermdump fra gruppe 2 sitt arbeid som viser ulike representasjonsformer.

Et annet element ved dette kjerneelementet er at elevene skal *bruke matematisk språk når de diskutere*. Funn tyder på at elevene har en vei å gå når det gjelder å bruke matematisk språk når de diskuterer, og arbeider, med matematikk. Eksempelvis bruker elev 2 begrepet *runding* i stedet for sirkel [O6G1 42].

O6G1-42	Elev 2	Oi, det ser nesten ut som en runding.

Gruppe 2 bruker i større grad enn gruppe 1 et matematisk språk i diskusjonen. Der gruppe 1 bruke ord som *runding*, bruker gruppe 2 konsekvent *sirkel* [O6G2 59-60,175].

Abstraksjon og generalisering

En del av dette kjerneelementet omfatter generalisering. Elevene skal *formalisere sammenhenger ved å bruke algebra og hensiktsmessige representasjoner*. I oppgave 2 utfordres elevene på å generalisere programmet slik at det tegner alle mangekanter på bakgrunn av tallverdien som velges for inndata. I programmeringsspråket kan denne generaliseringen gjøres ved å introdusere variabler i kodene. Elev 1 generaliserer programmet fra oppgave 1 ved å sette inn en spørre-blokk, som i praksis er en variabel [O6G1 34-39]. På denne måten generaliseres innholdet i programmet.

O6G1-34	Elev 1	Vent. Spør hvor mange. [*setter inn spørre-blokk og endrer spørsmålet slik at det er tilpasses oppgaven*]
O6G1-35	Elev 2	Sånn.
O6G1-36	Elev 1	Og vent. Jeg er ikke helt sikker på hva "og vent" betyr.
O6G1-37	Elev 1	Ah, jeg tror vi har det nå. Vi setter inn svar [*sett inn svar-blokk i løkka og som divisor*]
O6G1-38	Elev 2	Vi prøver med 10.
O6G1-39	Elev 1	1,2,3,4,5,6,7,8,9,10. [*Teller at det blir en tikant*]

Matematiske kunnskapsområder

Samtlige oppgaver i denne arbeidsøkten er tungt representert innen geometri. Oppgave 1 og 2 tegner mangekanter. På oppgave 3 utfordres elevene til å tegne en Pac-man som består av en sirkelsektor (hodet) og en sirkel (øyet). Gjennom generaliseringsprosessen som er beskrevet over så er også kunnskapsområdet algebra representert.

5 Drøfting

Denne kasusstudien bygger på to elevgruppers arbeid med to programmeringsoppgaver som en del av matematikkundervisningen. Til tross for et relativt begrenset datagrunnlag er det komme frem interessante observasjoner som vil diskuteres i dette kapitlet.

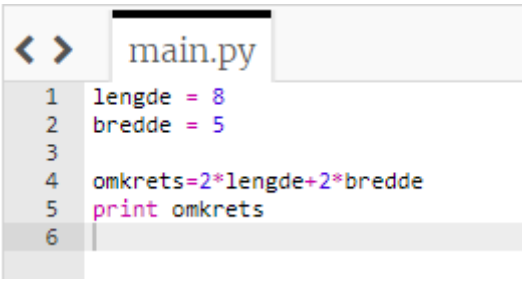
5.1 Instrumentell genesis

Fra analysedelen med utgangspunkt i instrumentell genesis er det tre hovedfunn som vil løftes frem i drøftingen.

1. *Regnerekkefølge og aritmetiske grunnprinsipper i programmering* der gruppenes arbeid med operatørblokkene i oppgave 4 byr på utfordringer.
2. *Testing som handlingsregel* der gruppe 1 i arbeid med oppgave 6 bruker hyppig testing for å undersøke hvordan de enkelte blokker påvirker programmet.
3. *Vindustrøbbel* der gruppe 2 i arbeid med oppgave 6 har utfordring når figuren kommer utenfor visningsvinduet til programmet.

5.1.1 Regnerekkefølge og aritmetiske grunnprinsipper i programmering

I arbeidet med oppgave 4 viser begge gruppene at de har god matematisk forståelse, men møter utfordringer i arbeidet med å endre kodene slik at programmet beregner omkretsen. Gruppe 1 begrunner sitt oppsett av blokker med en forventning om at programmet behandler regneoperasjonene på samme måte som de kjenner fra den matematikkundervisningen de er vant med. De viser med dette at de har god forståelse av hvordan den matematiske beregningen skal gjøres. De er ikke bevisste på konseptet ved programmeringsspråket Scratch, der blokkene fungerer som parenteser. Dette konseptet er ikke gjeldende for alle programmeringsspråk. Hadde elevene kodet programmet i Python ville deres argumentasjon og oppsett av koder løst oppgaven slik de ønsker. Figuren under viser hvordan beregningene kan kodes i programmeringsspråket Python. Sammenlignet med andre programmeringsspråk har Scratch på denne måten en begrensning når det kommer til aritmetiske og algebraiske beregninger.



Figur 17: Koder fra programmeringsspråket Python som beregner omkretsen av et rektangel.

Gruppe 2 har en noe annerledes begrunnelse for måten de koder programmet sitt. De antar at programmet utfører beregningene i den rekkefølgen de står, selv om de senere i arbeidsøkten synliggjør at de har forståelse av prioriteringsrekkefølgen. Det er forståelig at elevene tenker på denne måten, med tanke på hvordan kodene organiseres i programmering. I programmering er rekkefølgen av kodene viktig, og programmet leser kodene i fast rekkefølgen slik de står vertikalt nedover. Da er det forståelig at elevene tenker at denne rigiditeten gjelder for de enkelte kodene i horisontal retning også. Elevene avdekker at dette ikke er tilfelle gjennom validering av svarene sine. På denne måten bruker elevene sin matematiske forståelse til å avdekke konsepter ved programmeringen.

5.1.2 Testing som handlingsregel

I arbeid med oppgave 6 bruker gruppe 1 hyppig testing av programmet som en handlingsregel for å utforske hvordan de forskjellige blokkene påvirker programmet. Dette fortoner seg gjennom å gjøre én endring, teste programmet og analysere endringen og vurdere resultatet. Dette er en handlingsregel som også er kommet frem i annen forskning. Buteau et al. (2019) analyserer Jim sitt arbeid med programmering og matematikk som på dette tidspunktet er første års student på et bachelor-studium og har lite erfaring med programmering. Jim utfordres på å beskrive fremgangsmåten for å test og validere programmet sitt svarer han «Simple put (…) basically by running it. I would test each kind of function in isolation to make sure it's working properly.» (Buteau et al., 2019, s. 1032).

5.1.3 Vindustrøbbel

Analysen av gruppe 2 sitt arbeid med oppgave 6 fremhever situasjonen der elevene har generalisert kodene slik at programmet kan tegne alle slags regulære mangekanter. Elevene virker trygge på matematikken som ligger bak kodene og blir derfor overrasket når

programmet ikke fungerer på figurer med flere enn 13 sider. Dette avdekker en begrensning ved programmeringsspråket Scratch ved at pennen ikke klarer å følge koordinatene når den kommer utenfor visningsvinduet i nettleseren. Denne begrensningen forvirrer elevene og bidrar til at de konkluderer med at deres antatte generelle kode ikke er generell likevel. Sett i lys av komponentene i skjemautvikling fremstår denne begrensningen som et hinder for at elevene skal kunne trekke hensiktsmessige slutninger. I denne situasjonen bidrar programmet til at elevene trekker feil slutning. Det er teknisk mulig for elevene å tegne 21 kant også, men da måtte de kortet ned sidelengdene, samt sentrert figuren i origo slik at hele figuren kommer innenfor visningsvinduet.

I en naturlig undervisningssituasjon, med lærer til stede, ville trolig denne begrensningen blitt fremhevet og forklart. Det kunne til og med vært et godt utgangspunkt for matematisk diskusjon om generalisering. Misfeldt og Ejsing-Duun (2015) fremhever lærerens tilnærming og rolle som svært viktig når elevene skal utvikle matematisk forståelse gjennom programmering. Situasjonen over bidrar som et eksempel der lærerens rolle kan være avgjørende for elevenes forståelse og mulighet for å trekke slutninger.

5.1.4 Instrumentalisering og instrumentering

Første forskningsspørsmål av denne studien var å belyse elevers arbeid med instrumentell genesis som rammeverk. Prosessene instrumentering og instrumentalisering beskriver hvordan verktøyet og subjektet, i denne studien programmeringsspråket Scratch og elevene, påvirker hverandre. Observasjoner av elevenes arbeid viser at disse to prosessene ikke er representert i like stor grad i elevenes arbeid. Funn tyder på at deres faglige utbytte hovedsakelig knyttes til å lære seg å programmere, og utforske muligheter og begrensninger i programmeringsspråket. Dette skjer i form av at elevene bruker sin matematikkforståelse til å øke forståelsen og bruksområdet av programmet. Det matematiske utbyttet ser ut til å være representert i mindre grad. Prosessene i elevenes arbeid har på bakgrunn av dette i stor grad form som instrumentalisering.

5.1.5 P-skjema og m-skjema

Basert på analysen av denne studien tyder det på at elevene i stor grad har utviklet sin forståelse av programmeringsspråket Scratch, og i mindre grad utvidet sin matematiske forståelse. Sett i lys av instrumentell genesis er det instrumenteringsprosessen som kommer tydeligst frem i elevenes arbeid. Det kan tyde på at den nivåmessige skjevheten mellom elevenes matematikkforståelse og programmeringsferdighet fører til at elevene får begrenset matematisk utbytte av arbeidet med programmering. Resultatet av arbeidet fortoner seg da hovedsakelig som økt forståelse av de muligheter og begrensninger som ligger i programmeringsspråket. Dette oppleves som utfordrende med tanke på at læreplanens intensjon med å introdusere programmering i matematikkfaget nettopp er at elevenes skal bruke programmeringen til å utvikle matematisk forståelse. For at dette skal bli en realitet må trolig skjevheten mellom matematikk- og programmeringsforståelse hentes inn ved at elevene i større grad bruker programmering som et instrument. Sagt med andre ord kan det tyde på at elevene må ha en viss kompetanse innen programmering før de kan bruke dette som et instrument til å lære matematikk.

Læreplanen har introdusert kompetansemål rettet mot programmering så tidlig som 4. trinn. Dette medfører at hvert kull som kommer til ungdomsskolen de neste årene bør ha mindre grad av skjevheten beskrevet over. Dette øker elevenes forutsetning for å kunne bruke programmering som et instrument til å lære matematikk på ungdomsskolen, slik læreplanen legger opp til (Utdanningsdirektoratet, 2020). Dette kan diskuteres opp mot Buteau et al. (2020) sin introduksjon av p-skjema og m-skjema. Sees analysen fra denne kasusstudien opp mot dette rammeverket vil elevenes skjemautvikling hovedsakelig ligge på nivå 2. Elevene utvikler konseptforståelse gjennom arbeid med matematikkoppgaver og det er i stor grad p-skjema som utvikles. For å oppnå kompetansemål for 10. trinn i læreplanen må elevene trolig opp på nivå 3 og 4 i dette rammeverket, slik at teoremforståelsen utvikles ved at m-skjema i større grad tas i bruk.

Med utgangspunkt i drøftingen over, kan inndelingen av operasjonelle invarianter kunne relateres til Buteau et al. (2020) sin introduksjon av p- og m-skjema. Konseptforståelsen av

de operasjonelle invariantene vil da kunne relateres til Buteau et al. (2020) sin beskrivelse av p-skjerma, og teoremforståelse og vil være nært knyttet til m-skjema.

5.2 Kjerneelementene

I dette delkapitlet vil forskningsspørsmål 2 bli diskutert. Kort oppsummert er kjerneelementene i stor grad representert i elevenes arbeid. Funnene viser at de jobber i tråd med læreplanens beskrivelse av kjerneelementene. Dette skjer til tross for at de i all hovedsak jobbet uten tilsyn og veiledning av lærer, og at de har begrenset erfaring med programmering fra tidligere. Funn fra denne studien tyder på at programmering kan fungere som et hensiktsmessig verktøy i matematikkfaget. Dette kommer til syne ved måtene elevene må representere og generalisere matematikken på. Programmering kan bidra til å fremme matematikkforståelse og utføre komplekse og tidkrevende beregninger. Til tross for at potensialet er der, anses det ikke som en selvfølge at elevene utnytter disse mulighetene på en hensiktsmessig måte. Analysen av elevenes arbeid kan tyde på at didaktiske vurderinger til designet av undervisningsopplegget er avgjørende for måten kjerneelementene kommer til syne. I videre drøfting vil didaktiske vurderinger fra planleggingen av undervisningsopplegget belyses og diskuteres.

Problemløsning og algoritmisk tenkning

Problemløsning handler om at elevene skal utvikle metoder for å løse utfordringer de ikke kjenner fra før (Utdanningsdirektoratet, 2020). Åpne oppgaver kan bidra til at elevene jobber i tråd med prinsippene for problemløsning. I arbeidet med å designe undervisningsopplegget ble oppgave 6-3 designet for at elevene skulle utfordres på denne delen av kjerneelementene. De ble utfordret til å kode et program som tegner en figur som ligner på Pac-man, uten mulighet for å bygge videre på eksisterende koder. En utfordring med slike oppgaver kan være at elevene ikke makter å drive arbeidet fremover. I designet av oppgaven ble det derfor lagt ved noen hint som kan hjelpe de i denne prosessen. Analysen fra studien viser at begge gruppene løste denne oppgaven på en god måte.

Algoritmisk tenkning er sentral i kjerneelementet utforskning og problemløsning. I kjerneelementene beskriver Utdanningsdirektoratet (2020) at elevene skal utvikle strategier og fremgangsmåter for å bryte ned og løse problemer. Dette innebærer blant annet å bryte problemer ned til delproblemer. Programmering kan fungere som et hensiktsmessig verktøy til å utvikle kompetanse innen algoritmisk tenkning. Kodene kan isoleres i mindre bestanddeler og kan til enhver tid testes og analyseres for syntaksfeil eller visuelle feil. Funn fra gruppe 1 sitt arbeid med oppgave 6 tyder på at elevene jobber i tråd med algoritmisk tankegang ved måten de tester hyppig for å analysere enkelte sekvenser av kodene.

I oppgavedesignet er det lagt til rette for algoritmisk tenkning ved at de fleste oppgavene innledes med en diskusjonsdel der elevene utfordres på å diskutere seg frem til en felles forståelse av kodene funksjon i programmet. Ved å diskutere hva de enkelte kodene gjør, og hvordan de påvirker programmet, legges det til rette for at elevene jobber i tråd med algoritmisk tankegang.

Generalisering

Generalisering handler om at elevene oppdager sammenhenger og strukturer, og uttrykker disse gjennom algebra og hensiktsmessige representasjoner (Utdanningsdirektoratet, 2020). Funn fra denne studien viser at arbeid med programmering kan fremme, men også i noen situasjoner, hemme elevenes arbeid med å generalisere. Bruk av variabler er en naturlig del av programmeringsarbeidet. Elevene tvinges til å generalisere ved at de må forholde seg til variablene når de programmerer. I denne studien synliggjøres dette blant annet ved gruppenes arbeid med oppgave 4, der de må beregne omkretsen av rektangler ved bruk av variabler for lengde og bredde. Oppgave 6 legger opp til at elevene jobber i tråd med dette kjerneelementet ved at elevene må generalisere kodene når programmet skal kunne tegne alle mangekanter.

Denne studien viser også at begrensninger i programmeringsspråket kan bidra til å hindre elevenes arbeid med generaliseringen. Dette synliggjøres i gruppe 2 sitt arbeid med oppgave 6. Elevene konkluderer med at deres, korrekte, koder er feil. Dette baserer de på den visuelle fremstillingen i programmeringsspråket. Denne egenskapen ved visningsvinduet er

ikke tatt høyde for i designet av oppgaven. Denne situasjonen avdekker en begrensning i programmeringsspråket Scratch som det bør tas høyde for i videre arbeid med oppgavedesign.

Modellering

Modellering er en del av kjerneelementet modellering og anvendelse. Modelleringsdelen av kjerneelementet er ikke representert gjennom elevenes arbeid i denne studien. Modellering er et begrep som defineres forskjellig i litteraturen, med varierende grad av dybde. Blum (2015) beskrivelse modellering med en 7-trinns-modell. Utdanningsdirektoratet (2020) har en noe løsere definisjon der modellering defineres ved at elevene skal beskrive virkelige situasjoner på en matematisk måte. Uavhengig av hvilken definisjon som brukes for modellering vil denne arbeidsmåten i stor grad være avhengig av oppgavens design. Oppgave 4 og 6, som danner grunnlaget for datamaterialet, er ikke utformet på en måte som tilsier at elevene jobber etter prinsippene for modellering.

Til tross for at modellering ikke er representert i denne kasusstudien vil det påpekes av programmering kan være et hensiktsmessig verktøy for å jobbe med modelleringsoppgaver i matematikkfaget. I oppgave 5, se vedlegg 2, jobbet elevene med simulering av terningkast ved hjelp av programmering. Dette er en oppgave der en virkelig situasjon beskrives og effektiviseres gjennom programmeringen. Denne oppgaven vil derfor kunne være nærere knyttet til modellering.

Gruppearbeid

I denne kasusstudien har elevene jobbet i gruppestørrelser på 2-3. Bakgrunnen for at elevene jobber i en sosial læringssituasjon er at flere av kjerneelementene i læreplanen bygger på diskusjon og kommunikasjon. Arbeidsformen i denne studien er valgt for å legge til rette for at kjerneelementene som diskusjon, kommunikasjon og argumentasjon, opptrer som en naturlig del av elevenes arbeid. Datamaterialet i denne studien viser hvordan elevene kan diskutere seg frem til felles forståelse ved å jobbe på denne måten. Disse prosessene ville vært utfordrende å observere ved individuelt arbeid. Intervju under, eller i

etterkant av, arbeidsøkten vil trolig også kunne avdekke dette, men gjennom den sosiale læringsituasjonen skjer dette på en mer autentisk måte.

Funn i denne studien viser at elevene sin kommunikasjon bærer preg av knapphet i bruk av matematiske begreper i argumenteringen. Dette er ikke noe som trolig kan knyttes direkte til programmering som verktøy, men hva elevene er vant med fra tidligere undervisning i faget. Likevel er dette noe det bør reflekteres rundt når det skal planlegges og gjennomføres arbeid med programmering i matematikkfaget, og i prosessen med å designe oppgaver.

Remix eller bruk av nedlastede koder

Oppgavesettet brukt i denne studien gir elevene valget mellom å transkribere kodene fra oppgavene eller at de kan lastes ned via en lenke. Bakgrunnen for denne didaktiske vurderingen er todelt.

1. *Komme hurtig til matematikken.*
   Intensjonen med å jobbe med programmering er at det skal fremme matematisk forståelse. Dette gjøres blant annet ved å jobbe i tråd med kjerneelementene. I arbeidet med oppgave 4 ledes elevene hurtig inn i arbeidet med matematiske beregninger når de skal endre programmet til å finne omkretsen av rektanglet. I oppgave 6 ledes de raskt inn i generaliseringsprosessen med mangekantene.
2. *Senke terskelen og øke selvstendigheten.*
   Elevgruppen i denne studien har begrenset med programmeringserfaring. Remix, eller nedlastning av koder, kan senke terskelen til å komme i gang med den matematiske oppgaveløsningen, og kan bidra til at elevene arbeider mer selvstendig og med høyere grad av mestringsfølelse og læringsutbytte.

6 Avslutning

6.1 Oppsummering og konklusjon

Bakgrunnen for valg av tema er at programmering fra høsten 2020 er blitt en obligatorisk del av matematikkundervisningen i norsk skole. Dette har bydd på faglige og didaktiske utfordringer for matematikklærere, som må ta i bruk et verktøy de fleste har begrenset kjennskap til.

Målet med denne studien har vært å undersøke prosessen i elevenes arbeid med programmering i en matematisk kontekst. Gjennom første forskningsspørsmål ønskes det å undersøke hvordan elevene kan utvikle matematisk forståelse i arbeid med programmering. Instrumentell genesis er brukt som begrepsapparat for å systematisere og tolke disse komplekse og sammensatte prosessene. Forskningsspørsmål 2 tar for seg på hvilke måter kjerneelementene kommer til uttrykk når elevene arbeider med programmering.

I arbeidet med å undersøke forskningsspørsmålene er det utviklet et sett med seks oppgaver. To av disse danner grunnlaget for datamaterialet til denne studien. Datainnsamlingen er gjennomført ved bruk av skjerm- og lydopptak under elevenes arbeid.

Forskningsspørsmålene som har vært utgangspunkt for denne studien er:

1. **Hvordan kan elevenes arbeid med programmering forstås i lys av instrumentell genesis?**
2. **På hvilken måte kan kjerneelementene komme til syne gjennom elevers arbeid med programmering?**

Konklusjon forskningsspørsmål 1

Instrumentell genesis har bidratt med et hensiktsmessig begrepsapparat i arbeidet med å analysere observasjonene i denne studien. Komponentene i skjemautviklingen kan synliggjøres gjennom analysering av aktivitetsmål, handlingsregler, operasjonelle invarianter og muligheter for slutninger. Funn viser hvordan instrumenteringen og instrumentaliseringen kommer til syne gjennom analysering av hovedsakelig handlingsregler og de operasjonelle invarianter. Inndelingen av teoremforståelse og konseptforståelse

oppfattes som hensiktsmessige begreper til å beskrive hvordan elevene og programmeringsspråket påvirker hverandre.

Konklusjon forskningsspørsmål 2

Kjerneelementene er i stor grad representert i elevenes arbeid. Samtlige kjerneelementer, bortsett fra modellering, er identifisert i datamaterialet. Dette støtter opp om at programmering kan være et hensiktsmessig verktøy i matematikkfaget. Det kan konkluderes *at* kjerneelementene kommer til syne i elevenes arbeid med programmering. I studien drøftes det også *på hvilken måte* dette skjer. Funn indikerer at oppgavens design er en viktig faktor for at elevene arbeider i tråd med prinsippene fra kjerneelementene. Designelementer som bidrar til dette, er blant annet at elevene samarbeider og at de bygger videre på eksisterende koder gjennom remix fremfor å måtte lage programmene fra bunnen av.

6.2 Pedagogiske og didaktiske implikasjoner

I klasserommet vil det være en kontinuerlig refleksjon og vurdering av hva elevene gjør, hvordan de gjør det og hvorfor de gjør det på denne måten. Fra et lærerperspektiv er det nyttig å kunne gjenkjenne komponenter i elevenes mentale prosesser. Instrumentell genesis bidrar med et hensiktsmessig begrepsapparat som legger til rette for læreres arbeid med å identifisere og diskutere situasjoner fra undervisningen.

I løpet av denne studien er det avdekket noen begrensninger i programmeringsspråket Scratch. Oppgave 4 demonstrerer hvordan matematikken representeres på en annen måte i programmeringsspråket, enn slik elevene er vant med fra annen undervisning. Elevene må utvikle en konseptforståelse som gjør de i stand til å uttrykke matematiske idéer gjennom programmeringsspråket.

Datamaterialet fra oppgave seks synliggjør en begrensning i visningsvinduet til dette programmeringsspråket. Visningsvinduet består av piksler i et koordinatsystem. Hvis programmet tegner utenfor rammen til vinduet vil koordinatene forflyttes, og figuren for en annen form enn det kodene angir. I denne studien førte dette til at elevene trakk i tvil

arbeidet de hadde gjort med å generalisere programmet. I denne situasjonen førte denne begrensningen til at elevene ikke utviklet sin matematiske forståelse, heller tvert imot.

Innføringen av programmering byr på didaktiske og pedagogiske utfordringer matematikklærere. Funn fra denne studien gir indikasjoner på at elevenes matematikklæring ikke nødvendigvis kommer som et resultat av verktøyet i seg selv, men hvordan læreren designer undervisningsopplegget. Programmering er ikke en *quick fix* for å øke elevenes matematikkforståelse. Det er måten verktøyet utnyttes i undervisningen som er avgjørende for utviklingen av matematikkforståelsen hos elevene.

6.3 Avsluttende kommentar

Denne studien bidrar med innblikk i hvordan programmering kan brukes som et verktøy i undervisningen, og hvordan elevenes arbeid kan forstås med utgangspunkt i et etablert begrepsapparat og kjerneelementene fra læreplanen. Gjennom arbeid med denne studien har jeg fått en mer teoretisk tyngde for å forstå hva elevene holder på med, og hvordan det kan tilrettelegges, når de jobber med programmering i matematikk.

I studien er det blant annet blitt avdekket hvordan programmering kan bidra til algoritmisk tenkning og generalisering i faget. Det er også avdekket begrensninger ved programmeringsspråket og hvordan dette kan påvirke elevens læringsutbytte.

Programmering vil prege matematikkundervisningen i årene fremover. Det blir interessant å følge fremtidig forskning på dette temaet.

Litteraturliste

Artigue, M. (2002). Learning Mathematics in a CAS Environment: The Genesis of a Reflection about Instrumentation and the Dialectics between Technical and Conceptual Work. *International Journal of Computers for Mathematical Learning, 7*(3), 245-274. https://doi.org/10.1023/A:1022103903080

Assude, T. (2007). Teachers' practices and degree of ICT integration. I D. Pitta-Pantazi, & Philippou, G. (Red.), *Proceedings of the Fifth Congress of the European Society for Research in Mathematics Education* (s. 1339-1348). Larnaca, Cyprus: University of Cyprus.

Blum, W. (2015). Quality teaching of mathematical modelling: What do we know, what can we do? I J. C. Sung (Red.), *The Proceedings of The 12Th international Congress On Mathematical Education : intellectual and Attitudinal Challenges* (s. 73-96). Seoul, Korea: Springer, Cham.

Borg, A., Fahlgren, M. & Ruthven, K. (2020). Programming as a mathematical instrument: the implementation of an analytic framework. I A. Donevska-Todorova, E. Faggiano, J. Trgalova, Z. Lavicza, R. Weinhandl, A. Clark-Wilson & H.-G. Weigand (Red.), *Mathematics Education in the Digital Age (MEDA) PROCEEDINGS* (s. 435-442). Linz, Austria:

Buteau, C., Gueudet, G., Muller, E., Mgombelo, J. & Sacristán, A. I. (2019). University students turning computer programming into an instrument for 'authentic' mathematical work. *International journal of mathematical education in science and technology, 51*(7), 1020-1041. https://doi.org/10.1080/0020739X.2019.1648892

Buteau, C., Muller, E., Mgombelo, J., Sacristán, A. I. & Dreise, K. (2020). Instrumental Genesis Stages of Programming for Mathematical Work. *Digital Experiences in Mathematics Education, 6*(3), 367-390. https://doi.org/10.1007/s40751-020-00060-w

Drijvers, P. & Trouche, L. (2008). From artifacts to instruments: A theoretical framework behind the orchestra metaphor. I G. W. Blume & M. K. Heid (Red.), *Research on technology and the teaching and learning of mathematics* (bd. 2, s. 363-391). Charlotte, NC: Information Age.

Flø, E. E. (2021). Programmering i LK20. *Tangenten: tidsskrift for matematikkundervisning, 32*(1), 3-9.

Gueudet, G., Buteau, C., Muller, E., Mgombelo, J., Sacristán, A. I. & Rodriguez, M. S. (2022). Development and evolution of instrumented schemes: a case study of learning programming for mathematical investigations. *Educational studies in mathematics, 110*, 353–377. https://doi.org/10.1007/s10649-021-10133-1

Gueudet, G. & Trouche, L. (2009). Towards new documentation systems for mathematics teachers? *Educational studies in mathematics, 71*(3), 199-218. https://doi.org/10.1007/s10649-008-9159-8

Misfeldt, M. & Ejsing-Duun, S. (2015, Februar). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. *CERME 9 - Ninth Congress of the European Society for Research in Mathematics Education* (s. 2524-2530). Hentet fra https://hal.archives-ouvertes.fr/hal-01289367

NOU 2013:2. (2013). *Hindre for digital verdiskaping : utredning fra Digitalutvalget*. Hentet fra https://www.regjeringen.no/no/dokumenter/nou-2013-2/id711002/?ch=1

NOU 2015:8. (2015). *Fremtidens skole — Fornyelse av fag og kompetanser*. Hentet fra https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/?ch=1

Postholm, M. B., Jacobsen, D. I. & Søbstad, R. (2018). *Forskningsmetode for masterstudenter i lærerutdanningen*. Oslo: Cappelen Damm akademisk.

Rabardel, P. (2002). People and technology: a cognitive approach to contemporary instruments. I. Hentet fra https://hal.archives-ouvertes.fr/hal-01020705/document

Ruthven, K. (2013). From design-based research to re-sourcing 'in the wild': reflections on studies of the co-evolution of mathematics teaching resources and practices. *ZDM Mathematics Education, 45*(7), 1071-1079. https://doi.org/10.1007/s11858-013-0547-x

Ruthven, K. (2014). Frameworks for Analysing the Expertise That Underpins Successful Integration of Digital Technologies into Everyday Teaching Practice. I A. Clark-Wilson, O. Robutti & N. Sinclair (Red.), *The Mathematics Teacher in the Digital Era*. Dordrecht: Springer.

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*. Utdanningsdirektoratet: Hentet fra: http://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf.

Sevik, K., et al. (2016). *Programmering i skolen. Notat fra Senter for IKE i utdanningen, november 2016*. Hentet fra https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Trouche, L. (2004). Managing the Complexity of Human/Machine Interactions in Computerized Learning Environments: Guiding Students' Command Process through Instrumental Orchestrations. *International Journal of Computers for Mathematical Learning, 9*(3), 281-307. https://doi.org/10.1007/s10758-004-3468-5

Trouche, L. (2005). An Instrumental Approach to Mathematics Learning in Symbolic Calculator Environments. I D. Guin, K. Ruthven & L. Trouche (Red.), *The Didactical Challenge of Symbolic Calculators* (s. 137-162). Boston, MA: Springer.

Utdanningsdirektoratet. (2013). *Læreplan i matematikk fellesfag (MAT1-04)*. Hentet fra http://data.udir.no/kl06/MAT1-04.pdf

Utdanningsdirektoratet. (2019). *Algoritmisk tenkning*. Hentet fra https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2020). *Læreplan i matematikk fellesfag 1.-10. trinn (MAT01-05*. Hentet fra https://data.udir.no/kl06/v201906/laereplaner-lk20/MAT01-05.pdf?lang=nob

Verillon, P. & Rabardel, P. (1995). Cognition and Artifacts: A Contribution to the Study of Though in Relation to Instrumented Activity. *European Journal of Psychology of Education, 10*(1), 77-101. Hentet fra http://www.jstor.org/stable/23420087

Wing, J. M. (2006). Computational thinking. *Association for Computing Machinery, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Yin, R. K. (2009). *Applications of case study research* (4. utg.). Thousand Oaks, California: Sage.

Vedlegg

Vedlegg 1 – Informasjonsskriv og samtykkeerklæring

Vil du delta i forskningsprosjektet

" Matematikk og programmering"?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å se på læringsutbyttet i matematikk når elever jobber med programmering i matematikkfaget. I dette skrivet får du informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Fra og med 2020 er programmering blitt en del av matematikkfaget. Formålet med dette prosjektet er å se nærmere på hvordan matematikken kommer til syne når det jobbes med programmering. Denne forskningen er en del av en masteroppgave jeg skriver som student ved Universitet i Agder.
Hvem er ansvarlig for forskningsprosjektet?
Universitet i Agder er ansvarlig for prosjektet.
Hvorfor får du spørsmål om å delta?
Forskningen skal skje i matematikktimene i klasserommet. I denne perioden vil dere jobbe gruppevis. Hvis du har sagt deg villig til å bli observert vil du kunne komme på en av gruppene som observeres. Gruppene vil ha gruppestørrelser på 3-4 elever. Det er 1-2 grupper som skal observeres. Alle i klassen vil bli spurt.
Kjønnsbalanse og pedagogiske vurderingen vil legges til grunn for inndelingen av gruppene.
Hva innebærer det for deg å delta?
Hvis du velger å delta i prosjektet, innebærer det at du kan bli med på en gruppe som blir observert noen av mattetimene. Her kommer jeg til å se på hvordan dere løser noen oppgaver som dere får på PC. Denne observeringen skjer ved det tas skjerm- og lydopptak på PCen.
Skjerm- og lydopptak vil bli slettet ved prosjektslutt.
I ettertid kan det bli aktuelt med et intervju/samtale der vi diskuterer refleksjoner rundt hvordan oppgavene ble løst. I intervjuene kan det også bli diskutert hvordan dere oppfatter programmering som en del av matematikkfaget.
Tilbakemeldingene dine vil bli anonymisert i oppgaven.
Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Alle elevene i klassen kommer til å jobbe med de samme oppgavene i timene. Om du ønsker å trekke det vil du fortsette arbeidet på en gruppe som da ikke blir observert.

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.

Det er kun jeg, og min veileder ved UiA, som vil ha tilgang til observasjonene.
I masteroppgaven vil jeg ikke bruke ditt navn, men et kodenavn som ikke er sporbart til deg.
Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Opplysningene anonymiseres når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen er mai 2022. Skjerm- og lydopptak slettes ved prosjektslutt.
Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til 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 UiA 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:

- UiA ved Anders Skarpeteig Fidje (38 14 10 17 eller 482 26 407)
- Vårt personvernombud: https://www.uia.no/om-uia/si-ifra/informasjonssikkerhet-og-personvern/personvern-paa-uia

Hvis du har spørsmål knyttet til NSD sin vurdering av prosjektet, kan du ta kontakt med:

- NSD – Norsk senter for forskningsdata AS på epost (personverntjenester@nsd.no) eller på telefon: 55 58 21 17.

Med vennlig hilsen

Christian Ore
Student

Anders Skarpeteig Fidje
prosjektansvarlig/veileder

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet matematikk og programmering, og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i studien ved at det gjøres skjerm- og lydopptak av min gruppe når vi jobber i timene.
- ☐ å delta i studien ved at det gjørs et intervju i etterkant av gruppearbeidet.

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

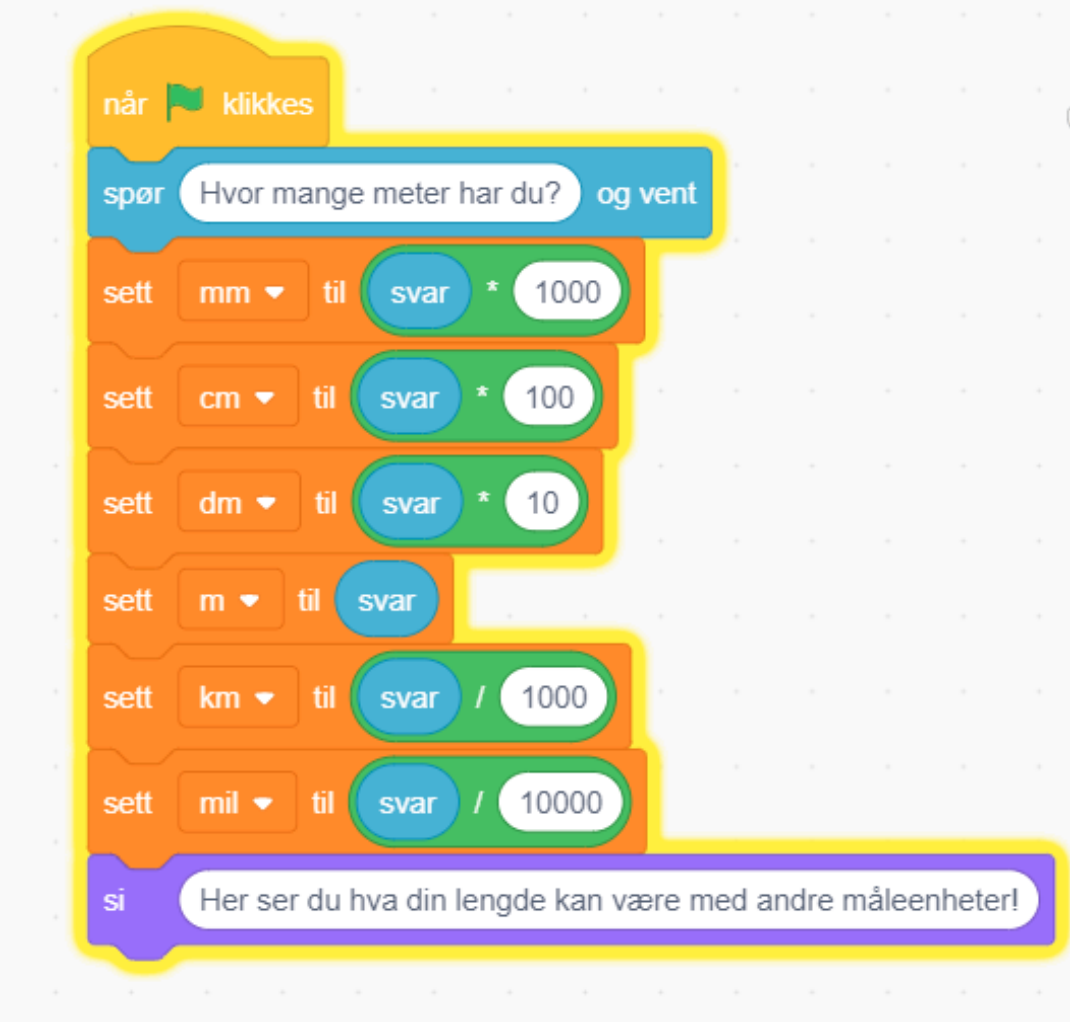
---
(Elevens navn, med blokkbokstaver)

(Signert av elev, dato)

(Signert av foresatt, dato)

Oppgave 1 - Diskutere

Bildet under viser et program som gjør om meter til andre målenheter.



1. Diskuter hva dere tror de forskjellige blokkene gjør.
2. Hva er  i dette programmet?
3. Hva skjer med svarene i de grønne blokkene?
4. Hva er variabler i dette programmet?
5. Hvordan kan programmet alltid vite hva det skal svare selv om du skriver inn forskjellige svar hver gang?

Oppgave 1 - Gjøre

1. Kode programmet selv eller finn det ferdig oppsatt på linken under.

   Link: https://scratch.mit.edu/projects/482563510
2. Endre programmet slik det gjør om fra **cm** til de andre måleenhetene.

Tips til diskusjon i gruppa:

1. Hvordan må den blå *spør*-blokka endre for at den skal passe?
2. Hvordan må de grønne regne-blokkene endres?
3. Hvordan må den lilla *si*-blokka endres?

Ekstraoppgave hvis dere er tidlig ferdig:

Legg inn to nye blokker i programmet som skal gjøre cm (svaret) om til dam og hm. Disse to nye blokkene passer naturlig inn mellom blokkene med m og blokka med km.

Hint:

1 dam (dekameter) = 10 m

1 hm (hektometer) = 100 m

10^3	kilo	k	Tusen	1 000
10^2	hekto	h	Hundre	100
10^1	deka	da	Ti	10
10^0			En	1
10^{-1}	desi	d	Tidel	0,1
10^{-2}	centi	c	Hundredel	0,01
10^{-3}	milli	m	Tusendel	0,001

Oppgave 2 - Diskutere

Bildet under viser et program som gjør om timer til minutter.

1. Diskuter hva dere tror de forskjellige blokkene gjør.
2. Hva er svar i dette programmet?
3. Hva er variabler i dette programmet?
4. Hva skjer i de grønne blokkene?
5. Hvor mange grønne blokker er det egentlig inni den lilla blokka?

Oppgave 2 - Gjøre

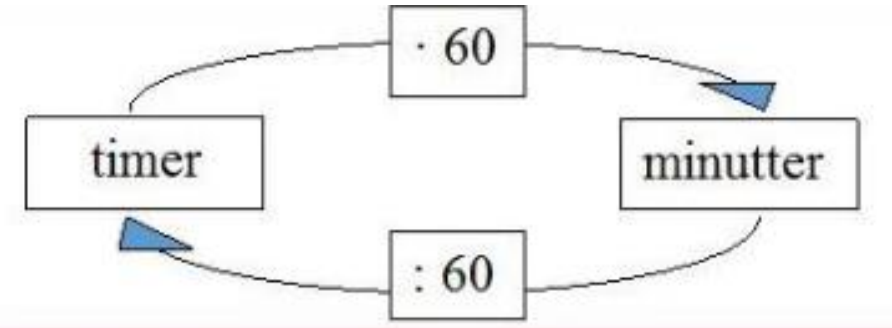
1. Kode programmet selv eller finn det ferdig oppsatt på linken under.

   Link: https://scratch.mit.edu/projects/575827362

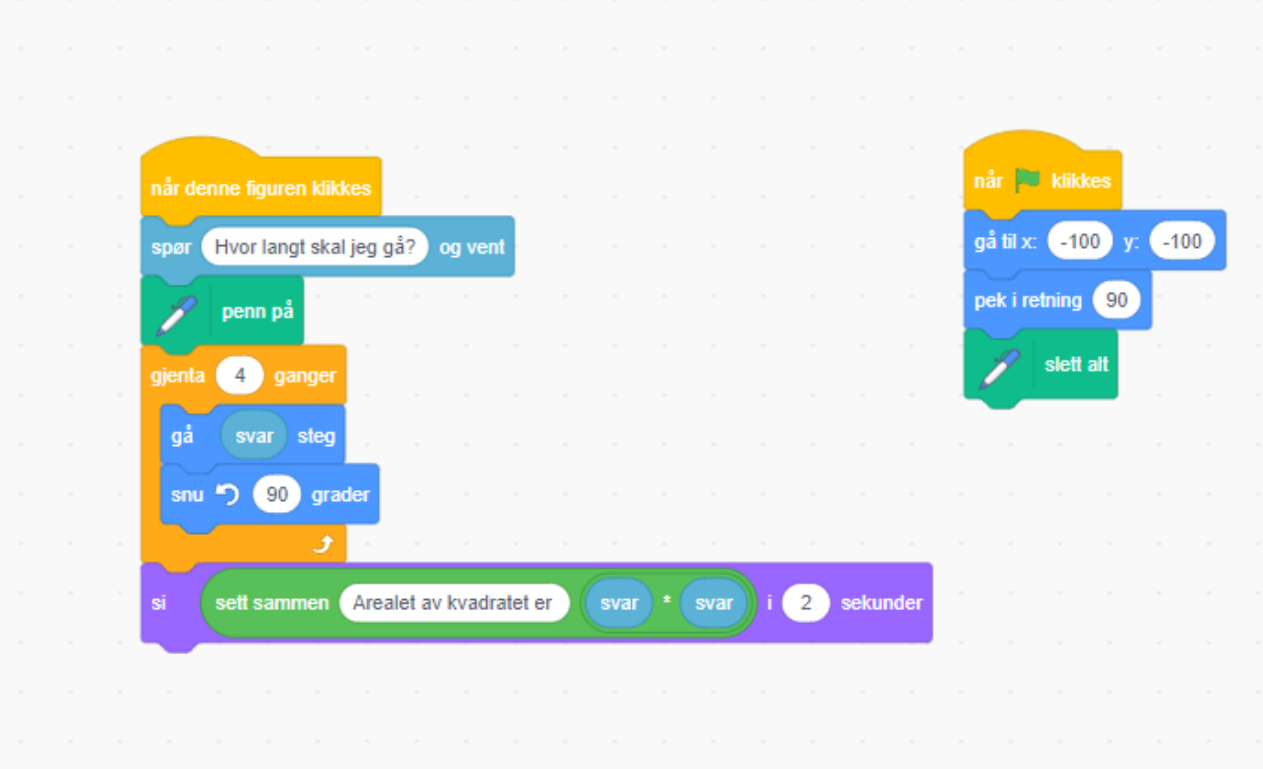
2. Endre programmet slik det gjør om fra **minutter til timer**.

Tips til diskusjon i gruppa:

1. Hvordan må den blå *spør*-blokka endre for at den skal passe?
2. Hvordan må den grønne regne-blokkene endres?
3. Hvordan må variabelen (orange blokk) endres?
4. Hvordan må den lilla *si*-blokka endres?



Oppgave 3 - Diskutere



1. Ta for dere hver av blokkene i programmet. Hva dere tror de forskjellige blokkene gjør.
2. Forklar til hverandre hvordan dere tror dette programmet vil fungerer når dere kun kan se kodene over.
3. Den lille blokkjeden til høyre henger ikke direkte sammen med hovedprogrammet.
    a. Hva gjør denne blokkkjeden?
    b. Hvorfor bør denne være med i programmet?

Oppgave 3 - Gjøre

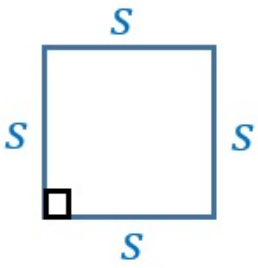
1. Kode programmet selv på https://scratch.mit.edu/ → programmering
2. Hva sier programmet at arealet blir du får om du svarer at katten skal gå 10?

   PS: Dere må flytte på katten for å se firkanten.
3. Hva ser programmet at arealet blir du får om du svarer at katten skal gå 20?
4. Dere har nå doblet lengden på sidene i kvadratet (fra 10 til 20). Hva gjør dette med arealene? Blir de også dobbelt så store? Hvorfor/hvorfor ikke?
5. Endre programmet slik det finner **omkretsen** i stedet for arealet av kvadratet.

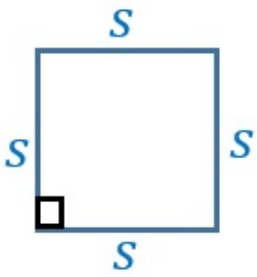
Tips til å løse oppgave 5:

1. Hva er omkrets?
2. Hva trenger vi å vite for å regne ut omkrets av et kvadrat?
3. Hvordan må den grønne regne-blokkene endres?
4. Hvordan må den lilla *si*-blokka endres?

$$A = s \cdot s = s^2$$

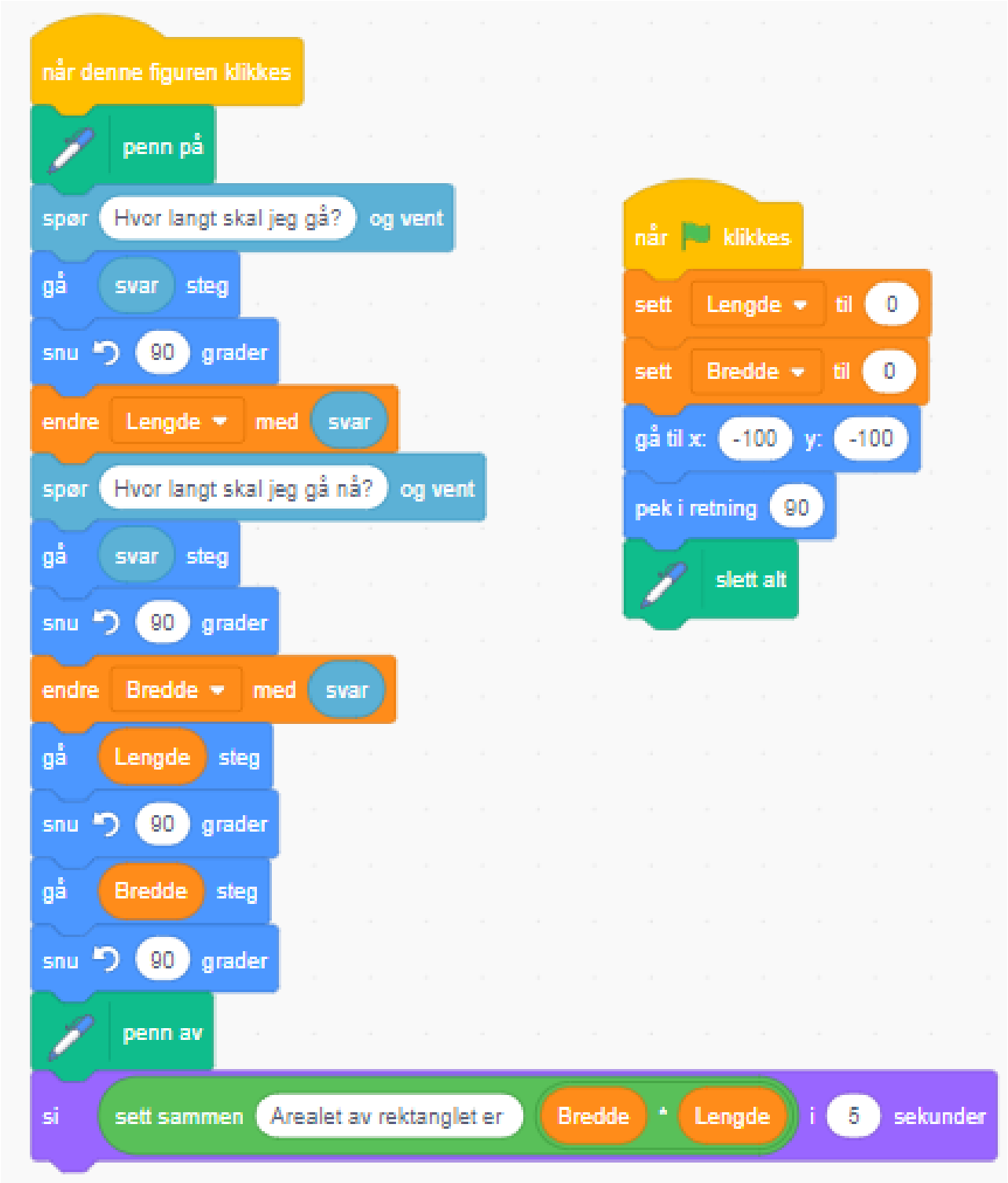


$$O = s + s + s + s = 4s$$



Oppgave 4 - Diskusjon

Hva tror dere dette programmet gjør?



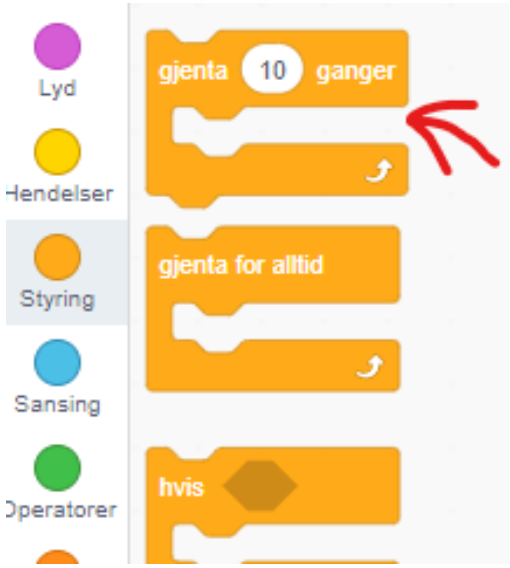
Oppgave 4 - Gjøre

1. Kode programmet selv eller finn det ferdig oppsatt på linken under.

   Link: https://scratch.mit.edu/projects/582261924

2. Endre programmet slik det finner **omkretsen** i stedet for arealet av rektanglet.
3. Prøv å endre programmet slik at dere bruker færre blokker.

   Det er mulig å effektivisere dette programmet (altså bruke færre blokker) ved å bruke en løkke-blokk.



PS: Det er 16 blokker i hovedkjeden nå. Ved bruk av løkke kan det bli redusert til 13 blokker. Man redder ikke verden av å gjøre programmet noen blokker kortere, men koding bør alltid være så effektiv som mulig.

Løsningsforslag finner dere på denne linken:

https://scratch.mit.edu/projects/582496808

Oppgave 5 - Diskutere

Bildene under viser et program som simulerer at en kaster to terninger og finner tverrsummen til dem. Minste tverrsum det er mulig å få er 2 (1+1) og høyeste tverrsum som er mulig er 12 (6+6).

1. Diskuter hva dere tror de forskjellige blokkene gjør. Ja, jeg vet at dette er et mer komplekst program. Vi prøver likevel ☺
2. Hva er svar i dette programmet?
3. Hvor mange variabler er der i dette programmet?
4. Hva skjer i de grønne blokkene?
5. Hvorfor er det flere kjeder (4 stk) med blokker som ikke henger sammen med hverandre?

Oppgave 5 – Gjøre

1. Åpne programmet ved å klikke på lenken under.

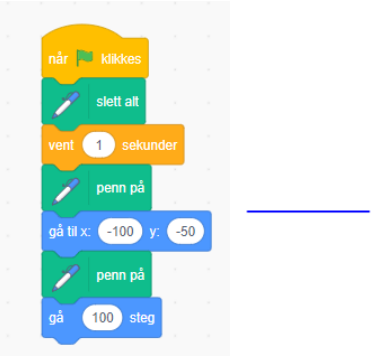
   Link: https://scratch.mit.edu/projects/579603360

2. Endre programmet slik at gjør 20 kast med terningene.
3. Velg en tverrsum mellom 2 og 12.
4. Hvor stor sannsynlighet mener programmet det er for å få den tverrsummen dere valgte?
5. Nullstill å prøv programmet 5 ganger med samme tverrsum. Får dere samme svar hver gang? Hvorfor tror dere det er slik?
6. Endre slik at programmet foretar 1000 kast med terningene.
7. Velg den samme tverrsummen som i oppgave 3.
8. Hvor stor sannsynlighet mener programmet nå det er for å få den tverrsummen dere valgte?
9. Nullstill å prøv programmet 5 ganger med samme tverrsum og antall kast. Får dere likere resultat denne gangen? Hvorfor tror dere det er slik?
10. Hvilket av svarene tror dere gir det beste resultatet? Begrunn svaret.
11. Google begrepet ***store talls lov***. Hvordan passer dette inn i denne oppgaven?
12. Prøv dere frem med forskjellige tverrsummer fra 2-12, og 1000 kast. Hvilken tverrsum er det størst sannsynlighet å få når en kaster to terninger?

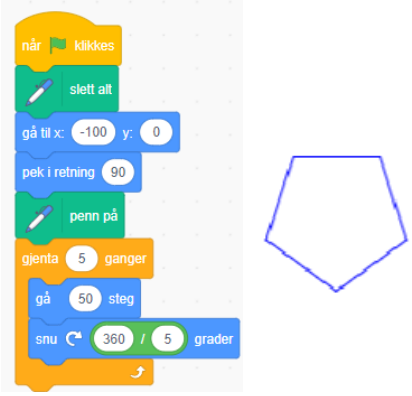
Oppgave 6

Eksempler på hvordan man lager figurer i Scratch:

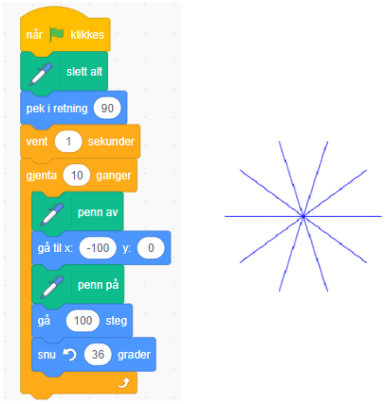
Eks1
Programmet tegner en strek.



Eks 2
Programmet tegner femkant.



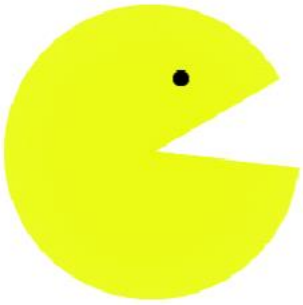
Eks 3
Programmet tegner en stjerne.



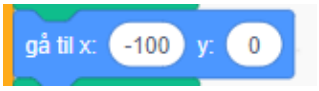
1. Løs oppgavene:
   a. Tegn en trekant.
   b. Tegn en firkant.
   c. Tegn en sekskant.
   d. Hvilke andre former klarer du å tegne?

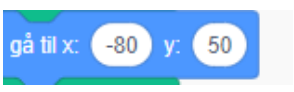
2. Lage et program som ber deg oppgi antall hjørner, og deretter tegner en mangekant med det gitte antall hjørner?
   For eksempel: Programmet spør deg: «Hvor mange hjørner?» - Du svarer: «10» - *Programmet tegner en 10-kant.*

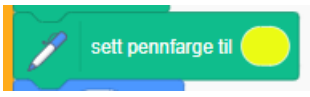
3. Bildet av Pac-man under er laget med utgangspunkt i kodene i eksempel 3 (Stjerne)
   Forsøk å lage et program som tegner en figur som er mest mulig lik Pac-man under.



Bildene under kan bruker som tips/inspirasjon i arbeidet:







Vedlegg 3 – Transkripsjon av oppgave 4 - gruppe 1

ID	Elev	Utsagn
O4G1-1	Elev 1	Da kan vi ta lengde gange 2 pluss bredde gange 2.
O4G1-2	Elev 2	Det blir jo riktig nå, blir det ikke det?
O4G1-3		*En liten periode uten snakk der elev gjør endringer i programmet.*
O4G1-4	Elev 1	Sånn!
O4G1-5	Elev 2	Men hvis du trykker på den så blir det litt mer klart.
O4G1-6	Elev 1	Ja, men må vi ikke endre noe her også [*henviser til den øvrig del av programmet med musepekeren*]
O4G1-7	Elev 2	Så ser vi det der bedre. [*Det er kommer ikke tydelig frem fra skjermopptaket hva denne eleven henviser til.*]
O4G1-8	Elev 1	Men disse blir jo ikke i sammen nå.
O4G1-9	Elev 1	Bredde gange 2…
O4G1-10	Elev 2	Pluss lengde gange 2. Det blir jo rett.
O4G1-11	Elev 1	Ja, han må jo bare ta ganginga først, før han tar plussen, hvis han er litt smart da.
O4G1-12	Elev 2	Jeg tror han skal ta plussen først. [*uten at han argumenterer eller utdyper ytterligere*]
O4G1-13	Elev 1	Må vi ikke ta dette også? [*henviser igjen til den øvrig del av programmet med musepekeren.*] Eller skal vi bare ta fire plusser?
O4G1-14	Elev 2	Vi har jo gjort det riktig nå.
O4G1-15	Elev 1	Ja. Test. Test?
O4G1-16	Elev 2	Ja.
O4G1-17	Elev 2	Bare trykk på…
O4G1-18	Elev 1	Ok, nå har jeg det rett her. 5 og 8. [*dette er verdiene som oppgis om inndata i programmet.*] Der! Omkretsen av rektanglet er 96.
O4G1-19	Elev 2	Der fant vi det da.
O4G1-20	Elev 1	Men blir det rett da?
O4G1-21	Elev 3	Ja
O4G1-22	Elev 1	Blir det rett? 96 liksom. Blir ikke det litt mye?
O4G1-23	Elev 1	Da må vi heller ta pluss her nede.
O4G1-24	Elev 1	Eller vi prøver en gang til der vi tar noen små tall. Hvis det er 1 så, hvis alle sidene er 1 og 2 så blir det jo, eh 6.
O4G1-25	Elev 2	Hvordan da? Det skal jo bli 4.
O4G1-26	Elev 1	Nei, fordi bredden er 2 og da blir det 4. Og så lenge er 1 og da blir det 2.
O4G1-27	Elev 1	2 + 4 er 6.
O4G1-28	Elev 1	Hvordan blir det 8?
O4G1-29	Elev 1	Han *[katten]* tenker at bredden er…
O4G1-30	Elev 2	Bredden er 2 gange 2.
O4G1-31	Elev 2	Det kan være han tar disse først.
O4G1-32	Elev 1	Ja, det er det jeg også tenker fordi denne her er jo 1 [*henvis til innerste operatørblokk*].

O4G1-33	Elev 1	Og dette er én. [*Henviser til nest innerste operatørblokk*]
O4G1-34	Elev 2	Vent, se her.
O4G1-35	Elev 1	Og dette er én. [*Henviser til ytterste av de tre operatørblokkene som utfører selve beregningene*]
O4G1-36	Elev 1	Hva skjer bak her?
O4G1-37	Elev 2	Nå har vi jo to der.
O4G1-38		*Elevene har noe utfordring med at deler av programmet kommer dobbelt. Dette skjer trolig ved at de har holdt inne høyre musetast under arbeidet og dermed laget en kopi. De bruker noe tid på å rydde opp igjen.*
O4G1-39	Elev 1	Så tar vi bort disse igjen. Vi tar bort dette så tar vi heller pluss. Vi må beholde lengde og bredde tror jeg.
O4G1-40	Elev 1	Ut. Ut. Så må vi ha mer pluss. Sånn, nå blir det rett.
O4G1-41	Elev 2	Hæ, hvordan har du fått det til? Åja.
O4G1-42	Elev 1	Sånn, og så må vi har flere variabler. En til bredde der og en lengde her.
O4G1-43		*Elevene diskuterer hva de skal i et annet fag mens eleven som styrer musa bruker noe tid på å sette sammen operatørblokkene og variablene.*
O4G1-44	Elev 1	Okei nå begynne vi. Vi tar noe lett. Vi tar 1 og 2 igjen.
O4G1-45	Elev 1	Nå ble det enda høyere enn i stad.
O4G1-46	Elev 2	Du tok jo først…
O4G1-47	Elev 1	Jeg tok jo 1 og 2.
O4G1-48	Elev 2	Lengde gange bredde er 6. Pluss lengde gange bredde.
O4G1-49		*Lærer kommer inn i rommet.*
O4G1-50	Elev 1	Det blir litt feil her.
O4G1-51	Lærer	Blir det feil?
O4G1-52	Elev 1	Ja, jeg tok 1 og 2 som bredde og lengde, og så blir det plutselig 4 og 2.
O4G1-53	Elev 1	Så blir det dobbelt så mye.
O4G1-54	Lærer	Ok, still spørsmål, knytter svaret opp mot lengde, stiller nytt spørsmål og knyttet seg opp mot bredde. [*"leser" seg gjennom kodene høyt.*]
O4G1-55	Lærer	Dette skal jo bli bra.
O4G1-56	Elev 1	Ja, men det blir ikke bra.
O4G1-57	Lærer	Ok, hvis vi nå spør katten. Han går 100. [*angi første inndata*] Da går han 100 slik, og snur 90 grader. Da er han kommer dit i programmet. Så stiller han spørsmål og vi sier 200.
O4G1-58	Lærer	Når vi nå trykker enter vil han få et svar. Går det svaret langt. Snur 90 grader. Lagrer svaret. Da vil ha stå der oppe.
O4G1-59	Lærer	Så skal ha gå bort og ned. Det er den og den [*indikerer på skjermen og trykker på enter*]
O4G1-60	Lærer	Det blir jo rett. Svaret blir 600.
O4G1-61	Elev 1	Hvordan blir det det?
O4G1-62	Lærer	100 + 200 + 100 + 200
O4G1-63	Elev 1	Det var ikke det som stod i stad. Da tok jeg 1 og 2. Så ble det 2 og 4 plutselig.

O4G1-64	Lærer	Det er ikke det at du ikke nullstilte programmet da?
O4G1-65	Elev 1	Det kan være.
O4G1-66	Lærer	1 og 2. [*bruker 1 og 2 som inndata*]
O4G1-67	Lærer	Omkretsen er 6.
O4G1-68	Lærer	Hvis du gjør dette en gang til nå. 1 og 2.
O4G1-69		*Kjører programmet en gang med 1 og 2 som inndata, uten å nullstille det. Programmet oppgir da at omkretsen er 12.*
O4G1-70	Elev 1	Ja, det var nok det som skjedde i stad.
O4G1-71	Lærer	Men da var det jo ikke noe galt. Da er jo alt perfekt.
O4G1-72	Elev 2	Ja
O4G1-73	Lærer	Så bra.
O4G1-74	Elev 1	Da er vi ferdige.

Vedlegg 4 – Transkripsjon av oppgave 4 - gruppe 2

ID	Elev	Utsagn
O4G2-1	Elev 4	Kode programmet selv eller… *[eleven leser oppgaven*]
O4G2-2	Elev 5	Omkrets av rektangel er. Hvor er musa? Der.
O4G2-3	Elev 4	Omkretsen av rektanglet er.
O4G2-4		*Elevene endrer testen i si-blokka.*
O4G2-5	Elev 5	Pluss og pluss.
O4G2-6	Elev 4	Hæ? [*Legger enn en addisjonsblokk til programmet*]
O4G2-7	Elev 4	Sånn? Nå er det det *[bredde]* pluss lengde gange to.
O4G2-8		*Tester programmet med 50 og 60 som lengde og bredde i rektanglet. Programmet gir svaret 164.*
O4G2-9	Elev 5	Prøv på nytt. [*Eleven gir ikke uttrykk for at det tror svaret er feil*] Skriv 100.
O4G2-10	Elev 4	100 [*eleven gir lengde verdien 100*]
O4G2-11	Elev 4	200 [*eleven gir bredde verdien 200*]
O4G2-12		Programmet beregner omkretsen med de verdiene til å bli 400
O4G2-13	Elev 5	Prøv 100 to ganger
O4G2-14		Programmet beregner omkretsen med de verdiene til å bli 300
O4G2-15	Elev 4	Det blir jo ikke 300. Det var ikke 300 da.
O4G2-16	Elev 5	100 + 100 + 100 + 100. Det må jo bli 400.
O4G2-17	Elev 4	Det blir jo 400. Vi prøver igjen.
O4G2-18	Elev 5	Okei, prøv med 50 nå. Da skal det bli 100 og.. nei 200.
O4G2-19		*Programmet gjør en ny beregning med 50 som verdi for lengde og bredde.*
O4G2-20	Elev 5	150. What?
O4G2-21	Elev 4	Han tar jo bare tre av sidene.
O4G2-22	Elev 4	Hvorfor gjør han det?
O4G2-23	Elev 4	Den tar jo bare tre av sidene.
O4G2-24		*Lærer kommer inn i rommet*
O4G2-25	Elev 4	Hallo, du. Han regner bare tre av sidene. Se nå. 50. 50.
O4G2-26		*Elev skriver inn 50 som lengde og 50 som bredde.*
O4G2-27	Elev 4	Det blir 150.
O4G2-28	Elev 5	Hvorfor blir det et kvadrat da?
O4G2-29		*Elev konstaterer at det tegnes et komplett kvadrat i programmet til tross for at beregningen av omkretsen kun tilsvarer tre sider.*
O4G2-30	Lærer	Ja, hvorfor gjør han det?
O4G2-31		*Lærer lar elevene få litt tid til å tenke.*
O4G2-32	Lærer	Dere sier han skal ta bredde pluss lengde ganger 2.
O4G2-33	Elev 5	Den er 2.
O4G2-34	Lærer	Hvorfor det?
O4G2-35	Lærer	Hvorfor blir det ikke rett nå?
O4G2-36	Elev 4	Men det er bare 3 sider.
O4G2-37	Elev 4	For at det er bredde én gang og lengde 2 ganger.

O4G2-38	Elev 5	Ah, vi ganger jo alltid før pluss, og da blir det lengde ganger 2 pluss bredde.
O4G2-39	Elev 5	Ja da blir det bare tre!

Vedlegg 5 – Transkripsjon av oppgave 6 - gruppe 1

ID	Elev	Utsagn
O6G1-1	Elev 1	Vi prøver med 50 steg. Sånn! Oi, det ble litt mye det.
O6G1-2	Elev 2	Ja, litt
O6G1-3	Elev 1	Vi prøver med 25 steg i stedet.
O6G1-4	Elev 2	Ok.
O6G1-5	Elev 1	Slett alt.
O6G1-6		*Legger inn en blokk for slett alt i slutten av programmet, men tar den bort når de ser at det sletter streken.*
O6G1-7	Elev 1	Så legger vi til snu. *[blokka].* Da mener jeg den kommer til å gå oppover.
O6G1-8	Elev 2	45.
O6G1-9	Elev 1	Og gjenta 3 [*løkka settes til å gjentas 3 ganger*].
O6G1-10	Elev 1	Og slett alt. [*setter inn blokka slett alt*]
O6G1-11	Elev 1	Se der *[latter]*
O6G1-12	Elev 1	Nei, fader, vi må ikke ha den på slutten der. Hvorfor ble den sånn nå?
O6G1-13	Elev 2	*[Ler]*
O6G1-14	Elev 1	Vent, så nå. [s*etter inn en operatørblokk*]. Så blir det jo 360 delt på 3. Blir det ikke det?
O6G1-15	Elev 2	Åh!
O6G1-16	Elev 1	Det ble rett.
O6G1-17	Elev 2	Da er det oppgave B.
O6G1-18	Elev 2	Nå skal vi lage en firkant. Det er jo mye lettere, er det ikke det?
O6G1-19	Elev 2	Da må vi fikse på den snu-blokka.
O6G1-20		*Endrer divisor i operatørblokk fra 3 til 4.*
O6G1-21		*Tester programmet med resultat at det bare tegner tre av sidene.*
O6G1-22	Elev 1	Nei, vi må jo endre løkka også til 4.
O6G1-23	Elev 2	Da har vi gjort B. Det var jo lett.
O6G1-24	Elev 2	Tegn en sekskant.
O6G1-25	Elev 1	Da kan vi jo, eh, den sånn og den sånn. [*endrer divisor og løkke til 6*]
O6G1-26		*Tester programmet og det tegnet en sekskant.*
O6G1-27	Elev 2	Hvilke andre former klarer du å tegne? [*leser oppgave d)*]
O6G1-28		*Elevene endrer divisor og løkke til 7, og tester programmet.*
O6G1-29		*Elevene endrer divisor og løkke til 8, og tester programmet.*
O6G1-30	Elev 2	1,2,3,4,5,6,7. Det er jo bare sju kanter. Hæ?
O6G1-31		*De aksepterer dette uten å utforske noe mer, og går videre på neste oppgave.*

O6G1-32		*Det er usikkert hvorfor programmet har tegnet et sjukant. Slik programmet står så skal det tegne en åttekant. Det er trolig en lagg i Scratch der det tegnet med utgangspunkt i forrige input.*
O6G1-33	Elev 2	Hvordan gjør vi dette?
O6G1-34	Elev 1	Vent. Spør hvor mange. [*setter inn spørre-blokk og endrer spørsmålet slik at det er tilpasses oppgaven*]
O6G1-35	Elev 2	Sånn.
O6G1-36	Elev 1	Og vent. Jeg er ikke helt sikker på hva "og vent" betyr.
O6G1-37	Elev 1	Ah, jeg tror vi har det nå. Vi setter inn svar [*sett inn svar-blokk i løkka og som divisor*]
O6G1-38	Elev 2	Vi prøver med 10.
O6G1-39	Elev 1	1,2,3,4,5,6,7,8,9,10. [*Teller at det blir en tikant*]
O6G1-40	Elev 1	Nå må vi kanskje prøve med flere da.
O6G1-41	Elev 1	15 [*setter svar til 15*]
O6G1-42	Elev 2	Oi, det ser nesten ut som en runding.
O6G1-43	Elev 1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15. Det blir rett det.
O6G1-44	Elev 2	Neste oppgave. Skal vi lage den som er der da?
O6G1-45	Elev 2	Da må vi ha den. Og den. Og den. Og den blå. [*eleven bygger opp programmet for stjerne slik det står i eksemplet i oppgaven*]
O6G1-46	Elev 1	Hvis vi prøver det nå.
O6G1-47		*De tester programmet*
O6G1-48	Elev 1	Vi må ha litt flere steg., [*endrer til 50*]
O6G1-49	Elev 2	Men hvordan kan vi få flere streker?
O6G1-50	Elev 1	Hva om vi gjentar 50 ganger?
O6G1-51		*Tester programmet som kun legger strekene over hverandre.*
O6G1-52	Elev 1	Og hvis vi endrer den [*rotasjons-blokka*] til 10 grader.
O6G1-53		*Tester programmet*
O6G1-54	Elev 1	Åh! Jeg tror dette kan gå faktisk.
O6G1-55	Elev 1	Hvis vi tar 0.1 [*i rotasjonsblokka*]
O6G1-56		*Tester programmet*
O6G1-57	Elev 1	Da må vi endre her. [a*ntall repetisjoner i løkka. Setter verdien til 100000*] Sånn!
O6G1-58	Elev 2	Oi, men hvordan skal vi få bort den delen der? [*Henviser trolig til Pac-man sin munn*]
O6G1-59	Elev 1	Det vet jeg ikke enda.
O6G1-60	Elev 2	Så må vi ha sett farge på den.
O6G1-61	Elev 1	Dette tar sykt lang tid. Kanskje det var litt mye med 0.1 grad, hehe
O6G1-62	Elev 2	Ja, litt. Skal vi sette inn den da? [*henviser til fargevelger for penn, og setter den inn i programmet*]
O6G1-63	Elev 1	Vi må ha endre grader. Sånn, nå kommer den til å gå litt fortere i hvert fall.
O6G1-64		*Tester programmet*
O6G1-65	Elev 2	Hvis vi stopper programmet før det er ferdig, sa får vi jo den figuren.
O6G1-66	Elev 1	Men det er jo nokså lett egentlig. Vi kan jo bare ta penn av etter

		hvert.
O6G1-67		*De tester programmet. Denne gangen drar de i katten når programmet kjører, og det dannes små sektorer som ikke får farge.*
O6G1-68	Elev 2	Nei, se! *[latter]*
O6G1-69	Elev 1	Jeg tror vi kan klare det nå. [*endrer løkka til 100*]
O6G1-70		*Tester programmet*
O6G1-71	Elev 1	Er ikke det 90 grader?
O6G1-72	Elev 2	Jo
O6G1-73	Elev 1	Æh, da vi må har 200 på 180 grader, og 270 på 300. 350! Vi prøver det.
O6G1-74		*Setter inn 350 i løkka og tester programmet.*
O6G1-75	Elev 1	Det er jo brukende.
O6G1-76	Elev 2	Ja, men munnen skal jo være der da.
O6G1-77	Elev 1	Ja, men det får vi ikke til tror jeg. Eller, det kan jo kanskje være at… Nå går jo den figuren… Nå starter figuren må å gå der. Hvis vi får den til å vente til han er der.
O6G1-78	Elev 1	Hva om vi får den til å gå i andre retningen. Der.
O6G1-79		*Endrer blokk som indikerer retningen fra 90 til -90 og tester programmet.*
O6G1-80	Elev 2	Nå går den jo helt feil
O6G1-81	Elev 1	Ah, vent litt. Se her.
O6G1-82	Elev 2	Hvordan kan vi få den til å.
O6G1-83	Elev 1	180 grader kanskje?
O6G1-84		*Tester programmet*
O6G1-85	Elev 2	Nei, vi må ha..
O6G1-86	Elev 1	Der [*foreslår 0 grader*]
O6G1-87	Elev 2	Nei, vi kan jo ikke bare ha den rett opp.
O6G1-88		*Tester programmet*
O6G1-89	Elev 1	Nei, nå har jeg det. Litt så [*setter verdien til 45 grader*]
O6G1-90		*Tester programmet*
O6G1-91	Elev 1	Ja, se der.
O6G1-92	Elev 2	Da er det øyet
O6G1-93	Elev 1	Da må vi sette inn sett farge til svart. [*setter inn blokkene for pennfarge og penn på i slutten av blokkjeden*]
O6G1-94	Elev 1	Please bli en Pac-man.
O6G1-95		*Tester programmet*
O6G1-96	Elev 2	*[latter]*
O6G1-97	Elev 1	Se nå. Han gå oss jo et tips. [*setter inn en ny koordinat-blokk etter løkka*]
O6G1-98	Elev 1	Den skal være -80 og den skal være 50.
O6G1-99		*Tester programmet*
O6G1-100	Elev 1	Vi har glemt penn av der.
O6G1-101	Elev 2	Men hvorfor har du penn på da?
O6G1-102		*Test programmet*

O6G1-103	Elev 1	Okei, hvis vi prøver å sette den lendre ned. [*ser at øyet vil komme for høyt i forhold til figuren*]
O6G1-104	Elev 1	Hvis vi setter den på, eh. Hvis vi setter den på 25 da.
O6G1-105	Elev 1	Endre pennens bredde? Kanskje den [*setter blokk for endret pennstørrelse inn i programmet*]
O6G1-106		*Her er en periode med tekniske problemer med lyden. Skjermopptak viser at det eksperimenterer med pennens bredde. Så stor bredde på pennen kan være en løsning for å tegne øyet, men ikke være utfordrende med tanke på munnen. De kommer seg etter hvert tilbake til pennbredde 1 og arbeidet med å tilpasse koordinatene.*
O6G1-107	Elev 2	Eh, -80. Og 50. [*henviser til koordinatene som ligger med som tips i oppgaven*]
O6G1-108	Elev 2	Vi må ha på pennen også. Og fargen.
O6G1-109		Tester programmet. Prikken som indikerer hvor øye blir kommer over figuren.
O6G1-110	Elev 1	100 da. [*endrer steg til 100 for å gjøre figuren større*] Da kan det jo være at prikken kommer på riktig plass.
O6G1-111		*Tester programmet*
O6G1-112	Elev 1	Sånn ca der vi vil ha den eller?
O6G1-113	Elev 2	Men hvordan får vi prikken større?
O6G1-114	Elev 1	Da må vi jo ta den og gjøre akkurat det samme. [*henviser til løkka*]
O6G1-115	Elev 2	Ja, da må vi jo gjøre det.
O6G1-116		*Setter inn en løkke til uten å beskrive dette med ord.*
O6G1-117	Elev 1	Da setter vi 55 på den [*y-verdien av koordinatblokka*] så får vi den litt høyere.
O6G1-118	Elev 1	Vi gjør sånn og sånn [*sett inn flere gå-blokk og snu-blokk*]
O6G1-119	Elev 1	Kan vi ikke bare ta 0.9 der også?
O6G1-120	Elev 2	Jo.
O6G1-121		*Tester programmet*
O6G1-122	Elev 1	Vi må gjenta flere ganger. 400 så går den jo helt rundt.
O6G1-123		*Tester programmet*
O6G1-124	Elev 2	Han lager jo øyet, men hvorfor er den streken der?
O6G1-125	Elev 2	Han ble kanskje litt stor [*endrer antall steg til 7.5 i løkka som tegner øyet*]
O6G1-126		*Tester programmet*
O6G1-127	Elev 2	Nå ble øyet fint, men det er en strek der. Hvordan får vi vekk streken?
O6G1-128	Elev 2	Hvorfor skal den [*penn av*] der, og ikke inni der [*som del av løkka*]?
O6G1-129	Elev 1	Vi prøver det.
O6G1-130		*Tester programmet med samme resultat.*
O6G1-131	Elev 1	Kanskje sånn her. [*endrer rekkefølgen på blokken som indikerer penn på og blokk for koordinater*] Nå må det gå!
O6G1-132		*Tester programmet*
O6G1-133	Elev 2	Det var lett! Kødda!

O6G1-134	Elev 2	Øyet ble litt stort da. Vi kan jo gjøre det litt mindre.
O6G1-135		*Endrer antall steg til 5 og tester programmet.*
O6G1-136	Elev 2	Sånn ja. Perfekt!

Vedlegg 6 – Transkripsjon av oppgave 6 - gruppe 2

ID	Elev	Utsagn
O6G2-1	Elev 4	Vi skal tegne en trekant. Trekant. 1,2 3,4,5.
O6G2-2	Elev 4	Denne. Hvorfor er den så liten?
O6G2-3		*Henviser til at blokken hun dro inn i arbeidsområdet er liten i størrelse. Finner zoom-knappen og øker størrelsen.*
O6G2-4	Elev 4	Sånn. Og så en penn.
O6G2-5	Elev 5	Der nede.
O6G2-6	Elev 4	Åja, her. Penn på. Penn slett alt. Er det rett, at den skal slette alt?
O6G2-7	Elev 5	Sikkert
O6G2-8	Elev 4	Slett alt
O6G2-9		*Setter blokka for slett alt inn i programmet.*
O6G2-10	Elev 4	Blå ting. Hvor er den?
O6G2-11	Elev 4	Kanskje det er utseende? Ja, det er det.
O6G2-12	Elev 5	Gå til x y.
O6G2-13	Elev 4	Skal vi skrive det samme som står der?
O6G2-14		*Skriver inn -100 som x- verdi og 0 som y-verdi*
O6G2-15	Elev 5	Og så den. [*Henviser til blokka "pek i retning"*]
O6G2-16	Elev 4	Den trenger vi ikke endre.
O6G2-17	Elev 4	Gjenta… Da skal den gjenta seg 3 ganger hvis det er en trekant.
O6G2-18	Elev 4	Gå 50 steg. Skal den fortsatt gå det?
O6G2-19	Elev 5	Ja, eller det har vel ingenting å si.
O6G2-20	Elev 4	Så var den andre grønne tingen. Delegreia.
O6G2-21		*Henviser til operatørblokka for divisjon. Skriver inn 360 som divisor og 5 som divisor.*
O6G2-22	Elev 5	Blir det ikke 3?
O6G2-23	Elev 4	360 delt på 5 grader. Jeg vet ikke helt.
O6G2-24		*Forsøker å kjøre programmet.*
O6G2-25	Elev 4	Han tegnet bare en halv. Han må snu… Åja, det var det du mente med delt på 3!
O6G2-26	Elev 5	Ja
O6G2-27		*Endrer divisor til 3.*
O6G2-28	Elev 4	Woo! Trekant. Hvordan får man den til å bli andre veien? [*henviser til at den likesidete trekanten peker nedover*]
O6G2-29	Elev 5	Andre vei?
O6G2-30	Elev 4	Ja, trekanten den er jo opp-ned.
O6G2-31	Elev 4	Vi lager heller firkant.
O6G2-32	Elev 4	da er det fire. [*Endrer divisor til 4*]
O6G2-33	Elev 4	Er det fire der også? [*Henviser til løkka*]
O6G2-34	Elev 5	Jo
O6G2-35	Elev 4	Er det noen andre plasser vi må bytte ut?
O6G2-36	Nei	Nei
O6G2-37		*De kjører programmet*

O6G2-38	Elev 5	Da er det sekskant.
O6G2-39	Elev 4	Ja, da må vi vel bare bytte på disse tallene. 6 og 6. [Endrer divisor og løkketeller til 6]
O6G2-40	Elev 4	1, 2, 3, 4, 5, 6. [*Teller antall sider i figuren*]
O6G2-41	Elev 5	Hvilke andre former klarer du å tegne? [*Leser fra oppgaven*]
O6G2-42	Elev 4	Sjukant? 21-kant? 21 [*Endrer divisor og løkketeller til 21*]
O6G2-43		*Tester programmet*
O6G2-44	Elev 4	Det funket ikke. Hvorfor funket ikke det?
O6G2-45	Elev 5	Har det noe med stegene hans å gjøre?
O6G2-46	Elev 4	Du ser jo inne der så er det mange små hakk. Men det ble jo ikke rett.
O6G2-47		*Prøver programmet flere ganger, med samme resultat.*
O6G2-48	Elev 4	Hva skjer a? Hvorfor fungerer ikke dette?
O6G2-49	Elev 5	Vet ikke
O6G2-50	Elev 4	Må han gå flere steg? Han kan gå 200 steg. [*Endrer gå-blokka til 200*]
O6G2-51	Elev 4	Det ble i hvert fall feil.
O6G2-52	Elev 4	Vi prøver med 12. [*Endrer divisor og løkketeller til 12 og steg tilbake til 50*]
O6G2-53	Elev 4	Det ble rett. Hvorfor fungerte det ikke med 21?
O6G2-54		*Endrer divisor og løkketeller til 15 og kjører programmet.*
O6G2-55	Elev 4	Der gikk den i skjeis. Hvorfor gjorde den det?
O6G2-56	Elev 5	Jeg vet ikke.
O6G2-57	Elev 5	Men hvordan lager man sirkler egentlig?
O6G2-58	Elev 4	Det er ikke en sirkel, det er en 15-kant
O6G2-59	Elev 5	Ja, men hvordan lager man en sirkel?
O6G2-60	Elev 4	Mmmm, en 100-kant? *[ler]* Da blir sidene så små at det blir en sirkel kanskje.
O6G2-61	Elev 4	Jeg vet ikke. Vi kan gå videre. Eller vent litt. Fungerer det med 14-kant? [*Endrer divisor og løkketeller til 14 og kjører programmet*]
O6G2-62	Elev 4	Jeg må bare sjekke det siden 13 fungerte.
O6G2-63		*Tester programmet*
O6G2-64	Elev 4	Ja! Det fungerer opp til 13- kant, ikke lengre.
O6G2-65	Elev 4	Lag et program der dere oppgir antall hjørner og deretter tegner en mangekant med det gitt antallet hjørner. [*leser fra oppgaven*]
O6G2-66	Elev 5	Gjør det ikke allerede det da?
O6G2-67	Elev 4	Gjenta 14 ganger. Det er jo hvor mange kanter det skal være. Er det ikke det?`
O6G2-68	Elev 5	Jo, det er sant.
O6G2-69	Elev 4	Da har vi jo allerede gjort oppgaven. Det var litt rart.
O6G2-70	Elev 5	Kanskje han vil at katten skal spørre eller noe.
O6G2-71	Elev 4	Da må vi ha spørreblokka.
O6G2-72		*Elevene bruker tid på å lete etter spørreblokka. Lærer kommer inn i rommet.*
O6G2-73		*Lærer kommer inn i rommet.*

O6G2-74	Elev 4	Vi finner ikke der den kan spørre om noe.
O6G2-75	Lærer	Okei?
O6G2-76	Elev 4	Sånn variabel-ting.
O6G2-77	Lærer	Du tenker at du ikke finner selve blokka?
O6G2-78	Elev 4	Ja
O6G2-79	Lærer	Er det ikke den du har der? Spør-blokka er jo rette der.
O6G2-80	Elev 4	Åja! [*flau latter*]
O6G2-81	Lærer	Så blir det å tenke på hvor denne passer inn i programmet.
O6G2-82	Elev 4	Det skal jo ikke gjentas. [*setter blokka inn før løkka*]
O6G2-83	Lærer	Det er fornuftig tenkt. Det blir slitsomt om spørsmålet blir gjentatt så mange ganger.
O6G2-84	Lærer	Prøv å skrive 5, bare for å teste med et tall. Hva skjer da?
O6G2-85		*Elevene prøver med 5 som input, men programmet tegner en 14-kant.*
O6G2-86	Lærer	Er det en femkant?
O6G2-87	Elev 4	Nei, det er ikke en femkant. Hva skjedde?
O6G2-88	Lærer	Hva er det dere har sagt at programmet skal gjøre?
O6G2-89	Elev 4	Han sletter jo ikke spørsmålet etterpå.
O6G2-90	Lærer	Jeg tenker spørsmålet er fint. Men hva er det løkka som kommer etter spørs målet gjør egentlig?
O6G2-91		*Lærer lar spørsmål henge i lufte, og forlater rommet.*
O6G2-92	Elev 5	Vi må bruke svar-blokka tror jeg.
O6G2-93	Elev 4	Ja, sånn ja! Du er smart! [*Endrer divisor og løkketeller til svar-blokk*]
O6G2-94	Elev 4	Si et tall
O6G2-95	Elev 5	5
O6G2-96		*Tester med 5 som input*
O6G2-97	Elev 4	Jada!
O6G2-98	Elev 4	Nytt tall. 3.
O6G2-99		*Tester med 3 som input*
O6G2-100	Elev 4	Det funker jo!
O6G2-101		*Elevene begynner på oppgave 3 der de skal på programmet til å tegne en figur som ligner på en Pac-man.*
O6G2-102	Elev 4	Der er et bilde av hvordan han fargelegger den. Strekene er mye tettere. Jeg tror de strekene må være veldig tette og så må du har et hull der. [*leser fra oppgaven*]
O6G2-103	Elev 4	Og øyet er en prikk.
O6G2-104	Elev 4	Skal vi slette det gamle programmet da?
O6G2-105	Elev 5	Ja
O6G2-106		*Fjerne alle blokkene fra forrige program*
O6G2-107	Elev 4	Hvor er slett alt?
O6G2-108	Elev 5	Det er på pennen
O6G2-109	Elev 4	Jeg glemmer det hele tiden. Pek i retning. Ja det skal være 90.
O6G2-110	Elev 5	Så er det vent 1 sekund.
O6G2-111	Elev 4	Gjente 10 ganger.

O6G2-112	Elev 5	Er det den?
O6G2-113	Elev 4	Ja
O6G2-114	Elev 4	Så er det penn av.
O6G2-115	Elev 4	Gå til x y.
O6G2-116	Elev 4	Gå 100 steg. Det var mange
O6G2-117	Elev 4	Okei, vi prøver den en gang før vi jobber videre.
O6G2-118		*Tester programmet*
O6G2-119	Elev 4	Den må jo gjentas mer enn ti ganger. Og pennfarge må settes til gul.
O6G2-120	Elev 4	Gå på penn. Der ja. Det den må settes inn etter blokka med penn på.
O6G2-121	Elev 4	Litt rar gulfarge, men det går jo fint. Så må det gjentar mange flere ganger. Vi prøver 100.
O6G2-122		*Tester programmet*
O6G2-123	Elev 5	De kommer på samme plass.
O6G2-124	Elev 4	Men det blir jo ikke gjentatt 100 ganger.
O6G2-125	Elev 4	Åja, de kommer på samme plass. Hæ? Nå skjønner jeg ikke. Skjønner meg ikke på Pac-man.
O6G2-126	Elev 5	Skal vi bruke den?
O6G2-127		*Setter inn enda en koordinatblokk og gir den andre koordinater. Tester programmet*
O6G2-128		*Elevene prøver seg frem med å endre flere av blokkene, uten at de eksplisitt uttrykker hva de gjør. Etter en del prøving og feiling kommer de frem til en konklusjon som tar de videre i riktig retning.*
O6G2-129	Elev 4	Åh, det har noe med snu 36 grader som har noe å si for hvor nær den skal være. Snu 1 grad heller. Da blir den mye tettere.
O6G2-130		De prøver programmet
O6G2-131	Elev 4	Oi, nå ble det to figurer. Hvorfor stoppet den? Den bare stoppet midt i. og de kommer jo ikke på samme plass.
O6G2-132	Elev 4	Vi må jo ha en dobbelt en, for at den skal blir skikkelig gul, men de er jo ikke på samme plass.
O6G2-133	Elev 5	Det er kanskje derfor. [*Henviser til koordinatblokkene*]
O6G2-134	Elev 4	De er jo ikke samme. Men hvorfor står det i oppgaven at det er et tips. Må vi bruke det? [*Endrer slik at koordinatene blir like*]
O6G2-135	Elev 4	Nå ble den skikkelig gul, men den går ikke helt rundt. Kanskje den må gjentas flere ganger?
O6G2-136	Elev 5	Ja
O6G2-137	Elev 4	Hvis vi velger 150 da.
O6G2-138	Elev 5	Den skal jo ikke gå helt rundt.
O6G2-139	Elev 4	Nei, det skal den jo ikke. Må bare sjekke om den går lengre rundt nå.
O6G2-140	Elev 5	Det ble jo perfekt jo!
O6G2-141	Elev 4	Men han skal jo være lengre oppe. Mer sånn? [*Henviser til at munnen til Pac-man skal komme høyere opp i figuren*]

O6G2-142	Elev 4	Hva er det som har noe å si for hvor den skal begynne? Er det -100 som bestemmer det?
O6G2-143	Elev 5	Prøv -80
O6G2-144	Elev 4	Nei, han begynner ikke noe høyere opp. Han skal begynne litt mer skrått.
O6G2-145	Elev 5	Prøv 50 [s*om y akse*]
O6G2-146	Elev 4	Han flytter seg bare lengre opp. Åh, det er jo hvor på skjermen den skal være.
O6G2-147	Elev 4	Vi er i hvert fall godt på vei. Vi har jo en halv Pac-man
O6G2-148	Elev 5	Hva om vi bytter den. [*Henviser til blokka "pek i retning" som endres til 75*]
O6G2-149	Elev 4	Jo! Du er smart!
O6G2-150	Elev 5	Prøv å endre litt til opp.
O6G2-151	Elev 4	Nå ble den bra! Det er perfekt faktisk.
O6G2-152	Elev 5	Så er det bare prikken igjen
O6G2-153		*Lærer kommer inn i rommet igjen*
O6G2-154	Elev 4	Vi har laget Pac-man, men ikke øyet.
O6G2-155	Lærer	Så bra!
O6G2-156	Elev 4	Hvordan lager man øyet?
O6G2-157	Lærer	Hvordan laget du Pac-man?
O6G2-158	Elev 4	Snurret den rundt.
O6G2-159	Lærer	Hva er et øye?
O6G2-160	Elev 4	Åh, det er en liten runding. Men hvordan vet man plasseringen?
O6G2-161	Lærer	Jeg har jo litt dere noen hint på arket. Til koordinater som jeg har brukt. Det er ikke sikkert dere velger å bruke de samme.
O6G2-162	Elev 4	Åja, den er på den vanlige og den er til øyet.
O6G2-163	Lærer	Dere velger hvilke koordinater dere vil, de må bare passe sammen.
O6G2-164	Elev 5	Ja
O6G2-165	Elev 4	Da må vi ha en egen løkke som må gjentas sikkert 200 ganger.
O6G2-166		*Lærer forlater rommet igjen.*
O6G2-167	Elev 4	Så må vi ha penn av. Og så må han gå til. Da skal han gå til -80 og 50. [Refererer til koordinatene]
O6G2-168	Elev 4	Trenger vi egentlig denne delen? [*Henviser til at den opprinnelige løkka har dobbelt opp med noen blokker*]
O6G2-169	Elev 4	Skal vi prøve uten den? Bare for å se hvordan det er?
O6G2-170		*Tester programmet*
O6G2-171	Elev 4	Tror bare han lager det samme og at de blokkene er unødvendige. Nei, det gjorde den ikke. [*Programmet tegner halvparten så stor sektor som tidligere*]
O6G2-172	Elev 4	Da må jo bare løkka gjentas flere ganger. Det er jo bare det.
O6G2-173	Elev 4	Vi endrer den til 260 ganger, i stedet for 160.
O6G2-174	Elev 4	Ne, litt til. 300
O6G2-175	Elev 4	Åh! 360 blir vel en sirkel. [*Det går opp for eleven at antall løkker korrelerer med grader*]
O6G2-176	Elev 4	Vi tar 320 så blir det passende munn.

O6G2-177	Elev 4	Dette er jo faktisk litt gøy da.
O6G2-178	Elev 5	Synes du?
O6G2-179	Elev 4	Ja, det er jo gøy å lage en Pac-man.
O6G2-180	Elev 4	Da trenger vi ikke disse. Bye bye. [*Eleven sletter blokkene som ble tatt ut av kodekjeden*]
O6G2-181	Elev 4	Penn på. Og så er det pennfarge svart i hvert fall. Black
O6G2-182	Elev 4	Gå 100 steg.
O6G2-183	Elev 5	Er det ikke bare noen få steg?
O6G2-184	Elev 4	Vi prøver med 1 steg.
O6G2-185	Elev 4	og så 1 grad.
O6G2-186		*Tester programmet*
O6G2-187	Elev 4	Oi, den ble liten. Vi prøver heller med 10 steg.
O6G2-188	Elev 4	Er ikke det litt stort da?
O6G2-189	Elev 5	Neida.
O6G2-190	Elev 4	Dette ble fint.
O6G2-191	Elev 5	Bra jobba!