BUKS – Tidsskrift for Børne- og Ungdomskultur
Nr. 71 2025 • Årgang 41 • ISSN online 2446-0648 • www.buks.dk

Thilde Emilie Møller, Eva Petropouleas, Pernille Lomholt,
Tomas Højgaard & Mie Buhl

Børn koder kunst: Computationel tankegang som en vej til kreativ udtryksform

Resume

I projektet *Algoritmisk kunst* undersøges, hvordan computationel tankegang manifesterer sig som koncepter, praksisser og perspektiver i elevernes arbejde med kreativ kodning i tværfaglige forløb. Analysen viser, at eleverne gradvist udvikler forståelse for computationelle koncepter som funktioner, gentagelser, vilkårlighed og interaktion og anvender dem i eksperimenterende praksisser præget af *trial and error* mere eller mindre fagligt funderet. Eleverne forhandler mellem idé, teknologi og æstetisk intention. Resultaterne peger på begyndende tegn på *computational empowerment*, hvor eleverne oplever handlekraft og ejerskab i deres digitale skaben.

Nøgleord: Algoritmisk kunst; kreativ kodning; teknologiforståelse; computationel tankegang; computational empowerment

Introduktion

Algoritmisk kunst er en del af samtidens kunstformer og kendetegnet ved at være konceptuel, interaktiv, iterativ og socialt involverende. Udtryksrepertoiret afspejler dermed praksisformer i samtidens visuelle kultur, som børn og unge deltager i som brugere af sociale medier og i gaming. »Jeg spiller også et spil om krig … jeg kan ikke forstå, hvordan de kan kode det…«. Denne udtalelse fra en elev i 4. klasse, der lærer at kode kunst, viser, hvordan børn kan få blik for deres mediekulturelle verden som noget, der er skabt gennem algoritmer.

Genstandsfeltet algoritmisk kunst kræver arbejde med kreativ kodning, der er en praksis, hvor programmering bruges som værktøj til at udforske og udtrykke kreativitet inden for kunstneriske domæner (Petropouleas, 2024). I stedet for at se programmering som en teknisk disciplin betragtes den som et kreativt medium, hvor koden fungerer som kunstnerisk materiale og kan skabe interaktive værker, visuelle effekter eller lydinstallationer (Petropouleas, 2024). I projektet *Algoritmisk kunst* lærer børn at kode i OpenProcessing, en online platform baseret på *p5.js* (JavaScript).

Artiklen undersøger, hvordan børn arbejder med kreativ kodning, hvordan de reflekterer over deres processer og hvilke værker, de skaber. Disse områder belyses gennem kompetenceområderne 'computationel tankegang' (CT) og 'teknologisk handleevne' (TH) fra forsøgsfaget teknologiforståelse, som blev afprøvet mellem 2018 og 2021.

Forskningsspørgsmål:
Hvordan manifesterer computationel tankegang sig i elevernes arbejde med at nedbryde og forme ideer til digitale værker i tværfaglige forløb om algoritmisk kunst?

State of the art

Kreativ kodning integrerer programmeringssprog som medium for kunstnerisk og personligt udtryk (Greenberg, 2007; Peppler & Kafai, 2009). I 2006 genintroducerede Wing (2006) begrebet *computational thinking* som den måde, en datalog tænker på, og hun argumenterede for, at CT er en grundlæggende færdighed for alle. Siden har CT fået politisk og videnskabelig udbredelse og indgået i mange læseplaner internationalt. I de fleste definitioner forstås CT som evnen til at nedbryde problemer (dekomposition), forenkle ideer (abstraktion), forstå og anvende algoritmer (algoritmisk tænkning), genkende mønstre og arbejde iterativt gennem *debugging* (Grover & Pea, 2013; Shute et al., 2017).

Et studie fra New Zealand (Woo & Falloon, 2022) viser, at elever, som er nybegyndere i kodning, ofte anvender *trial and error* eller *copy-paste*-strategier, som forskerne betegner som »low level strategies«. Mange elever omgår problemer frem for at løse dem, fx ved at ændre den oprindelige idé i stedet for at fejlfinde koden. Forfatterne er kritiske over for antagelsen om, at kreativ kodning automatisk fremmer CT, og peger på behovet for en dybere forståelse af kodningskoncepter, før elever kan anvende CT-strategier som dekomposition og abstraktion.

Et amerikansk studie (Unahalekhaka & Bers, 2022) viser, at selv små børn med begrænset kodningserfaring kan udvise høj intention og kreativitet i deres projekter. Ved at indføre begrebet *purposefulness* viser studiet, hvordan børn bruger lyd, billeder og tegninger meningsfuldt, selv med enkel kode. Dermed argumenteres der for, at tekniske færdigheder og designmæssige kvaliteter bør vurderes samlet. Netop i samspillet mellem kodning og design udfolder børns kreative potentiale sig.

Computationelle koncepter, praksisser og perspektiver

Denne artikel tager udgangspunkt i Brennan og Resnicks (2012) teoretiske rammesætning for CT, fordi det tilbyder en bred forståelse, der ikke kun fokuserer på tekniske færdigheder, men også på tilgange og computational empowerment, eller digital myndiggørelse, som det betegnes i en dansk kontekst.

Da eleverne i projektet har arbejdet med kunst og teknologi snarere end med problemløsning i klassisk forstand, er denne teoretiske rammesætning velegnet som fundament for både undervisningsudviklingen og analysen. Det har muliggjort en tæt kobling mellem teori og praksis, hvor lærerne anvendte rammesætningen aktivt i planlægningen af forløbene.

Følgende teoretiske afsnit beskriver Brennan & Resnicks tredimensionelle rannesætning der indeholder computationelle koncepter, computationelle praksisser og computationelle perspektiver. Slutteligt er et teoretisk afsnit om computational empowerment som supplement til Brennan & Resnicks tredje dimension i rammesætningen, computationelle perspektiver.

Computationelle koncepter

De computationelle koncepter omfatter grundlæggende strukturer i programmering, som sekvenser, løkker, hændelser, betingelser, operatorer og data (Brennan & Resnick, 2012, s. 3). Disse danner byggestenene i algoritmisk tænkning og kan knyttes til elevernes arbejde med at udforme og strukturere digitale produkter.

Computationelle praksisser

Brennan og Resnick understreger, at CT også handler om, *hvordan* man lærer, altså om refleksion og proces. Computationelle praksisser omfatter blandt andet afprøvning og fejlfinding (debugging), trinvis og iterativ udvikling, genbrug og omstrukturering af kode samt arbejdet med dekomposition og abstraktion (Brennan & Resnick, 2012, s. 7).

McCauley et al. (2008) beskriver debugging som både en teknisk og kognitiv aktivitet, hvor den lærende udvikler en forståelse af årsagssammenhænge i programmet. Effektiv fejlfinding indebærer systematiske strategier som at gennemgå koden linje for linje, undersøge output eller teste hypoteser om fejlens årsag. Studier viser, at begyndere inden for programmering ofte eksperimenterer uden systematik, mens erfarne programmører tester og reflekterer mere målrettet (McCauley et al., 2008). For at støtte denne udvikling bør undervisningen eksplicit stilladsere strategier for debugging (Angeli, 2022; Kim et al., 2022), så eleverne lærer at se fejlfinding som en meningsfuld og refleksiv del af den kreative proces.

Computationelle perspektiver

Den tredje dimension, computationelle perspektiver, handler om at bruge teknologi som udtryksform, som refleksion og som kritisk praksis (Brennan & Resnick, 2012, s. 10). Ifølge forfatterne bør unge føle sig empowered til at stille spørgsmål både *om* og *med* teknologi. Dette perspektiv forbindes med Iversen et al. (2018, 2022), som definerer computational empowerment som en kombination af praktisk-skabende og refleksiv aktivitet, »hands-on/mind-in«. Elever skal ikke blot bruge teknologi, men forstå og forholde sig kritisk til dens rolle i samfundet og i deres eget liv.

Selvom Brennan og Resnick selv anerkender, at grænserne mellem koncepter, praksisser og perspektiver ofte flyder sammen, kan opdelingen fungere analytisk. Den giver et blik for, hvordan CT manifesterer sig i elevernes konkrete handlinger og refleksioner.

Computational empowerment

Begrebet computational empowerment tilbyder en ramme til at forstå, hvordan arbejdet med algoritmisk kunst kan styrke elevernes oplevelse af myndiggørelse.

Empowermentbegrebet bruges ofte bredt, men kan blive uklart i sin betydning. Van Mechelen et al. (2021) peger på, at begrebet rummer mange uafklarede forståelser, hvilket risikerer at udvande dets analytiske kraft. Iversen, Dindler & Smith (2022) definerer computational empowerment som udviklingen af evner til at engagere sig kreativt i teknologiudvikling, forstå teknologiens samfundsmæssige rolle og forholde sig kritisk til dens betydning i eget liv.

Dette perspektiv har rødder i både Paperts (1993) konstruktionisme og den participatory design-tradition, hvor empowerment forstås som en demokratisk og kritisk relation til teknologi (Dindler, Smith & Iversen, 2022). Paperts begreb *mathetics*, at lære gennem at skabe, betoner, at læring sker, når børn oplever, at de selv kan udforske og forstå gennem konkrete erfaringer. Det er i mathetics, at børnene empower egen læreproces.

Katterfeldt et al. (2015) kobler dette til et dannelsesperspektiv, hvor empowerment handler om at kunne materialisere egne ideer gennem digitale teknologier og opnå en følelse af *self-efficacy* (Bandura, 1997), som betyder at kunne mestre og forholde sig bevidst til teknologiens muligheder og begrænsninger. Set i dette lys handler empowerment ikke kun om teknisk kunnen, men om at blive en aktiv deltager i den digitale kultur og at kunne forme og fortolke teknologiens rolle i egne og fælles udtryksformer.

Baggrund

Projektet

Studiet tager udgangspunkt i udviklings- og forskningsprojektet *Algoritmisk kunst* (2023-2025), som havde til formål at styrke læreres kompetencer til at undervise tværfagligt om og med digital teknologi i fagene matematik, billedkunst og teknologiforståelse. Projektet var forankret i en design-based research-tilgang (Amiel & Reeves, 2008; Brown, 1992), der kombinerede udvikling og forskning gennem gentagne afprøvninger i praksis.

Gennem to år har lærere og forskere deltaget i fælles seminarer og workshops, hvor ny viden blev omsat til undervisningsforløb, afprøvet i praksis og efterfølgende evalueret i fællesskab. Denne cykliske proces gjorde det muligt løbende at justere forløbene ud fra erfaringer i klasserne og skabe tæt kobling mellem teori og praksis.

Kompetenceudvikling og co-teaching

Lærernes kompetenceudvikling omfattede to introduktionsseminarer, tre tematiske workshops og en afsluttende opsamlingsdag pr. projektår. Mellem disse afviklede lærerne deres egne afprøvninger i undervisningen. Et centralt element var co-teaching (Højholt, 2017), hvor lærere samarbejdede med konsulenter om planlægning og gennemførelse.

Deltagere

I projektet deltog 40 lærere fra otte skoler og over 500 elever fra 3.-8. klasse. Lærerne repræsenterede fagene matematik, billedkunst og teknologiforståelse. Til artiklen er seks undervisningsforløb udvalgt med 13 lærere og 149 elever som datagrundlag.

Forskningsdesign

Artiklens data består af noter, fotos, videooptagelser og elevernes digitale værker inspireret af kvalitative metoder i praksisnær skoleforskning (Møller et al., 2021). Derudover er der gennemført semistrukturerede gruppeinterviews med 19 elever. Tre forskere foretog deltagende observationer i 128 lektioner, hvor de førte feltnoter, fotograferede og havde samtaler med elever og lærere. Empirien blev efterfølgende diskuteret i forskergruppen gennem en iterativ analyseproces, hvor empiri først blev kodet tematisk ud fra Brennan & Resnicks (2012) tre CT-dimensioner: koncepter, praksisser og perspektiver. Analysen fokuserede på mønstre i observationer, elevprodukter og interviews. Herefter identificerede vi i gruppen repræsentative eksempler på, hvordan CT manifesterede sig i praksis (Merriam, 1998). Disse eksempler har vi analyseret med udgangspunkt i enkelte nedslag fra empirien. Herunder er et overblik over studiets samlede empiri.

Skole og klasse	**Forløb**	**Antal elever**	**Antal lektioner**	**Empiri**	**Antal lektioner ifht. dataproduktion**
Skole A, 7. årgang	Mesterværk: Der skal være bevægelse i værket. Tænk i mønstre, geometriske former, farvesammensætning, komposition og bevægelse. Opgaveformulering og et moodboard . Dit mesterværk skal afspejle din opgaveformulering og dit moodboard.	18	4 uger, 15 lektioner	Video, foto, noter, samtaler	16 (to personers observation til sammen)
Skole B, 4. årgang	Mondrian: Farver og former. Lav et værk inspireret af Mondrian.	44	5 uger, 12 lektioner	Video, foto, noter, 9 gruppeinterviews (19 elever deltog i interviews)	20 (to personers observation tilsammen)
Skole B, specialklasse	Vilkårlighed og dynamik	9	3 dage, 18 lektioner	Foto, noter, samtaler	20 (to personers observation til sammen)
Skole B, 8. årgang	Snefnug	15	6 lektioner	Video, foto, noter og samtaler	8 (to personer til sammen)
Skole C, 4. + 5. årgang	Byg dit eget blomstertæppe.	22	4 dage, 24 lektioner	Foto, noter og samtaler	30 (to personers observation til sammen)
Skole D, 4. og 5. årgang	Designer og programmør	41	2 dage, 12 lektioner	Foto, noter, samtaler	28 (tre personers observation til sammen)

Resultater

Analysen er som tidligere nævnt struktureret ud fra Brennan og Resnicks (2012) rammesætning for CT, som omfatter de tre gensidigt forbundne dimensioner: computationelle koncepter, computationelle praksisser og computationelle perspektiver. Det første afsnit fokuserer på de computationelle koncepter, som eleverne måtte tilegne sig og anvendte i deres værker. Det andet afsnit belyser de computationelle praksisser, hvor eleverne eksperimenterede, afprøvede og fejlrettede gennem kreative processer. Det tredje afsnit udfolder de computationelle perspektiver, hvor eleverne begyndte at udtrykke sig, skabe forbindelser og reflektere kritisk gennem arbejdet med algoritmisk kunst, hvilket vi anskuer som en begyndende computational empowerment.

Computationelle koncepter

I følgende afsnit er fokus på elevernes anvendte computationelle koncepter. Projektets valg af programmeringsværktøj, OpenProcessing, har fra starten betydet, at eleverne har skullet arbejde med at forstå computationelle koncepter eksplicit. Mens andre, mere visuelle og intuitive programmeringsværktøjer, lægger op til eksperimenter og prøven sig frem, kræver OpenProcessing, at eleverne har en grundlæggende viden, før de kan bruge værktøjet.

Grundlæggende computationelle koncepter

Overordnet handler de grundlæggende koncepter bl.a. om at forstå forskellen på henholdsvis opsætnings- og tegnefunktionen, brug af store og små bogstaver, og hvordan statiske objekter (geometriske figurer) konstrueres. En cirkel tegnes fx ved at skrive navnet »circle« efterfulgt af en parentes, som indeholder tre argumenter i en bestemt rækkefølge adskilt af komma, inden instruktionen afsluttes med semikolon:

circle (X-punkt for cirklens centrum, Y-punkt for cirklens centrum, cirklens diameter);

For så vidt kan eleverne godt lære syntaksen »circle(et tal, et tal, et tal);« uden at forstå den semantiske betydning, nemlig at de to første tal beskriver placeringen af cirklens centrum på x og y-aksen, og det sidste tal angiver cirklens diameter, men uden den semantiske betydning vil eleverne ikke vide, hvilket tal, der flytter cirklen til højre/venstre, op/ned eller gør den større/mindre.

Eleverne har også skullet forstå, at programmet tegner sekventielt fra top til bund i koden, og at rækkefølgen af instruktioner derfor er helt afgørende. Farver skal fx angives, før objektet tegnes, og hvis det endelige projekt udgøres af flere objekter ovenpå hinanden som øjet nedenfor, skal instruktionerne arrangeres, så det øverste objekt (pupillen) tegnes sidst, da den ellers vil blive skjult af den blå iris.

```
mySketch
function setup() {
  createCanvas(600, 600); //opretter lærred 600 x 600
  background('white'); //gør baggrund hvid
}

function draw() {
  stroke('black');//Konturlinje oprettes med farven sort
  strokeWeight(10); //Konturlinjens tykkelse angives til 10 punkter bred
  noFill(); //Ingen udfyldning, da lærredets baggrundsfarve er hvidt
  circle(300, 300, 200); //Bagerste cirkel tegnes med centrum X 300 og Y 300 og med diameter 200
  noStroke(); //Konturlinje ophæves
  fill('blue') //Udfyldning med blå
  circle (300, 300, 100); //Blå cirkel tegnes med centrum X 300 og Y 300 og med diameter 100
  fill('black') //Udfyldning med sort
  circle (300, 300, 50); //Sort cirkel tegnes med centrum X 300 og Y 300 og med diameter 50
}
```

Billede 1: Find koden digitalt her, hvor du kan eksperimentere med forskellige ændringer. Du kan fx ændre farver, størrelser eller måske prøve at bytte rundt på nogle af instruktionerne: https://openprocessing.org/sketch/2656748

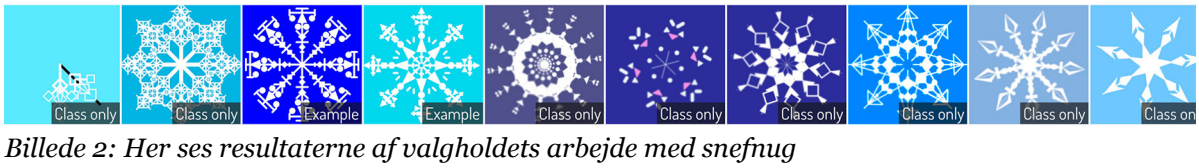
Behovet for at forstå disse grundlæggende koncepter fra starten har betydet, at eleverne i deres allerførste værker har arbejdet vidensbaseret og systematisk, både med design af deres idé og med debugging. De har alle forholdsvis let mestret at lave små figurative og statiske værker ud fra bevidste sammensætninger af former og farvevalg.

Nye koncepter og øget kompleksitet

Efter de indledende øvelser har eleverne arbejdet videre med koncepter som betingelser og gentagelser, vilkårlighed, bevægelse, interaktion og modularisering vha. nye funktioner. Hver af disse koncepter har givet eleverne nye udtryksmuligheder, men har også øget kompleksiteten betydeligt. I denne fase har vi på tværs af klasser og årgange set en langt større spredning i elevernes tilgange til at nå i mål med at kode deres idéer. Dette udfoldes i analyseafsnittet om computationelle praksisser. I dette afsnit vil vi i stedet se på, hvordan nogle af de nye kodekoncepter har manifesteret sig i tre forskellige klasser.

Valghold i billedkunst, 8. årgang: Snefnug

Valgholdet skal i gang med et tema om metamorfose, og i den forbindelse skal eleverne bl.a. designe og kode et snefnug, der jo netop er vand i en bestemt tilstand.



Billede 2: Her ses resultaterne af valgholdets arbejde med snefnug

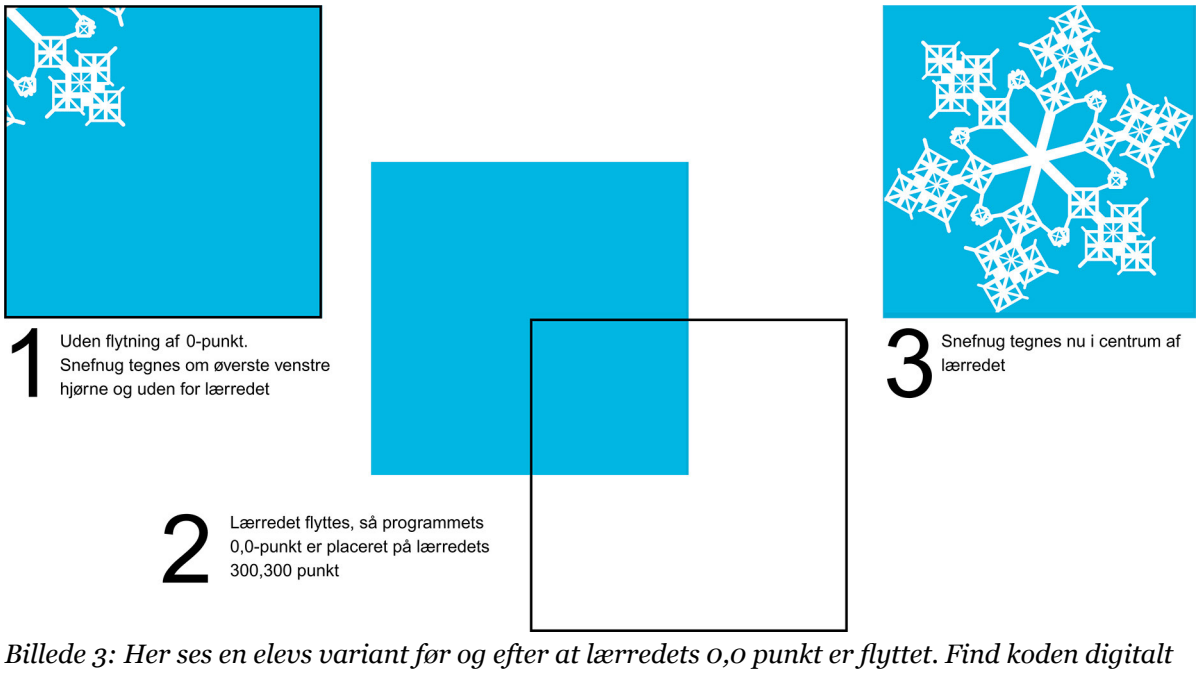
Eleverne har arbejdet med OpenProcessing tidligere og har ret godt styr på de grundlæggende koncepter. De instrueres i første trin i at lave et snefnug. De skal oprette et lærred med størrelsen 600 x 600 og kode en enkelt arm på snefnugget ved hjælp af linjer og geometriske former, som skal starte i punktet 0,0 (øverste venstre hjørne på lærredet). Det er nemt konceptuelt, men kræver en del kodelinjer, hvis armen skal være detaljeret.

De øvrige trin består i at:

– gruppere armens instruktioner i en ny funktion, som ligger uden for tegnefunktionen,
– lave et funktionskald i tegnefunktionen, som henter koden fra den nye funktion
– oprette en variabel, som bruges i de instruktioner, der får armen til at rotere om dens eget udgangspunkt med et valgfrit antal grader,
– flytte lærredet, så dets midtpunkt er samme sted som programmets 0,0 (det gør, at snefnugget ikke roterer om venstre øverste hjørne på lærredet, men om midten af lærredet.

Eleverne bliver udfordrede, da der er virkelig mange nye koncepter i spil på én gang, og det er en stor hjælp, at lærerne har forberedt små eksempelprogrammer, der trin for trin introducerer de ukendte kodeinstruktioner. Det gør, at eleverne kan fokusere deres arbejde på at programmere selve armen og eksperimentere med baggrundsfarver, rotationsgrader og filtre, heriblandt et, som slører det tegnede og giver det et lidt loddent udtryk, der passer godt til sne. Der er stor spredning i elevernes interesse for og forståelse af de nye koncepter.

Nogle elever, især dem, som ikke som udgangspunkt synes, programmering har noget med billedkunst at gøre, starter med at lave simple arme, fordi de ikke orker at lave alle de kodelinjer, som deres skitserede idé kræver, men da først programmet er færdigt, og de oplever deres færdige snefnug, får de alligevel lyst til at eksperimentere videre og udbygge armen, så udtrykket bliver mere lig deres idé. Andre elever synes det er fedt med kodning, og de er ambitiøse fra starten og knokler på for at få lavet så mange detaljer som muligt.



Billede 3: Her ses en elevs variant før og efter at lærredets 0,0 punkt er flyttet. Find koden digitalt her, hvor det er muligt at se, hvor mange koderinstruktioner, eleven har lavet for at skabe den meget detaljerede arm. Det er også muligt at eksperimentere med forskellige ændringer direkte i koden: https://openprocessing.org/sketch/2656751

mySketch

```
let angle = (0);//opretter variabel til at beregne vinkel og sætter værdi til 0

function setup() {
  createCanvas(600, 600); //opretter lærred 600 x 600
  background('■rgb(48,48,153)'); //angiver baggrundsfarve v. hj. af RGB værdier
  angleMode(DEGREES); //bestemmer at vinkel skal beregnes i grader
}

function draw() {
  translate(300,300); //flytter lærredets 0,0-punkt til midten af lærredet
  rotate(angle);//bestemmer, at objektet skal rotere med den værdi, som variablen "angle" indeholder
  minTegning(); //funktionskald - programmet tegner det, der er skrevet i funktionen "minTegning"
  angle = angle+360/6;//Tilskriver ny værdi til variablen "angle", nemlig den værdi, den allerede har
        //+ 360/6, dvs. 60. Ved hvert gennemløb roteres objektet altså 60 grader, inden det tegnes.
}

function minTegning(){
  //Her skrives de koder, der tilsammen danner en arm af snefnugget.
  //Koderne kan bestemme farve, konturlinjer, udfyldning på de former, der tilsammen udgør armen.
  //Eleverne har brugt cirkler, rektangler, linjer, ellipser og trekanter til at modellere deres individuelle
  //udtryk.

}
```

Billede 4: Grundprogrammet for snefnuggene, som eleverne skulle anvende, for at rotere og tegne deres arm i midten af lærredet.

4. og 5. årgang: Blomstertæppe

Inspireret af snefnugsprojektet vil en anden skole arbejde med samme idé, men med 4. og 5. årgang og med blomster som tema i stedet for snefnug. Grundprogrammet er altså det samme, men blomsterbladene kan tegnes med langt færre former end snefnugarmene, og på den måde gøres opgaven lettere at kode for eleverne. Der lægges også vægt på, at eleverne forstår semantikken i grundprogrammet, altså hvad de enkelte kodeinstruktioner gør, men ikke syntaksen, bortset fra de få steder, de kan ændre i koden, bl.a. i angivelsen af antal grader, der skal roteres med. Det er ikke vigtigt, at de kan genskabe grundprogrammet på egen hånd, men at de får brugt programmet til at lave deres egne blomstervarianter.

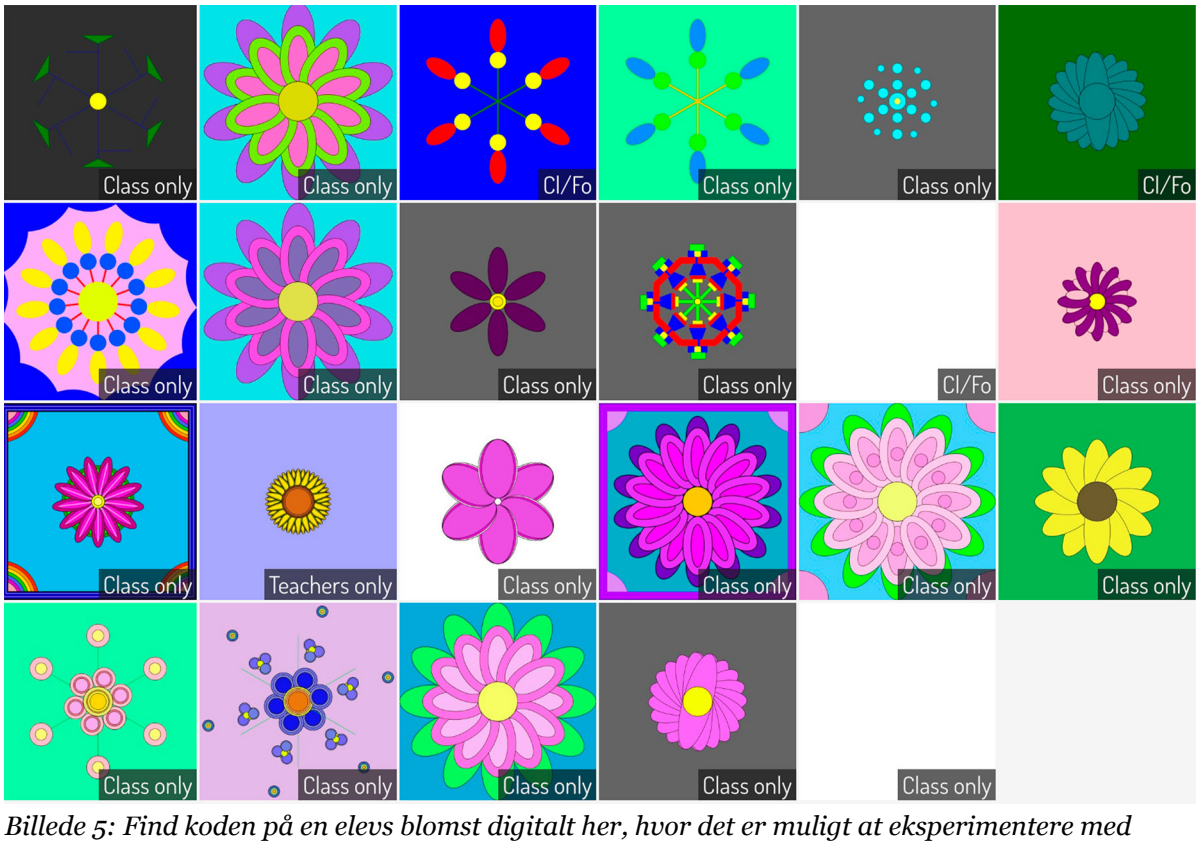
Til gengæld lægger temaet blomster op til at gøre mere ud af farvevalg og –kombinationer, end snefnuggene, der skulle være hvide på en blå baggrund.

En stor del af eleverne vælger hurtigt at samarbejde, selvom de laver hver sin blomst, og det ses tydeligt, da eleverne præsenterer deres individuelle resultater, da nogle af blomsterne har samme udtryk, men med personlige farvevalg, størrelser og antal kronblade. Det drejer sig om:

- Blomst nr. 2, 7, 16, 17, 21: Fem store blomster, som ligner hinanden, men farverne og antallet af kronblade er forskellige. En af dem har i øvrigt små cirkler i midten af kronbladene, som giver en særlig effekt (blomst 21).
- Blomst nr. 3, 4, 7: Tre abstrakte blomster med forskellige farvevalg. Nr. 7 skiller sig mærkbart ud ved at have mere end dobbelt så mange kronblade, en cirkel i midten og desuden en blå ramme.
- Blomst nr. 19, 20: To detaljerige blomster, som udover forskellige farvevalg også har eksperimenteret med forskellige måder at lave »støvdragere« på.
- Blomst nr. 6, 22: To blomster, hvor kronbladene er lavet vha. ovaler, der går hen over midten. Dette gør, at man meget tydeligt ser, hvor programmet på et givent tidspunkt er i gang med at tegne.

De øvrige elevers værker har hver sit eget udtryk, særligt blomst nr. 1, 5, 10 og 14.

Derudover har eleverne ladet sig inspirere af hinanden på tværs, hvilket vi ser i de værker, som benytter former i yderkanten af lærredet til enten at lave en ramme langs alle kanter (blomst nr. 7, hvor en stor blå cirkel roterer rundt delvist uden for lærredet) eller markerede hjørner (blomst nr. 13, 16 og 17).



Billede 5: Find koden på en elevs blomst digitalt her, hvor det er muligt at eksperimentere med forskellige ændringer: https://openprocessing.org/sketch/2656753

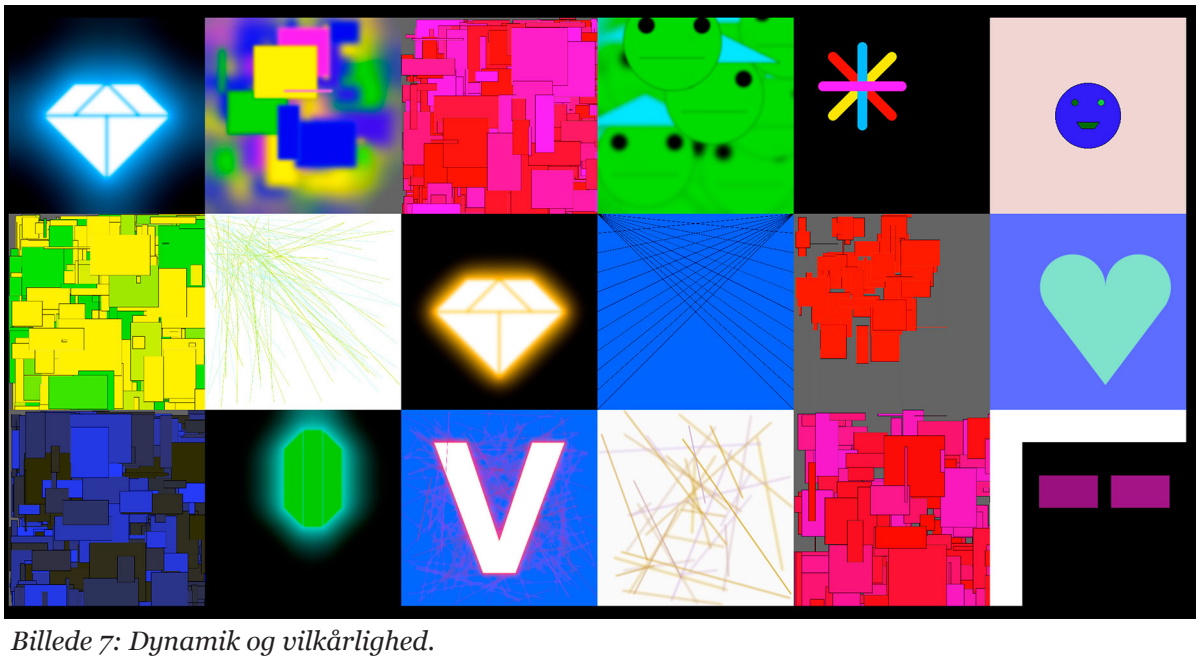
Specialklasse 4.-5. årgang: Vilkårlighed og dynamik

I denne klasse vil lærerne gerne fokusere på noget af det, der er særligt ved at kode frem for at tegne i hånden, og de vælger derfor at lave et forløb, der tager udgangspunkt i vilkårlighed som koncept. Det betyder, at eleverne skal overlade nogle beslutninger til computeren. Det kan fx være placeringen af objekterne, deres størrelser eller deres farver. Her bevæger eleverne sig ind på et felt, hvor computeren bliver medskaber af det færdige resultat. Elevernes værker skal desuden være dynamiske, dvs. ændre sig løbende, når de afspilles. Eleverne synes, opgaven er sjov, og de får mange forskellige idéer, som de afprøver.

```
mySketch

function setup() {
  createCanvas(600, 600);
  background(100);
}

function draw() {

fill(200,0,random(0,200));
//farve angives v.hj.a. RGB (Red = 200, Green = 0, Blue = vilkårligt mellem 0 og 200)

rect(random(0,600),random (0,600), random(0,200), random(0,200));
//Rektangler placeres vilkårligt med X mellem 0 og 600 og Y mellem 0 og 600 (øverste venstre hjørne)
//Derefter angives bredde vilkårligt mellem 0 og 200, og længde som vilkårligt mellem 0 og 200.

}
```

Billede 6: Her ses en elevs kode, som på meget simpel vis skaber noget meget virkningsfuldt. Find koden her og afprøv den: https://openprocessing.org/sketch/2656755

En enkelt elev bliver dog optaget af, at computeren også kan hjælpe med at udføre noget helt nøjagtigt, som man beder den om. Det handler om fjerde værk på midterste linje, hvor linjernes placeringer netop ikke er vilkårlige. Da det lykkes, udbryder eleven højt: »Eij, jeg troede det ville være meget sværere at kode end at tegne i hånden, men det er jo megafedt, for det kan blive PRÆCIS som jeg vil have det«.



Billede 7: Dynamik og vilkårlighed.

De øvrige elever starter med at gøre så meget som muligt vilkårligt, men får også langsomt behov for at få en vis styring over deres projekter. De begynder derfor at arbejde bevidst med at kontrollere rammerne, som computeren »eksperimenterer« indenfor, ved at skrue på spektrene for vilkårlighed.

Det ses især i værk 2, 3, 7, 11 og 13, som alle er vilkårlige firkanter, men hvor værk 2 benytter fire farver og sløringsfilter, værk 3, 7, 11, 13 og 17 benytter variationer inden for forskellige farveskalaer, og værk 11 benytter præcis samme farve og i øvrigt definerer et mindre område på lærredet, hvor firkanterne kan genereres.

Lærernes ambition om, at eleverne skal opleve, hvordan de vha. kode kan samskabe et værk med computeren, er altså lykkedes, og i særlig høj grad hos de elever, der har forstået, at de kan bestemme, hvordan computeren eksperimenterer, ved at skrue op og ned for dens mulighedsrum.

Opsummering

Elevernes forståelse af computationelle koncepter udvikledes gradvist fra teknisk tilegnelse til meningsfuld anvendelse. Efterhånden som nye koncepter som betingelser, gentagelser, vilkårlighed, bevægelse, interaktion og modularisering vha. nye funktioner blev introduceret, steg både kompleksiteten og variationen i elevernes tilgang. Analyseeksemplerne viser, hvordan elevernes forståelse blev forbundet med æstetiske og meningsskabende dimensioner – fra de strukturerede snefnug i 8. klasse til de eksperimenterende blomster og dynamiske vilkårlighedsprojekter i de yngre klasser.

Computationelle praksisser

I følgende analytiske afsnit om computationelle praksisser er fokus særligt på CT-elementerne abstraktion, dekomposition og debugging, da vi i empirien kunne se, at det var i disse processer, at eleverne var særligt udfordret eller i disse processer, de formåede at rykke sig videre i arbejdet og i værkerne.

Abstraktion og dekomposition i 7. klasse valgfagshold i billedkunst

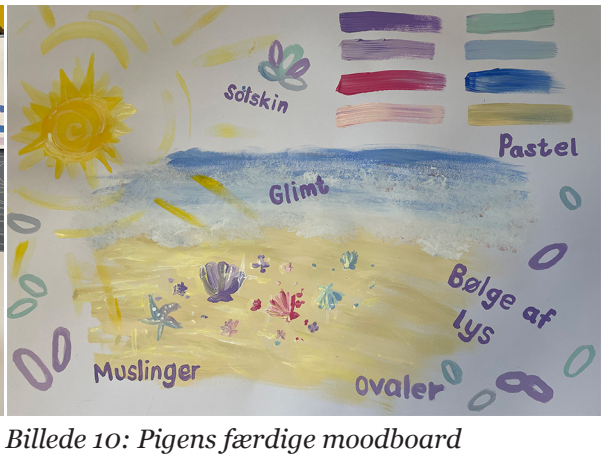
I vores observationer af elevernes arbejdsprocesser ser vi, at der særlig tit opstår udfordringer, når eleverne skal gå fra idé til kode. Særligt er dette at se i de forløb, hvor elever arbejder med mere åbne opgaveformuleringer. I et forløb om at skabe figurative og non-figurative værker på et valgfagshold i billedkunst i 7. klasse, har eleverne ikke svært ved at få ideer til deres værker. De kommer hurtigt i gang med deres analoge moodboard, som læreren har stillet som et krav til arbejdsprocessen. Derudover skal der være bevægelse i værket og eleverne skal tænke i mønstre, geometriske former, farvesammensætning og komposition. Herunder ses en af pigernes opgaveformulering, der giver udtryk for, at hun gerne vil lave et digitalt billede, som »viser en søstjerne, der ligger på en strand med en bølge der skyller op«. I forhold til mønstre og gentagelse skriver hun at »søstjernen ligger bare«; ved geometriske former, står der »stjerne«; hun vil gerne bruge »orange, blå, gul og sandgul«; ved komposition skriver hun »en søstjerne, en sandstrand, bølge«, og ved bevægelse står der »en bølge der skyller op på stranden«. Pigen ønsker ikke interaktion i værket.

- **Jeg vil gerne lave et digitalt billede som:** viser en søstjerne der ligger på en strand med en bølge der skyller op
- ***Jeg har tænk over:***
- Mønstre/gentagelse: Søstjernen ligger bare
- Geometriske former: stjerne
- Farver: orange, blå, gul, sandgul
- Komposition: En søstjerne, en sandstrand, bølge
- Bevægelse: En bølge der skyller op på stranden
- *Eventuelt interaktion??* nej

Billede 8: Pigens opgaveformulering

Efter at have udarbejdet denne opgaveformulering arbejder pigen på sit moodboard. Hun har valgt at anvende maling til at lave sit moodboard, som kan ses herunder.

Billede 9: Pigen er i gang med at male sit moodboard

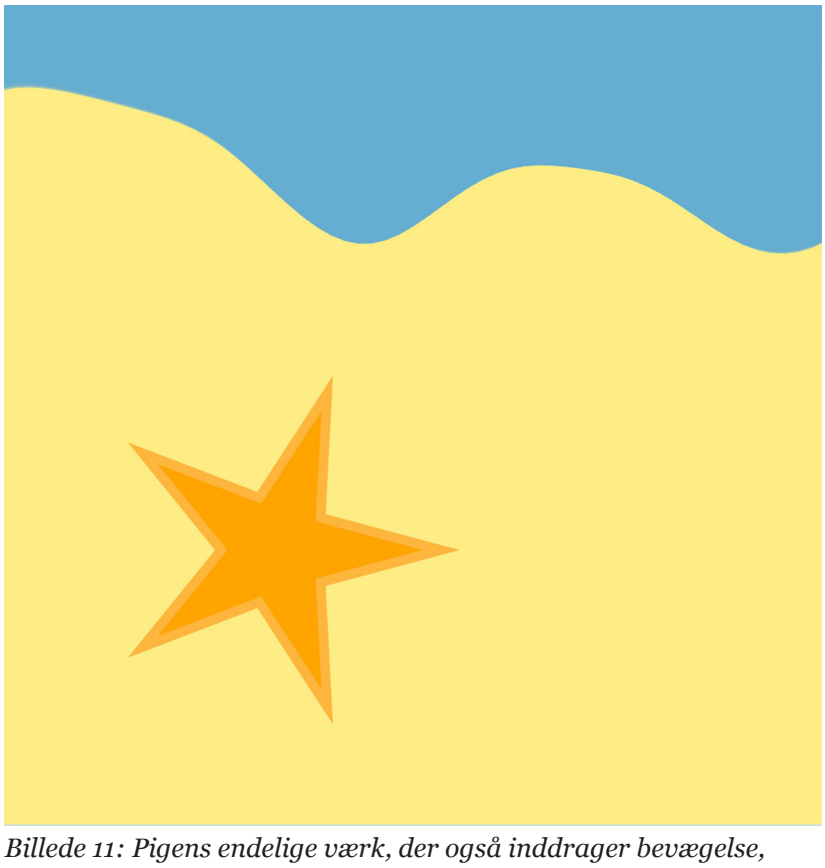


Billede 10: Pigens færdige moodboard

Efter at have fået godkendt sit moodboard af læreren, åbner pigen OpenProcessing. Hun støder på udfordringer, da hun skal kode sin muslingeskal, og åbner ChatGPT for at få hjælp. Efter flere iterationer med at prompte ChatGPT på engelsk, hvor hun skriver »seashell« og guider ChatGPT til at udvikle koden i kodesproget p5.js, finder pigen ikke resultatet tilfredsstillende, og hun erfarer, at hendes færdigheder ikke rækker til at kode en muslingeskal, heller ikke med hjælp fra ChatGPT.

Eksemplet viser, hvordan arbejdet med egne idéer aktiverer og udfordrer elevens computationelle tankegang, særligt i form af CT-praksisser som abstraktion og dekomposition. Muslingeskallen bliver for hende det essentielle i værket. Undervejs erfarer hun, at hendes tekniske færdigheder sætter grænser for, hvad hun kan realisere. Hermed bliver det tydeligt, at det at tænke computationelt også indebærer at kende egne begrænsninger og mulighedsrum i forhold til teknologiens affordances. Vi ser, at de fleste elever ændrer i deres intention, når det ikke helt bliver, som de havde troet, hvilket Woo & Falloon (2022) betegner »work around« (s. 8). Denne form for »work around« ser vi dog ikke så entydigt på i forhold til

at ændre på intentionen. Når elever beslutter at nedjustere sine æstetiske ambitioner for at opnå kontrol over det digitale udtryk, er det ikke altid blot en ændring i intentionen og en teknisk tilpasning – vi ser det også som en praksis, hvor eleven forhandler og balancerer mellem egen forestillingsevne og teknologisk mestring. For hos nogle elever har der forinden været ihærdige forsøg og tid med at løse udfordringen. Som vi så med muslingeskallen, bliver kreativ kodning en proces, hvor eleverne træffer meningsfulde valg, og hvor det at mestre teknologien betyder at kunne bruge den til at forme noget meningsfuldt.

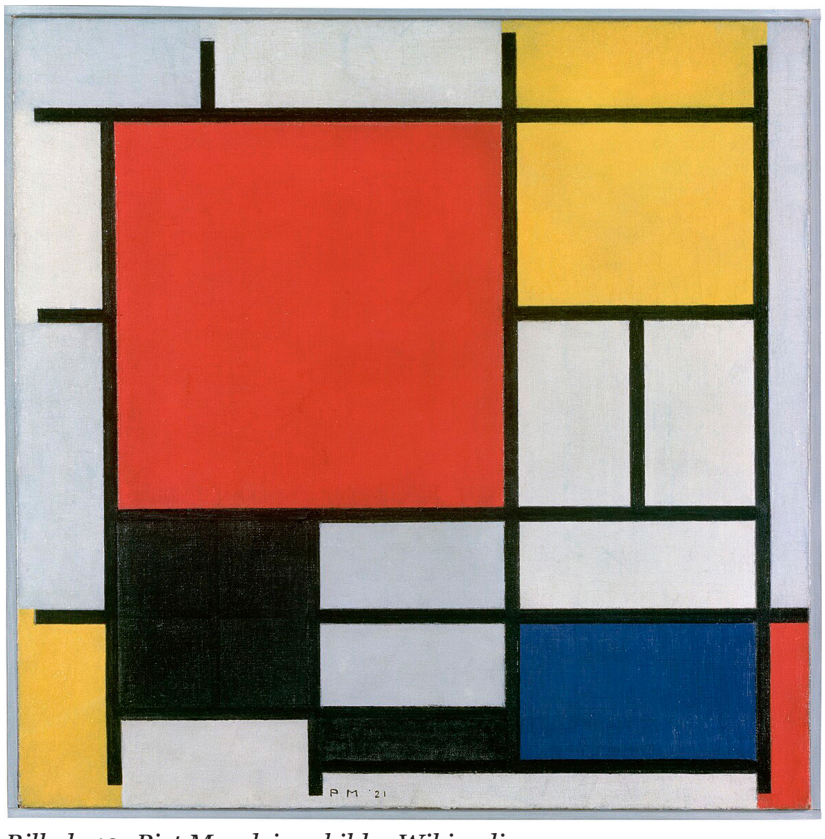


Billede 11: Pigens endelige værk, der også inddrager bevægelse, der ligner blå bølgeskvulp.

Debugging i 4. klasse

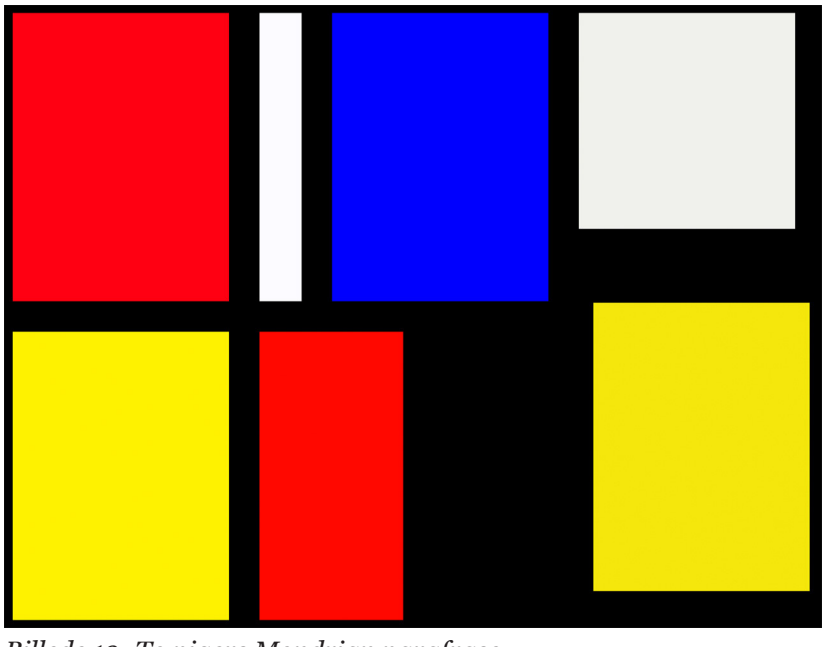
I vores empiri ser vi tydelige forskelle i, hvordan eleverne angriber *debugging*-processen. Når eleverne ændrer i koden uden en klar hypotese og uden at kunne forklare, hvorfor de foretager ændringen, svarer det til det, McCauley et al. (2008) betegner som *brute-force* eller *random debugging*. Som de skriver: »Novices often resort to random or brute-force changes to their code until the error seems to disappear« (s. 78). Denne form for usystematisk strategi er kendetegnet ved, at eleven ikke tester en specifik antagelse, men snarere håber på tilfældigt at ramme en løsning. Anderledes forholder det sig, når eleverne kan redegøre for, hvorfor de ændrer i koden, og samtidig har en plan for, hvad næste skridt skal være. Denne tilgang svarer til det, McCauley et al. (2008) beskriver som mere succesfulde debugging-strategier, hvor systematisk hypoteseafprøvning står centralt. Med andre ord er evnen til at forklare og planlægge sine ændringer et vigtigt skel mellem usystematisk gætteri og kvalificeret, hypotese-baseret fejlfinding.

I et forløb i 4. klasse, arbejder elever med at lave parafraser af Piet Mondrians værker, der udelukkende består af forskelligformede firkanter i forskellige farver med sort mellemrum.



Billede 12: Piet Mondrian, kilde: Wikipedia.

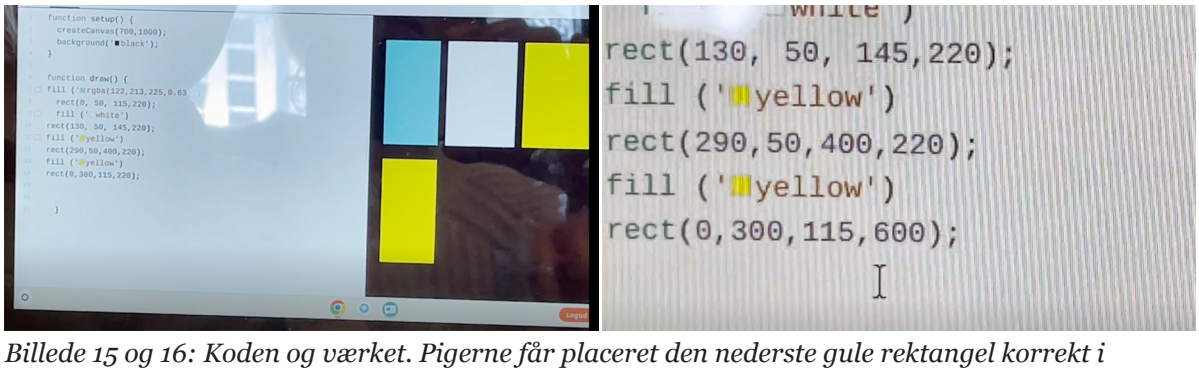
Eleverne skal kende koden for et rektangel og kunne kode rektanglernes placering korrekt i forhold til hinanden. Det at placere sine forskelligformede firkanter korrekt kræver for nogle elever mange forsøg med at ændre i koden. To piger fortæller, at de begyndte med at sætte alle deres øverste firkanter i værket til 100 på den lodrette akse på lærredet. Pigerne ved, at deres firkanter dermed vil ligge på samme vandrette linje i værket. Denne strategi viser, at de arbejder bevidst ud fra en konceptuel viden om, at det digitale lærred består af en matrix af punkter jf. afsnit om grundlæggende koncepter højere oppe.



Billede 13: To pigers Mondrian parafrase

I en anden gruppe er to piger ved at kode deres skitse af en Mondrian parafrase. De har først tegnet og farvet det på papir. Nu sidder pigerne sammen og skal til at kode et rektangel i OpenProcessing. På spørgsmålet om de vil forklare, hvad de gør på skærmen, siger den ene pige, at hun ikke ved, hvordan hun skal forklare det. Hun forsøger dog og fortæller, at de begynder med at indsætte farvekoden til det rektangel, de skal til at kode, og at denne farvekode skal indsættes over koden til selve koden for rektanglet, for ellers bliver rektanglet ikke farvet. Herefter forklarer begge piger, at de begynder på tallene. Det første punkt skal være 0 og så 50 siger den ene pige, men den anden peger på rektanglet over og siger »prøv at se, hvor langt nede den er«, og den første pige forstår, at hendes makker dermed mener, at koden skal tænkes i forhold til koden til den firkant, som den første pige har peget på. Hun retter det derfor til 300. De ser resultatet og er enige om, at rektanglet skal være længere. De retter det sidste tal i koden til 400, ser resultatet og retter det en sidste gang til 600. Nu er de tilfredse. Rektanglet er placeret lige under samme x punkt som rektanglet over og med den rette størrelse ned mod bunden af lærredet. Pigerne retter i koden to gange for at få det digitale Mondrian værk til at ligne deres skitse. Eksemplet viser, at pigernes samarbejde kvalificerer *debugging*-processen, fordi de understøtter hinanden i at kvalificere koden yderligere. Eksemplet viser også, at pigerne er bevidste om, hvad kodens enkelte dele gør, altså delenes semantiske betydning, og at deres debugging praksis er fagligt funderet.

Billede 14: En af pigerne i gang med at farvelægge en Mondrian skitse.



Billede 15 og 16: Koden og værket. Pigerne får placeret den nederste gule rektangel korrekt i forhold til deres ide.

Opsummering

Elevernes computationelle praksisser viser sig særligt i arbejdet med abstraktion, dekomposition og debugging, hvor de bevæger sig fra idé til kode. I de åbne opgaver udfordres eleverne af at oversætte deres æstetiske intentioner til koder. Når eleverne justerer deres idéer ud fra tekniske begrænsninger, kan dette også være en meningsfuld forhandling. I debugging-processerne ses forskellen mellem tilfældig og systematisk fejlfinding. Samarbejde og dialog understøtter elevernes debugging-processer, idet eleverne sammen kvalificerer og forstår kodens logik.

Computationelle perspektiver – computational empowerment

I følgende analytiske afsnit er fokus særligt rettet mod det Brennan og Resnick (2012) beskriver computationelle perspektiver. Som tidligere beskrevet handler det om, hvordan børn udvikler måder at se sig selv og verden gennem teknologisk skaben – herunder at udtrykke sig, skabe forbindelser og stille kritiske spørgsmål. Paperts begreb (1993) om computational empowerment samt Katterfeldts et al. (2015) dannelsesorienterede tilgang til empowerment tilbyder begge indsigter i, hvordan elevernes arbejde med OpenProcessing kan ses som både teknisk læring og personlig dannelse.

Handlekraft og kontrol over teknologien

Papert (1993) definerer som tidligere beskrevet computational empowerment som børns oplevelse af at kunne lære og skabe gennem teknologi på egne præmisser, hvilket svarer til Paperts idé om mathetics, og at teknologien kan fungere som katalysator for børns iboende læringspotentiale. Når eleven, der laver linjer , udtaler, at det er: »(…) mega fedt, for det kan blive PRÆCIS som jeg vil have det«, afspejler det Paperts forståelse af empowerment som en oplevelse hos eleven af handlekraft og kontrol, der opstår gennem engagementet med at skabe gennem teknologi (Papert, 1993).

Hvor Papert især fremhæver den individuelle læringsproces, lægger Katterfeldt et al. (2015) vægt på empowerment som et dannelsesorienteret fænomen. Som tidligere fremhævet definerer de empowerment som børns mulighed for at materialisere egne ideer gennem teknologi og dermed erfare self-efficacy, en følelse af kontrol og mestring i mødet med det digitale medium. Dette perspektiv genfindes i snefnug- og blomsterprojekterne, hvor eleverne udtrykker sig æstetisk gennem variationer i farver, rotationer og former. Ifølge Katterfeldt et al. (2015) opstår empowerment, når børn ikke blot lærer tekniske færdigheder, men oplever sig selv som producenter af meningsfulde digitale artefakter.

Forhandlinger og samspil

Samtidig ser vi, at empowerment også indebærer en forhandling mellem intention og teknologiens affordances. Eksemplet med eleven, der ønskede at kode en muslingeskal, men i stedet skabte en stjerne, illustrerer dette. Selvom resultatet ikke svarede til hendes oprindelige idé, oplevede hun stolthed over det opnåede produkt. Katterfeldt et al. (2015) anskuer ikke nødvendigvis dannelse gennem teknologi som noget, der handler om at realisere det oprindeligt planlagte, men at det handler om at udvikle indsigt i egne muligheder og begrænsninger samt erfaringer med at udtrykke sig digitalt. Vi ser, at eleverne motiveres gennem arbejdet med det kreative og det at skulle skabe noget, som de selv designer, inden for forskellige rammer. Elevernes idéer kræver, at de lærer kodekoncepter, der kan løfte idéerne. På den måde kan man sige, at det er i samspillet mellem kodning og design, at deres kreative, tekniske og udtryksmæssige potentiale udfolder sig, hvilket Unahalekhaha & Bers undersøger bl.a. i forhold til »purposefulness« (2022).

Opsummering

Projektet *Algoritmisk kunst* kan forstås som et empirisk eksempel på computational empowerment, hvor elever udvikler en begyndende handlekraft, kreativitet og refleksion i mødet med digital teknologi.

Diskussion

Når eleverne arbejder med kreative digitale processer i dette projekt, ser vi kun computational empowerment som en begyndende fornemmelse af at kunne påvirke og forme teknologi snarere end blot at bruge den. Dette stemmer overens med Tanggaards (2010) pointe om, at kreativitet og mestring tager tid, fordi læring og fornyelse vokser ud af kontinuitet snarere end brud.

I arbejdet med algoritmisk kunst bliver elevernes møde med materialet, som her er koden, algoritmen og de digitale værktøjer, en form for samskabelse, hvor redskabet også »…gør noget ved« den lærende (Tanggaard, 2010, s. 36). I denne proces bliver teknologien både et udtryksmiddel og en medspiller, der stiller krav, tilbyder modstand og åbner muligheder. Det er netop i denne vekselvirkning mellem styring og uforudsigelighed, at eleverne begynder at udvikle et mere reflekteret forhold til teknologi.

Den kreative proces kan derfor forstås som en gradvis tilblivelse, hvor eleverne langsomt lærer at læse, afkode og forme deres digitale værker på måder, der åbner for refleksion og kritisk tænkning. At udvikle kritisk tænkning indebærer her, som Tanggaard beskriver, evnen til at træde uden for sin praksis og genkende, hvordan egne kompetencer kan anvendes i nye sammenhænge.

De situationer, hvor eleverne eksperimenterer, justerer og forhandler mellem idé og teknisk mulighed, viser, hvordan de begynder at udvikle et element af computational empowerment. De lærer at »tale tilbage« til teknologien, og at forstå hvordan den fungerer, og hvordan de kan bruge den til at skabe udtryk, der opleves som deres egne.

Det vi ser er ikke fuld realiseret empowerment i projektet, men de første spor af, at børnene begynder at tage magten over deres teknologiske og æstetiske udtryk. Med fortsat arbejde og gentagne erfaringer med kodning som skabende praksis kan dette udvikle sig til egentlig computational empowerment, hvor eleverne ikke kun mestrer teknologien, men også bruger den til at udtrykke sig, udfordre normer og skabe nye forståelser af verden.

Konklusion

Artiklen har vist, hvordan børn gennem kreativ kodning lærer at mestre computationelle koncepter og udvikler computationelle praksisser. I forhold til forskningsspørgsmålet kan vi konkludere, at computationel tankegang manifesterer sig både i elevernes kode og i deres refleksioner over egne processer.

De computationelle koncepter, som rækkefølge, syntaks og konstruktion af simple former, er fundamentale for at kunne skabe selv de mest enkle digitale udtryk. Eleverne kan komme langt ved at kopiere og tilpasse koder, så længe de forstår, hvilke parametre, de kan manipulere. I takt med at deres ambitioner vokser, opstår der behov for en dybere forståelse af funktioner, løkker, operatorer og variable.

Den kontinuerlige vekselvirkning mellem instruktion og output, hvor eleverne eksperimenterer, ændrer og evaluerer, gør koncepterne håndgribelige og styrker forståelsen af deres

virkemåde. Kodekoncepter bør ikke blot *læres i brug*, men bruges mange gange i varierede situationer for at fæstne sig.

Elevernes computationelle praksisser er præget af trial and error-strategier, hvor »at prøve sig frem« fungerer som en grundlæggende tilgang. Hos nogle elever er denne strategi fagligt kvalificeret og ledsaget af sproglig refleksion, mens den hos andre fremstår mere intuitiv og tavs. I forløb, hvor eleverne selv skal udvikle idéer, bliver abstraktion og dekomposition afgørende CT-praksisser. Det, at kunne udvælge det væsentligste i en idé og nedbryde den i mindre dele kræver både forståelse af teknologiens muligheder og indsigt i egne evner som koder.

Vi ser computationelle perspektiver i elevernes kreative kodningspraksisser, værker og refleksioner, men de kredser primært om forhandlinger mellem idé og teknologisk kunnen snarere end om kritiske spørgsmål til teknologiens rolle i deres eget liv eller i samfundet. Det er forståeligt, da både elever og lærere har været nybegyndere i arbejdet med algoritmisk kunst.

Disse første erfaringer udgør imidlertid et solidt grundlag for fremtidige forløb, hvor fokus kan flyttes fra teknisk forståelse til mere kritiske og analytiske dimensioner af computationel tankegang. Når eleverne næste gang arbejder med algoritmisk kunst gennem kreativ kodning, kan de bygge videre på deres erfaringer og bevæge sig fra at forstå og anvende teknologi til at udfordre og forme den.

Referencer

Amiel, T., & Reeves, T. C. (2008). Design-based research and educational technology: Rethinking technology and the research agenda. *Journal of Educational Technology & Society, 11*(4), 29–40.

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction, 31*, 100328. https://doi.org/10.1016/j.ijcci.2021.100329

Bandura, A. (1997). *Self-efficacy: The exercise of control*. W. H. Freeman.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association* (pp. 1-25).

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences, 2*(2), 141-178.

Børne- og Undervisningsministeriet. (2018). *Teknologiforståelse i folkeskolen – faglighed og dannelse*. https://www.emu.dk/sites/default/files/2020-10/Teknologiforstaelse_folkeskolen.pdf

Dindler, C., Smith, R. C., & Iversen, O. S. (2020). Computational empowerment: Participatory design in education. *CoDesign, 16*(1), 66-80. https://doi.org/10.1080/15710882.2020.1722173

Greenberg, I. (2007). *Processing: Creative coding and computational art*. Apress. https://doi.org/10.1007/978-1-4302-0310-0

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Haseski, H. İ., İlic, U., & Tugtekin, U. (2018). Defining a new 21st century skill – computational thinking: Concepts and trends. *International Education Studies, 11*(4), 29-42. https://doi.org/10.5539/ies.v11n4p29

Højholdt, A. (2017). *Co-teaching – samarbejde om undervisning*. Hans Reitzels Forlag.

Iversen, O. S., Smith, R. C., & Dindler, C. (2018). From computational thinking to computational empowerment: A 21st century PD agenda. In *Proceedings of the 15th Participatory Design Conference – Full Papers, Volume 1 (PDC '18)* (pp. 1-11). ACM. https://doi.org/10.1145/3210586.3210592

Kafai, Y. B., & Peppler, K. A. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. *Review of Research in Education, 35*(1), 89-119. https://doi.org/10.3102/0091732X10383211

Katterfeldt, E. S., Dittert, N., & Schelhowe, H. (2015). Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops. *International Journal of Child-Computer Interaction, 5*, 3-10. https://doi.org/10.1016/j.ijcci.2015.08.001

Kim, C., Vasconcelos, L., & Belland, B. R., et al. (2022). Debugging behaviors of early childhood teacher candidates with or without scaffolding. *International Journal of Educational Technology in Higher Education, 19*(1), 26. https://doi.org/10.1186/s41239-022-00319-9

Kinnula, M., Iivari, N., Molin-Juustila, T., Keskitalo, E., Leinonen, T., Mansikkamäki, E., Käkelä, T., & Similä, M. (2017). Cooperation, combat, or competence building: What do we mean when we are 'empowering children' in and through digital technology design? *ICIS 2017 Proceedings*, Paper 15. https://aisel.aisnet.org/icis2017/TransformingSociety/Presentations/15

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. *Computer Science Education, 18*(2), 67-92. https://doi.org/10.1080/08993400802114581

Merriam, S. B. (1998). *Qualitative research and case study applications in education*. Jossey-Bass.

Møller, H., Jensen, M. E., Schrøder, V., & Gregersen, A. S. (2021). *Kvalitative undersøgelser i læreruddannelsens BA-projekt: Inspiration fra praksisnær skoleforskning*. Samfundslitteratur.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. Basic Books.

Petropouleas, E. (2024). *Algoritmisk kunst*. https://cfu.phabsalon.dk/algoritmisk-kunst-0

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

Tanggaard, L. (2010). Kreativitetens materialitet. *Nordiske Udkast, 38*(1), 7-26. https://doi.org/10.7146/nu.v38i1.133855

Unahalekhaka, A., & Bers, M. U. (2022). Evaluating young children's creative coding: Rubric development and testing for ScratchJr projects. *Education and Information Technologies, 27*(5), 6577–6597. https://doi.org/10.1007/s10639-021-10873-w

Van Mechelen, M., Musaeus, L. H., Iversen, O. S., Dindler, C., & Hjorth, A. (2021). A systematic review of empowerment in child-computer interaction research. In *Proceedings of the Interaction Design and Children Conference (IDC '21)* (pp. 1-12). ACM. https://doi.org/10.1145/3459990.3460701

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.
Woo, K., & Falloon, G. (2022). Problem solved, but how? An exploratory study into students' problem-solving processes in creative coding tasks. *Thinking Skills and Creativity, 46*, 101193. https://doi.org/10.1016/j.tsc.2022.101193

Biografier

Thilde Emilie Møller, ph.d., er lektor ved Professionshøjskolen Absalon, Center for Skole og Læring. Hun arbejder med forskning og udvikling inden for teknologiforståelse som en ny faglighed i skolen og underviser på læreruddannelsen i teknologiforståelse. Hun er særligt optaget af at forstå børns digitale liv – deres interesser, engagement og måder at udtrykke sig og skabe mening med digital teknologi – samt hvordan skolen kan inddrage og bygge videre på disse erfaringer i undervisningen.

Mie Buhl er professor ved Institut for Kommunikation og Psykologi. Hun er leder af forskningscentret Visual Studies and Learning Design (ViLD) og medstifter af den nordiske masteruddannelse i visuelle studier og kunstpædagogik (NoVA), som er et samarbejde mellem Aalborg Universitet og Aalto Universitet. Hun har en baggrund som folkeskolelærer med fokus på billedkunst, læreuddanner i billedkunst, samt kandidatudannelsen i billedpædagogik. Hendes forskning fokuserer på visuel læring og videndannelse, især i relation til digital teknologier i visuelle og sociale praksisser.

Pernille Lomholt, Cand. Mag, Master i IKT og Læring, er lektor på Professionshøjskolen Absalon, Center for skole og læring og underviser i Efter- og Videreuddannelsen i moduler, der kredser om teknologiforståelse, digital understøttelse og didaktisk design i eksperimenterende læringsmiljøer. Pernille er, når hun deltager i udviklings- og forskningsprojekter, optaget af forskellige deltagelsesmuligheder og -former i undervisningen og særligt i faget eller fagligheden teknologiforståelse.

Tomas Højgaard er lektor i matematikkens didaktik ved DPU, Aarhus Universitet. Hans forskning handler for en stor dels vedkommende om at undersøge muligheder og vanskeligheder ved at bruge faglige kompetencebeskrivelser som didaktisk udviklingsværktøj. Herudover forsker han i vilkår for og vanskeligheder ved at etablere et gensidigt frugtbart samspil mellem forskningsfeltet matematikkens didaktik og matematikundervisningens praksis.

Eva Petropouleas, Cand.Mag, er pædagogisk konsulent for teknologiforståelse ved CFU, Professionshøjskolen Absalon og arbejder i forskellige sammenhænge med udvikling af teknologiforståelsesfagligheden, bl.a. gennem nationale projekter, kompetenceforløb og udformning af undervisningsressourcer. Eva er særligt optaget af, hvordan programmering og computationel tankegang kan understøtte børn og unge i at være konstruktivt skabende med digital teknologi i praktisk-/musiske kontekster.