Hovedrapport: Coding Class

Dokumentation og evaluering

PROFESSIONSHØJSKOLEN METROPOL

AD NYE VEJE AALBORG UNIVERSITET

IT-Branchen

INDHOLDSFORTEGNELSE:

Indledning

Denne rapport rummer evaluering og dokumentation af Coding Class projektet[1]. Coding Class projektet blev igangsat i skoleåret 2016/2017 af IT-Branchen i samarbejde med en række medlemsvirksomheder, Københavns kommune, Vejle Kommune, Styrelsen for IT- og Læring (STIL) og den frivillige forening Coding Pirates[2]. Rapporten er forfattet af Docent i digitale læringsressourcer og forskningskoordinator for forsknings- og udviklingsmiljøet Digitalisering i Skolen (DiS), Mikala Hansbøl, fra Institut for Skole og Læring ved Professionshøjskolen Metropol; og Lektor i læringsteknologi, interaktionsdesign, design tænkning og design-pædagogik, Stine Ejsing-Duun fra Forskningslab: It og Læringsdesign (ILD-LAB) ved Institut for kommunikation og psykologi, Aalborg Universitet i København. Vi har fulgt og gennemført evaluering og dokumentation af Coding Class projektet i perioden november 2016 til maj 2017.

Coding Class projektet er et pilotprojekt, hvor en række skoler i København og Vejle kommuner har igangsat undervisningsaktiviteter med fokus på kodning og programmering i skolen. Evalueringen og dokumentationen af projektet omfatter kvalitative nedslag i udvalgte undervisningsinterventioner i efteråret 2016 og foråret 2017. Undervisningsinterventionerne har haft flere formål:

1. at fremme elevernes forståelse for den digitaliserede verden, der omgiver dem nu og i fremtiden
2. at fremme elevernes interesse for it
3. at igangsætte undervisning, der fremmer elevernes evne til selv at arbejde mere kreativt og skabende med it i grundskolen
4. at sætte fokus på it, kodning og computationel tænkning som vidensdomæne i grundskolen

Projektet har taget afsæt i Coding Pirates' erfaringer med at skabe uformelle læringsrum for børn og unge via klubaftener, hvor fokus er på aktiviteter, der fremmer børn og unges skabende og kreative it-kompetencer.

Coding Class projektet repræsenterer således et forsøg på at overføre erfaringerne med de uformelle læringsrum fra Coding Pirates til skolernes mere formelle læringsrum. Igangsættelse af undervisningsinterventionerne i skolerne har været faciliteret af fire Coding Class instruktører fra Coding Pirates (to i København og to i Vejle). Idéen har været at skabe en bevægelse fra undervisningsaktiviteter primært faciliteret af Coding Class instruktørerne (i starten) hen imod gradvist, at undervisningsaktiviteterne skulle kunne overtages og føres videre af de deltagende lærere og skoler. Fælles[3] afsæt har været programmering i Scratch og udvikling af spil. Undervisningsinterventionerne har som hovedregel taget afsæt i, at eleverne i grupper har oprettet egne virksomheder, hvor de har udviklet et spil. Først har

[1] Coding Class projektets hjemmeside: https://itb.dk/articles/fremtidens-kompetencer/hvad-er-coding-class
[2] Coding Pirates hjemmeside: https://codingpirates.dk/
[3] Skolerne og klasserne har imidlertid haft meget forskellige afsæt for at arbejde med Coding Class aktiviteterne, og de har valgt meget forskellige tilgange til arbejdet med Coding Class projektet og undervisningsinterventionerne. Det vil vi vende tilbage til.

eleverne arbejdet med instruktionsvideoer og instruktionsforløb i Scratch, og dernæst har de arbejdet med egne spilproduktioner. De deltagende klasser har haft fire sessioner med Coding Class instruktørerne, og mange klasser har haft et afsluttende møde med en it-virksomhed eller anden ekstern aktør, som i de fleste tilfælde har vurderet og givet eleverne respons på deres arbejde med spiludviklingen. Imellem sessionerne med Coding Class instruktørerne, har klasserne selv kunnet prioritere at arbejde med Coding Class aktiviteter. Projektet havde fælles opstartsdage i både København og Vejle[4], hvor lærerne mødte hinanden og blev introduceret til Scratch via en instruktionsvideo og eget arbejde med en opgave i Scratch. Både i Vejle og København er lærerne endvidere bragt sammen i netværk via sociale medier (Slack i København og Facebook i Vejle).

Coding Class projektet er igangsat af IT-Branchen med den meget eksplicitte politiske dagsorden, at skubbe aktivt til arbejdet med it som fag og med computationel tænkning[5] som vidensdomæne i grundskolen. Denne dagsorden har internationalt fyldt i en årrække, hvor en række lande (fx USA, Bulgarien, Cypern, Tjekkiet, Estland, Grækenland, Irland, Italien, Litauen, Polen, Portugal, Sverige og UK) allerede har sat informatik, computationel tænkning, kodning og programmering på skoleskemaet. I UK har computing fx været en del af det nationale curriculum siden 2013 (Caspersen, 2017). Denne dagsorden har ikke hidtil fyldt tilsvarende mærkbart i Danmark.

Når IT-Branchen reagerer nu, så skal det blandt andet ses i lyset af den mangel på it-kompetencer, der blev konstateret i regeringens kortlægning af "Virksomhedernes behov for digitale kompetencer", maj 2016[6]), der blandt andet peger på, at Danmark vil mangle ca. 19.000 it-specialister i 2030. Ifølge Danmarks Vækstråd (2016) vil Danmark, ud over it-specialister, i det hele taget mangle medarbejdere med både almene it-kompetencer og avancerede it-kompetencer. Fremtidens jobmarked spås at være sammensat af langt flere stillinger, hvor den primære jobfunktion ikke er it, men avancerede it-kompetencer alligevel ses som væsentlige for at kunne udføre arbejdet. Funderet i den stigende digitalisering af samfundet og den globale verden, forudser flere internationale rapporter, at der vil udvikle sig et stort "digitalt gab" mellem de borgere, der har kundskaberne til at deltage i et højteknologisk samfund, og dem der ikke har kundskaberne. Med reference til en række rapporter peger Danmarks Vækstråd således på et behov for, at ikke bare grundskolen, men hele uddannelsessystemet, kommer mere aktivt i gang med at uddanne børn og unge, der kan bidrage aktivt og forholde sig kritisk og medskabende til fremtidens samfundsdannelser.

Diskussionen om behovet for at indføre informatik som almendannende fag eller fagområde på alle uddannelsesniveauer er i dag ved at tage fat for alvor i Danmark[7], hvor blandt andet Digitalt Vækstpanel i en rapport (2017) anbefaler regeringen, at der sættes ind på alle

[4] De to kommuner har derudover valgt at understøtte Coding Class projektet på meget forskellige måder. Det vil vi vende tilbage til.
[5] Vores egen oversættelse af "computational thinking". Computational thinking og "computing" er betegnelser, der benyttes internationalt.
[6] Se VIRKSOMHEDERS BEHOV FOR DIGITALE KOMPETENCER, https://erhvervsstyrelsen.dk/sites/default/files/media/rapport_-_virksomheders_behov_efter_digitale_kompetencer.pdf

[7] Se fx http://www.altinget.dk/uddannelse/artikel.aspx?emne=4441

uddannelsesniveauer. Gymnasiereformen anno 2016 har allerede Informatik som nyt almendannende fag (Caspersen, 2017). Siddende undervisningsminister, Merete Riisager, har i foråret 2017 nedsat Rådgivningsgruppen for teknologi i undervisningen, der skal sparre med ministeren om en langsigtet plan for digital læring[8], og Undervisningsministeriet har nedsat en arbejdsgruppe, der i foråret 2017 har formuleret et etårigt forsøgs valgfag i udskolingen i grundskolen med fokus på informatik og elevernes teknologiforståelse[9].

It eksisterer ikke som selvstændigt fag i grundskolen, hvor "It og medier"[10] ligesom "Innovation og entreprenørskab"[11] er tværgående obligatoriske temaer for alle fag i hele grundskolen i dag. Coding Class projektet er undervisningsinterventioner, der kan knytte an til de tværgående temaer, men IT-Branchen refererer ikke til de tværgående temaer. Ifølge IT-Branchens hjemmeside[12] er fokus i Coding Class projektet på at:

> "*Kodning handler i høj grad om at forstå verden omkring os fra et nyt perspektiv. At forstå hvordan teknologien virker og påvirker os – og hvordan vi kan påvirke og ændre teknologien til at skabe helt nye løsninger*."

Ifølge hjemmesiden[13] er fokus i Coding Class blandt andet på "algoritmisk tænkning", "problemnedbrydning i praksis" og "mønstergenkendelse i hverdagen". Algoritmisk tænkning, problemnedbrydning og mønstergenkendelse er tre analytiske tænkemåder, der knytter an til det vidensdomæne og fagfelt, som internationalt betegnes som computationel tænkning.

IT-Branchen skriver på Coding Class projektets hjemmeside[14]:

> "*Ved at indføre kodning som et selvstændigt fag, er vi med til at udvikle børns evne til at analysere og skabe de rigtige løsningsstrategier til både små og komplekse problemer.*
>
> *At mangle en grundlæggende forståelse for det digitale fundament, vores fremtidige samfund bygges på, er et problem ikke bare for vores børn men for Danmark som helhed. Som samfund taber vi nemlig, hvis vi ikke lykkedes med mere end blot at skabe dygtige og konstante brugere af fremtidens løsninger.*
>
> *Vi skal lære de nye generationer at være problemløsende, skabende og innovative med teknologi. Derfor har vi brug for et kreativt og skabende it-fag i folkeskolen. Derfor har vi brug for at få kodning på skoleskemaet."*

Coding Class projektet tager altså afsæt i antagelsen om, at der er brug for et selvstændigt it-fag i skolen, hvor arbejdet med kodning står centralt.

---

[8] Se pressemeddelelse: http://www.uvm.dk/aktuelt/uvm/udd/folke/2017/mar/170330-undervisningsministeren-nedsaetter-raadgivningsgruppe-for-digital-laering
[9] Se resultatet her: http://www.emu.dk/modul/teknologiforst%C3%A5else-valgfag-fors%C3%B8g-%E2%80%93-f%C3%A6lles-m%C3%A5l-og-l%C3%A6seplan
[10] Se vejledningen til "It og medier": http://www.emu.dk/modul/it-og-medier-vejledning
[11] Se vejledningen til "Innovation og entreprenørskab": http://www.emu.dk/modul/innovation-og-entrepren%C3%B8rskab-vejledning-0
[12] Lokaliseret 28. maj 2017.
[13] Lokaliseret 28. maj 2017.
[14] Lokaliseret 28. maj 2017.

IT-Branchen refererer på Coding Class hjemmesiden til Wikipedia om ”Computational Thinking” og Center for Computational Thinking (Carnegie Mellon), men definerer i øvrigt ikke, hvad “computational thinking” er i Coding Class projektet. Der mangler en egentlig oversættelse af begrebet computational thinking til dansk. I det følgende afsnit, der beskriver, hvordan vi har valgt at gribe evalueringen og dokumentationen af projektet an, præsenterer vi vores tilgang til computationel tænkning.

Arbejdet med informatik, computationel tænkning, elevernes kreative it-kompetencer, teknologiforståelse og -interesser i grundskolen er væsentlige, men også meget komplekse og udfordrende dagsordener. Coding Class projektet indskriver sig i en højaktuel dagsorden, på et tidspunkt, hvor den først er ved at tage fart i Danmark. Vi håber at denne rapport vil bidrage til den fortsatte udvikling, nuancering og kvalificering af arbejdet i de kommende år.

Evaluering og dokumentation

I det følgende vil vi beskrive baggrunden for vores evaluering og dokumentation af Coding Class projektet. Det er her væsentligt at nævne, at der er tale om en relativt lille kvalitativt orienteret evaluering og dokumentation af Coding Class projektet. Endvidere er Coding Class projektet at betragte som et spædt initiativ, der i det hele taget har haft begrænset tid og ressourcer til rådighed. Coding Class projektet er et pilotprojekt, som har haft helt særlige vilkår[15], og derfor er det væsentligt, ifølge vores overbevisning, ikke at drage for stærke konklusioner på baggrund af Coding Class projekterfaringerne. IT-Branchen har søsat Coding Class projektet med afsæt i antagelsen om et behov for at indføre et it-fag med fokus på elevernes skabende og kreative it-kompetencer via arbejde med kodning og computationel tænkning. I praksis har Coding Class aktiviteterne dog etableret sig i forhold til eksisterende skolehverdage, hvor Coding Class aktiviteterne på nogle skoler har været knyttet til eksisterende fag og lektioner og har strukket sig over et halvt år, mens Coding Class på andre skoler har varet hele intensive dage henover et par måneder.

Evaluering og dokumentation af Coding Class projektet har til formål at dele kommune-, skole-, lærer-, og eleverfaringerne, erhvervet gennem Coding Class projektet, hvor de undervisningsaktiviteter vi har fulgt, har haft en intervenerende form. Der er altså tale om de første erfaringer og spæde forsøg, som lærere og elever har gjort sig. Der er ikke tale om, at de deltagende skoler har etableret et it-fag, men de har iværksat Coding Class aktiviteter i skolehverdagen, og eftersom der ikke findes etablerede praksisser, er disse observationer vigtige.

Evaluering og dokumentation har fokus på, hvordan de deltagende klasser og skoler har arbejdet med elevernes kreative og skabende it-kompetencer i Coding Class projektet. Omdrejningspunktet er at synliggøre de læringsmæssige og pædagogiske perspektiver i Coding Class aktiviteterne. Endvidere har evaluering og dokumentation til formål at udpege potentialer og udfordringer i arbejdet med forankring og udbredelse af udviklingen af

[15] Dem beskriver vi nærmere.

elevernes kreative og skabende it-kompetencer, som en del af hverdagspraksis på skoler i Danmark.

Evaluering og dokumentation af Coding Class projektet består af forskellige kvalitative nedslag i Coding Class aktiviteterne. Disse nedslag skal hovedsageligt ses i lyset af hvilke aktiviteter, der har været i gang og været tilgængelige for os i perioden fra november 2016 til maj 2017. Projektet var allerede i gang, da vi blev tilknyttet projektet, og flere Coding Class forløb var allerede mere eller mindre afsluttet. Undersøgelsens størrelse, den relativt korte tidshorisont, og det faktum, at Coding Class aktiviteterne på mange måder repræsenterer nye aktiviteter i undervisningen i grundskolen, betød, at vi prioriterede få kvalitative case studier i København og Vejle. Coding Class undervisningsaktiviteterne er landet i to meget forskellige kommuner og på skoler med meget forskellige lærer- og elevforudsætninger. For at understøtte de kvalitative casebaserede nedslag med et helikopterperspektiv på projektet og samtidig nuanceret blik for variationerne i aktiviteterne – og dermed også erfaringer, pædagogiske og læringsmæssige udfordringer og muligheder - har vi interviewet Coding Class instruktørerne i København og Vejle i efteråret 2016 og foråret 2017. I tilkøb til disse interview, har vi i foråret 2017 interviewet de to kommunale Coding Class koordinatorer, der har evalueret, koordineret og understøttet skolernes deltagelse i henholdsvis København og Vejle.

Interviewene og observationer har haft fokus på:

- *Coding Class projektets relation til skolens praksis (fag, læringsmål og it-kompetencer, betingelser for succes)*
- *Coding Class projektets pædagogik og bagvedliggende antagelser*
- *Deltagernes oplevelse af projektet (læringsudbytte, motivation, fokusområder, tærskler og potentialer)*
- *Deltagernes indgang til og håndtering af projektet (organisering, motivation, kompetenceudvikling)*
- *Coding Class projektets effekt på skolernes it-initiativer (forankring, videreudvikling, tærskler og potentialer)*
- *Virksomhedernes rolle i projektet set fra lærer og elev perspektiv*
- *Deltagernes anbefalinger til videreudvikling af undervisningsaktiviteter med fokus på kodning og programmering i folkeskolen*

I skoleåret 2016/2017 har syv folkeskoler fra København og tre fra Vejle deltaget i Coding Class projektet[16]. Casestudierne er foretaget via nedslag på to skoler i København og to skoler i Vejle. I rapporten refererer vi til kommunerne, men har valgt at anonymisere på skole, elev- og lærerniveau. Ligeledes refererer vi til Coding Class instruktørerne og de kommunale Coding Class koordinatorer. Casestudierne omfatter interview med én lærer fra en skole i København, der havde afsluttet deres forløb, og observationer på én skole i København og to skoler i Vejle, på forskellige tidspunkter i Coding Class forløbene. Udover interview med de lærere, der har været tilknyttet Coding Class aktiviteterne på skolerne, har

[16] I København har to specialskoler endvidere deltaget. En enkelt af skolerne i København har haft fokus på Coding Class som valgfag. Evaluering og dokumentation har afgrænset sig til at fokusere på Coding Class i almenskoler og som aktivitet for alle elever. Coding Class aktiviteterne er foregået i 4.-9. klasse. Nogle skoler har deltaget med én klasse, mens andre har valgt en hel årgang med fx tre 6. klasser.

vi gennemført gruppeinterview med elever fra hvert observeret forløb. I alt har vi interviewet tre lærere og 11 elever fordelt på tre grupper (fem piger, tre drenge, to piger og en dreng) fra to skoler i Vejle, og to lærere og fem elever fordelt på to grupper (en pige og en dreng, to drenge og en pige) fra København. Vi har endvidere indsamlet eksempler på elevproduktioner, elevernes refleksioner undervejs i forløbet gennem uformelle samtaler med eleverne og skriftligt, samt lærernes evaluering af Coding Class projektet, forestået af Københavns Kommunes Coding Class koordinator.

Computationel tænkning – hvad handler det om?

Som allerede nævnt, knytter Coding Class projektet ikke i udgangspunktet an til en bestemt forståelse af, hvad det vil sige at arbejde kreativt og skabende med it, kodning og computationel tænkning. Coding Class projektet har haft en projekthjemmeside, men ikke en egentlig projektbeskrivelse, idet projektet blev etableret meget hurtigt i et åbent og co-creation forløb i samspil mellem IT-Branchen, Coding Pirates, STIL og de involverede kommuner. Fra ide til projektmidlerne var etableret gik der således 14 dage. Coding Class aktiviteterne har været løst defineret med fokus på spiludvikling, kodning og programmering via Scratch, og overførsel af Coding Pirates erfaringerne til skolen. Det har hverken af Coding Class instruktørerne eller af IT-Branchen, kommunerne eller skolerne været tydeligt ekspliciteret, hvad der har været de anvendte pædagogisk-didaktiske greb og hvilke læringsmål, der har været i fokus i Coding Class projektet. Vi har heller ikke fundet skriftlige beskrivelser af, hvad det vil sige, at eleverne arbejder *kreativt* og *skabende med it* i Coding Class projektet. Vores afsæt for at vurdere og dokumentere Coding Class aktiviteterne har derfor først og fremmest været gennem vores adgang til empiriske iagttagelser og beskrivelser af Coding Class aktiviteterne via interview og observationer.

For at kunne vurdere, hvordan Coding Class aktiviteterne har knyttet an til elevernes kreative og skabende it-kompetencer, kodning og computationel tænkning, og hvilke læringsmæssige og pædagogiske gevinster, der knytter sig til Coding Class aktiviteter, har vi derfor haft behov for at ridse scenen op og tydeliggøre vores ståsted for at dokumentere og evaluere Coding Class aktiviteterne samt bidrage til diskussionen af kompetencefeltet. I det følgende vil vi gøre dette med reference til international litteratur og eksisterende viden om arbejdet med computationel tænkning i grundskolen. Valget af dette skal ses i lyset af, at IT-Branchen selv fremhæver computationel tænkning på Coding Class projektets hjemmeside.

Informatik og computationel tænkning

I en mindre, men meget internationalt citeret og dagsordensættende (”viewpoint”) artikel fra 2006, formulerer Jeanette M. Wing sit synspunkt på, hvad computationel tænkning er og bør handle om:

> *”Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers?*

And What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions. Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability.

…Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science. Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it?... Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation. Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code… It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance. Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable…

Computer science is the study of computation— what can be computed and how to compute it…. Computer science is not computer programming. Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction…"

Af Wings tilgang fremgår det, at computational tænkning har fokus på at relatere analytiske tænkemåder fra vidensdomænet datalogi/informatik ("computer science") til andre vidensdomæner og fagområder. Wing præsenterer hermed en bevægelse fra et fokus på datalogi og informatik som noget nogle få interesserede arbejder med, til et fokus på datalogi og informatik som et område *alle* kommer til at arbejde med. Wings argument er derfor, at alle skal kende til disse analytiske tænkemåder - alle skal lære at tænke som en computer scientist. Computationel tænkning er altså i Wings terminologi en problemløsende tilgang til verden og dens computerrelaterede problemstillinger. I en artikel fra 2008 beskriver Wing computationel tænkning som et tværdisciplinært område:

"Computational thinking is a kind of analytical thinking. It shares with mathematical thinking in the general ways in which we might approach solving a problem. It shares with engineering thinking in the general ways in which we might approach designing and evaluating a large, complex system that operates within the constraints of the real world. It shares with scientific thinking in the general ways in which we might approach understanding computability, intelligence, the mind and human behaviour."

(Wing, 2008, s. 3717)

Wing betoner her forbindelsen mellem computationel tænkning og fagene inden for Science Technology Engineering and Mathematics (STEM). Computationel tænkning og forståelse er ikke et nyt område, men det er et område, der har været meget lidt fokus på i uddannelsessystemerne internationalt i en årrække. Dette ses blandt andet repræsenteret i Danmark ved, at vi i dag ikke har it eller informatik som selvstændigt fag i grundskolen.

Der eksisterer forskellige tilgange til undervisning med fokus på computationel tænkning, ligesom, at der eksisterer en mangfoldighed af begrebsudfoldelser af vidensdomænet computationel tænkning og domænets underområder (Committee for the Workshops on Computational Thinking, 2011). Caspersen (2017) fremhæver, at argumentet for, at informatik og computationel tænkning skal være en del af almendannelsen (i gymnasiet), hentes i betydningen af udbredelsen af "informatikkens grundlæggende principper, tænkemåder, udtryksformer og arbejdsformer" (s. 1). Forfatteren fremhæver, at informatik kan ses som:

- Det 21. århundredes mikroskop, der giver os radikalt nye erkendelsesmuligheder
- Det 21. århundredes udtryksværktøj, hvorigennem vi kan skabe helt nye udtryk,
- Et domæne, der kollapser etablerede grænser (fx fysiske, geografiske, tidslige, politiske), og giver radikalt nye sociale og fællesskabsmuligheder.

Ifølge Caspersen er informatik et selvstændigt videnskabsområde, der har eksisteret i mere end 80 år, og det betragtes af nogle som et videnskabsdomæne på linje med naturvidenskab, humaniora og samfundsvidenskab.

Arbejdet med at udvikle elevernes computationelle tænkning hænger sammen med det perspektiv, aktørerne, der udfører arbejdet, anlægger på computationel tænkning, pædagogik og læring samt vægtning af de forskellige relaterede underbegreber (fx dekomposition, algoritmisk tænkning). Kafai (2016) bygger videre på Wing, men pointerer, at der med det aktuelle fokus på computationel tænkning ikke bare er tale om en gentagelse af 80'ernes edb (elektroniske databehandling) undervisning. Kafai udvider Wings forståelse og gør computationel tænkning endnu mere aktuel med sin formulering af et "social turn i K-12[17] computing", hvor hun argumenterer for et:

> *"…greater focus on underlying social and cultural dimensions of programming. We should rethink what and how students learn to become full participants in networked communities. Students need strategies to cope with the vulnerability of sharing one's work for others to comment on and remix."*

Ifølge Kafai er det væsentligt at have blik for hvilke strategier, der er styrende for arbejdet med computational tænkning i skolen, idet fx Wings tilgang ikke nødvendigvis inkluderer de sociale og kulturelle dimensioner af programmering. Med afsæt i Kafai er det væsentligt at have blik for, at arbejdet med programmering og kodning også handler om at kunne blive fuldgyldig deltager i netværksbaserede lærende fællesskaber:

> "*To learn to code students must learn the technicalities of programming language and common algorithms, and the social practices of programming communities… Education activist Paulo Freire once said that "reading the word is reading the world." He was right. Today, reading code is about reading the world. It is needed to understand, change, and remake the digital world in which we live*."
>
> (Kafai, 2016, s. 27)

[17] K-12 er forkortelsen for grundskole og ungdomsuddannelse i USA.

Fra Wing's formulering i 2006 til Kafai's formulering af computationel tænkning i 2016 sker der en bevægelse hen imod mere deltagende computationel tænkning, der understreger de sociale og fællesskabende aktiviteter, og hvor en væsentlig pointe er, at arbejdet med computationel tænkning også handler om at kunne samskabe med verden som platform.

Brennan og Resnick (2012) knytter ligeledes an til en mere deltagende tilgang til computationel tænkning, idet de vægter design-baserede og kreative læreprocesser. Brennan, Resnick og Kafai referer til en tilgang, der udover STEM fagene inkluderer de kreative og humanistiske fag ('arts') - Science, Technology, Engineering, Arts and Mathematics (STEAM). Med STEAM udvides fokus fra en teknisk, naturvidenskabelig og samfundsorienteret tilgang til også at inkludere et designperspektiv og de musisk-kreative og humanistiske fag.

Angeli et al. (2016) fremhæver, at et væsentligt aspekt af arbejdet med computationel tænkning er at designe et autentisk curriculum, hvor fokus er på at arbejde med meningsfulde og reelle problemer i verden. Computationel tænkning er dermed fortsat en tilgang til at håndtere problemer i verden. I tråd med dette har Alfred Aho fremhævet fem forskellige udbredte argumenter for at inkludere computationel tænkning i skolens curriculum (Committee for the Workshops on Computational Thinking, 2011, s. 36-37):

1. Computationel tænkning har indflydelse på stort set ethvert menneskeligt aspekt gennem anvendelse indenfor mange fagområder fx jura, lægevidenskab, arkeologi, journalisme og biologi.

2. Der er store samfundsmæssige og menneskelige risici forbundet ved dårlig computationel tænkning fx softwarefejl. I en verden fyldt med modelleringer og simulationer har vi brug for god software.

3. Med computationel tænkning kan vi udvikle nye og forbedrede måder at skabe, forstå og manipulere repræsentationer. Repræsentationer kan dramatisk ændre på den måde, vi forstår og ser problemer på.

4. Arbejdet med kreative programmeringsprojekter som afsæt for at udvikle computationel tænkning kan motivere elever til at søge videre uddannelse og arbejde med fokus på informatik / datalogi.

5. Relationer mellem informatik og uddannelser kan aktivere undervisning i nye områder, og understøtte arbejdet med nye metoder og nye idéer til problemløsning, der matcher videnspraksisserne, i det samfund som børn og unge skal deltage i.

Der findes mange forskellige mulige tilgange til arbejdet med computationel tænkning i grundskolen - også knyttet til fag (ibid.). Hver tilgang hænger sammen med fx vægtningen af curriculær/ekstracurriculær læring, måder at se og forstå læring på, måder at handle og opfatte kapaciteter på, anvendelse af undersøgelsesmetoder (fx eksperimentel laboratorium versus feltbaseret læring), tilgængelig tid og fokus på samarbejde.

Coding Class projektet har haft fokus på Coding Class som en selvstændig aktivitet, der både skulle anspore elevernes interesse for it og computationel tænkning og åbne elevernes

forståelse for betydningen af digital teknologi for samfundsdannelser og egne kreativt skabende handlemuligheder. Med Scratch som afsæt har projektet prioriteret et redskab, der er udviklet med børn som målgruppe. I Scratch er mange instruktionsvideoer og tidligere projekter tilgængelige for eleverne. Disse materialer er overvejende relateret til spilkulturen. Coding Class har primært fokuseret på spiludvikling, hvilket giver mulighed for at trække på elevernes egen erfaringshorisont, da digitale spil er noget langt de fleste børn kan relatere til, har erfaringer med og bruger meget tid på. Havde projektet fx havde valgt et programmeringsredskab målrettet et professionelt fællesskab og ansporet eleverne til at udvikle andre typer programmer, så ville elevernes læringsudbytte, repertoire og motivation have været en anden.

Computationel tænkning pædagogisk-didaktisk ifølge forskning

Weinberg (2013) peger i sin gennemgang af publikationer om computationel tænkning i perioden 2006-2011 på, at computationel tænkning stadigvæk er et underudviklet område pædagogisk-didaktisk og forskningsmæssigt. Ifølge Weinberg er forståelsen af computationel tænkning præget af brede termer, der er svære at operationalisere. Endvidere foreligger der ifølge Weinberg (2013) ikke dokumentation af aktiviteter og elevers læring, og det er stadigvæk et meget åbent spørgsmål, hvordan man lærer og – ikke mindst – evaluerer computationel tænkning i grundskolen.

Weinberg (2013, s. 18) refererer til to forskellige definitioner af computationel tænkning:

> ”*Two definitions for computational thinking stand out at this point. The first is offered by the CSTA [Computer Science Teacher Association - vores tilføjelse],who defines computational thinking as a problem solving process that includes (a) formulating problems in a way that enables us to use a computer and other tools to help solve them, (b) logically organizing and analyzing data, (c) representing data through abstractions, such as models and simulations, (d) automating solutions through algorithmic thinking, (e) identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources, and (f) generalizing and transferring this problem-solving process to a wide variety of problems*…
>
> *A second prominent definition is offered by Google’s Exploring Computational Thinking initiative, the first largescale program to provide an operational definition, disseminate resources, and promote discussion among K-12 educators about computational thinking. Google describes a process that includes four computational thinking techniques: decomposition, pattern recognition, pattern generalization and abstraction, and algorithm design...*” [18]

I lighed med disse definitioner bekræfter Angeli et al. (2016) ligeledes i deres gennemgang af forskningslitteraturen, at computationel tænkning er bredt anerkendt som tænkeprocesser, der involverer elementerne: abstraktion, generalisering, dekomposition, algoritmisk tænkning og debugging (fejlsøgning og korrigering af fejl). Forfatterne udfolder og foreslår en ramme

[18] Se CSTA: https://www.csteachers.org og Google: ”Exploring Computational Thinking”, http://www.google.com/edu/computational-thinking/ (lokaliseret 28. Maj 2017).

for at arbejde med disse elementer af computationel tænkning i relation til K-6[19]. Til trods for ovennævnte forsøg på at definere computationel tænkning, så er arbejdet med elevernes computationelle tænkning dog stadig svært at vurdere, fordi det er uklart, hvad det fx vil sige at kunne dekomponere, mønstergenkende, generalisere og abstrahere mønstre samt designe algoritmer? Endvidere eksisterer der ifølge Weinberg ikke nogle evidensbaserede og anerkendte metoder til at lære elever computationelle tænketeknikker. Der er fortsat meget lidt viden om, hvordan lærere (kan) undervise(r) i computationelle tænketeknikker (Weinberg, 2013).

I Coding Class projektet har Coding Class instruktørerne taget afsæt i Scratch. Scratch[20]definerer sin egen tilgang til computationel tænkning, der understreger bestemte begreber, praksisser og perspektiver. Brennan og Resnick (2012) er en del af forskningsholdet bag Scratch. De har med Scratch skabt grundlag for design-baserede læringsaktiviteter, hvor børn og unge mennesker lærer gennem design af egne interaktive medier. På baggrund af flere års udvikling af Scratch (2007-) og forskning i Scratch online fællesskabet og Scratch workshops har forfatterne udviklet en definition af computationel tænkning, der involverer tre dimensioner: *computationelle begreber* (de begreber designere anvender, når de programmerer), *design- og deltagelsespraksisser* (de praksisser designere udvikler gennem programmering) og *computationelle perspektiver samt et udforskende og skabende mindset* (de perspektiver på verden og dem selv, som designere former). De tre dimensioner kan ses som aspekter af udviklingen af børn og unge som digitale, kreative skabere. En proces, der, ifølge Brennan og Resnick, tager lang tid. De tre dimensioner er formuleret med afsæt i Scratch, men ifølge forfatterne kan de overføres til andre programmeringssammenhænge - og andre sammenhænge, hvor elever arbejder kreativt skabende med it og digitale medier. I det følgende har vi oversat de tre dimensioner til dansk.

Computationelle begreber omfatter ifølge Brennan og Resnick (2012):

- Sekvens: at identificere en serie handlinger og trin for computerens løsning af en opgave fx gå frem fem skridt, drej til venstre og gå frem 10 skridt.
- Loops: gentagelse af den samme sekvens af handlinger flere gange
- Hændelser: at en ting er årsag til at en anden ting sker. Fx ved museklik på et objekt ses en taleboble med teksten "hej".
- Parallelitet: at få flere ting til at ske på samme tid fx ved museklik på objekt afspilles en melodi og en taleboble med teksten "hej" viser sig
- Betingelser: at tage beslutninger baseret på betingelser - hvis… så.... fx hvis avataren rammes af en kugle, så mister man et liv.
- Operatorer: at understøtte matematiske og logiske udtryk såsom addition, subtraktion, multiplikation og division
- Data: lagring, indhentning, og opdatering af værdier via fx lister og variable

[19] K-6 dækker børnehave til 6. klasse.

[20] "Scratch, a computational authoring environment developed by the Lifelong Kindergarten research group at the MIT Media Lab. With Scratch, young people can design their own interactive media – including stories, games, animations, and simulations – by snapping together programming-instruction blocks, just as one might snap together LEGO bricks or puzzle pieces… In addition to the authoring environment, there is an online community where young people can share their projects, just as videos are shared on YouTube." (Brennan og Resnick, 2012, s. 2)

Brennan og Resnick (2012) peger på, at ovennævnte computationelle begreber alene handler om, *hvad* børn og unge lærer i designprocesserne. Det er ifølge forfatterne nødvendigt også at interessere sig for, *hvordan* børn og unge lærer gennem design- og deltagelsespraksisser:

Design- og deltagelsespraksisser omfatter (Brennan & Resnick, 2012):

- Inkrementel experimenteren og iteration: at udvikle trinvist, afprøve, for så at udvikle mere
- Testning og debugging: at sikre at tingene fungerer - at identificere og løse problemer, når de opstår
- Genbrug og remix: at skabe noget ved at bygge videre på eksisterende projekter og idéer
- Abstraktion og modularisering: at udforske forbindelser mellem helheden og de enkelte dele

Udover computationelle begreber samt design- og deltagelsespraksisser, har Brennan og Resnick bemærket, at børns og unges arbejde med digital design af interaktive medier (ikke kun via programmering) også rummer ændrede perspektiver på deres eget forhold til computeren og herunder udviklingen af et udforskende og skabende mindset:

Computationelle perspektiver og et skabende og udforskende mindset omfatter evnen til at kunne (Brennan & Resnick 2012):

- Udtrykke sig: at realisere computation som medium for kreation, “jeg kan skabe”
- Forbinde: at indse potentialet ved at kunne skabe med og for andre, "jeg kan gøre forskellige ting, når jeg har adgang til andre"
- Udforske og stille spørgsmål: at føle sig i stand til at stille spørgsmål om verden, "jeg kan (anvende computation til at) stille spørgsmål og skabe forståelse for (computationelle ting i) verden.”

Alle klasserne i Coding Class projektet refererer i en eller anden forstand til Scratch. Derfor har vi valgt, at relatere vores analyser af arbejdet med udviklingen af elevernes computationelle tænkning i Coding Class projektet til Brennans og Resnicks tre dimensioner.

I det følgende bevæger vi os fra, *hvad* der er i fokus, når skoler underviser i computationel tænkning (indhold), og *hvordan* børn og unge lærer om og gennem computationel tænkning (design- og deltagelsespraksisser samt perspektiver og mindset), til hvad lærerne skal kunne for at udvikle undervisningsaktiviteter med fokus på computationel tænkning.

Undervisning i computationel tænkning - pædagogiske greb og vurdering

Angeli et al. (2016) har udviklet Technological Pedagogical Content Knowledge (TPCK) for computational thinking ($TPCK_{CT}$) som en ramme for at definere den viden, som lærere skal have for at undervise i computationel tænkning:

Content Knowledge (CK_{CT}) er den viden om computationel tænkning og de kundskaber, som eleverne forventes at opnå, og som læreren derfor også bør have.

Learner Knowledge (LK_{CT}) er viden om, hvad eleverne kan have svært ved, fx når de skal udvikle abstraktioner, generalisere fra en løsning til en anden ved at identificere mønstre, dekomponere komplekse problemer og opdele i mindre og mere simple problemer, samt benytte algoritmiske tænkning til at løse problemer.

Pedagogical Knowledge (PK_{CT}) er den pædagogiske viden og de pædagogiske greb, der hører særligt til computationel tænkning og/eller kan overføres fra andre vidensdomæner til computationel tænkning. For eksempel er det væsentligt pædagogisk-didaktisk at iscenesætte og stilladsere elevernes arbejde med at modellere problemløsninger på iterative og inkrementelle måder, og elevernes arbejde med og inspiration fra andres arbejde (genbrug og remix).

Technology Knowledge (TK_{CT}) er den viden og de færdigheder, som lærerne skal have for løbende at kunne engagere forskellige teknologier, udvikle forskellige teknologier og redskaber, samt løse opgaver via forskellige digitale processer, metoder og værktøjer.

Derudover fremhæver Angeli et al. lærerens behov for **Context Knowledge** (CX_{CT}), hvilket refererer til lærerens bevidsthed om værdien af computationel tænkning i skolen, for samfundet og for eleverne. Kontekst viden omfatter også den bevidsthed læreren har om særligt lokale subjektive, kulturelle, organisatoriske og socioøkonomiske forudsætninger, der har betydning for arbejdet med computationel tænkning i undervisningen. For eksempel kunne en relevant viden her være de store regionale forskelle på it-kompetencebehov i Danmark (Danmarks Vækstråd, 2016). It-kompetencebehovene hænger sammen med de virksomheder og digitale praksisser, der eksisterer i forskellige dele af Danmark. Set i lyset at disse forskelle, vil arbejdet med computationel tænkning kunne tage sig forskelligt ud fx i København og i Vejle - ikke mindst hvis arbejdet med at udvikle undervisningsaktiviteter med fokus på computationel tænkning også knytter an til autentiske problemstillinger stillet i samarbejde med virksomheder fra skolernes lokalmiljøer. Sammenfattende foreslår forfatterne, at:

> "… TPCK for computational thinking ($TPCK_{CT}$) is defined as knowing how to: (1) Identify a range of creative and authentic computational thinking projects; (2) Identify a range of technologies with an appropriate set of affordances in terms of providing the necessary technological means for practicing/teaching the whole range of computational thinking skills with each project; and (3) Use the affordances of technology to transform CK_{CT} and PK_{CT} using representations that make the overall computational thinking experience comprehensible for all learners…"
>
> (Angeli et al. 2016., s. 54)

Lærerne skal altså ikke blot have styr på Scratch, men også have designmæssige kompetencer og en grundlæggende it-forståelse for at kunne undervise i computationel tænkning. I Coding Class projektet har omdrejningspunktet været spiludvikling. Det stiller yderligere krav om, at lærerne har blik for denne særlige form for teknologi.

Internationalt peger eksperter på at en af de grundlæggende udfordringer for lærere, der underviser med fokus på computationel tænkning er at lade elevernes interesse være i

centrum for arbejdet med problemløsninger (Committee for the Workshops on Computational Thinking, 2011, s. 27). Andre anerkendte udfordringer i arbejdet med computationel tænkning i skolen er, at computationel tænkning ikke har været en del af curriculum i læreruddannelsen, og langt de færreste lærere har derfor de it-kompetencer, der skal til for at kunne integrere informatik i deres undervisningsaktiviteter - lærerne er i regelen udfordret hele vejen rundt i forhold til tidligere nævnte TPCK$_{CT}$.

Samtidig peger samme eksperter (ibid.) på, at eleverne på nogle områder kan have stærkere it-kompetencer end lærerne. Et pædagogisk oplagt greb er derfor at arbejde med et ressourceperspektiv som tilgang til elevernes læring, hvor elever tilbydes mulighed for at være dygtigere end lærere såvel som deres jævnaldrende. Elever kan fx agere klasseeksperter, hjælpe med fejlsøgning og debugging, instruere andre elever og lærere i de dele af Scratch, som de allerede er superbrugere på. Lærere, ligesom elever, bliver kompetente over tid gennem brug af teknologi og erfaringer med at guide elevernes arbejde med at rejse forskellige spørgsmål, udfolde og udforske problemstillinger. Det kan således være hensigtsmæssigt, set fra et lærerperspektiv, at starte småt med afsæt i materialer der anvendes i en kort periode, for at se om eleverne opnår læringsfordele ved den pågældende tilgang. Endvidere har det (ibid.) vist sig effektivt at bede lærere selv identificere de temaer som de finder relevante for deres efter- og videreuddannelsesbehov. Ofte starter lærere med et ønske om hjælp til teknologien, derpå vejledning til at arbejde med udforskende spørgsmål og derpå hjælp til at arbejde med visualiseringer. Professionel udvikling kan derfor med fordel (ibid.) tage afsæt i videooptagelser af varierede undervisningspraksisser og efterfølgende dialog, hvor lærere i fællesskab diskuterer alternativer og identificerer gode og relevante skolepraksisser.

Coding Class projektet har, som tidligere nævnt, taget afsæt i en ambition om etablere et it-fag for alle elever i skolen. Dette udgangspunkt suppleret af Coding Pirates inspirerede aktiviteter, har betydet, at Coding Class aktiviteterne har været relativt frakoblet de forskellige fag og fagmål i skolen og i stedet knyttet til at lære om spiludvikling igennem programmering.

En række eksperter (Committee for the Workshops on Computational Thinking, 2011) har fremhævet, at potentialerne i computationel tænkning bedst realiseres i samspil med forskellige fagområder og discipliner:

> ”...the power of computational thinking is best realized in conjunction with some domain-specific content. Thus, to understand the human genome, individuals need to combine computational thinking and concepts in genetics. The diversity of possible contexts in which computational thinking applies illustrates its power…. Developing expertise in computational thinking involves learning to recognize its application and use across domains.”
>
> (ibid., s. 9)

Coding Class projektet knytter computationel tænkning til spildomænet, og tilbyder dermed blot én mulig åbning til arbejdet med elevernes computationelle tænkning i skolen. For eksempel kunne projektet have understreget udviklingen af andre pædagogiske miljøer med fokus på interaktive visualiseringer og simuleringer, modellering og fejlsøgning eller

mønstergenkendelse i store datasæt. Scratch er endvidere ét ud af flere mulige læringsmiljøer. Eftersom at instruktionsvideoerne til Scratch har fokus på spiludvikling, så er det umiddelbart intuitivt, at undervisningsaktiviteter med Scratch får denne retning.

Marcia Linn (ibid., s. 45-49) har rejst spørgsmålet om, hvorvidt arbejdet med computationel tænkning forudsætter et særligt eksklusivt forløb med fokus på computationel tænkning. Ifølge Linn er det i spændingsfeltet mellem og i relation til forskellige fagdiscipliners viden, at elever udvikler computationel tænkning. Set ud fra dette perspektiv bliver det afgørende at arbejde med computationel tænkning i tæt relation til faglig viden og faglige temaer fx inden for matematik eller dansk. Pointen, understreger Linn, er at eleverne skal opleve, at computationel tænkning netop ikke længere er et eksklusivt fagområde relevant for nogle få, men i stedet er essentielt, relevant og på mange måder grundlæggende for alle discipliners arbejde med viden i dag.

Computationel tænkning som kompetenceområde går heller ikke fri af den klassiske transferproblematik: Hvordan har vi sikkerhed for, at det eleverne lærer i en undervisningssammenhæng også kan aktiveres af eleverne ind i andre sammenhænge? Det er netop derfor, at Kolodner (Committee for the Workshops on Computational Thinking, 2010, s. 57) fremhæver det væsentlige i, at computationel tænkning bør være noget eleverne lærer i og på tværs af forskellige faglige sammenhænge i skolen. Med reference til transferlitteratur, argumenterer Kolodner for, at elever kun lærer at arbejde med computationel tænkning på tværs af kontekster, hvis (1) computationel tænkning som kompetenceområde praktiseres på tværs af sammenhænge, (2) kompetencerne anvendes og artikuleres i og på tværs af sammenhænge, (3) arbejdet med computationel tænkning sammenlignes og kontrasteres i og på tværs af sammenhænge, og (4) eleverne udfordres til at tænke på andre situationer, hvor de kunne anvende de samme kompetencer og analytiske tænkemåder. Kolodner fremhæver dog, at:

> "... a student's reflecting on a computational activity, being able to teach or help someone else learn the concepts, or being able to effectively articulate the relevant computational process at issue can be seen as likely indications that the student is learning computational thinking. As students are able to use increasingly elegant, efficient, and sophisticated approaches to tackle computational thinking tasks, this ability can also demonstrate learning and improvement in computational thinking… Although programming may be one tool that is used to teach or highlight computational concepts, it is not synonymous with computational thinking...
> a good definition of computational thinking is needed—both so that curricula will be designed to promote computational thinking and so that achieving capability in computational thinking can be measured well."
>
> (ibid., s. 60).

I et blogindlæg med overskriften "*Computer programming in schools… Can we avoid coding ourselves into a corner*" (25. maj, 2017) problematiserer Selwyn og Hillman det aktuelle fokus på kodning og programmering i det svenske skolesystem. De peger på, at der internationalt mangler evidens for, at det er via kodning og programmering, at eleverne opnår bedre muligheder for at lære andre fagligheder som fx matematik. Endvidere peger forfatterne på, at arbejdet med kodning og programmering i skolerne ofte antager en 'poppet'

quick-fix karakter, der formodentlig har meget ringe overførselsværdi til andre sammenhænge.

Arbejdet med elevernes computationelle tænkning i skolen er en ny dagsorden, der først er ved at vinde terræn og etablere sig. Dette arbejde kan og skal både gribes ved at fokusere på informatik *i* fag og *som* fag og vidensdomæne (Caspersen, 2017). Det faktum, at det er en ny dagsorden ses også afspejlet i Coding Class projektet, hvor Coding Class hjemmesiden knytter an til arbejdet med computationel tænkning i skolen *som* fag, men vi har hverken mødt Coding Class instruktører, kommunale koordinatorer eller lærere, der har artikuleret, hvordan Coding Class aktiviteterne har relateret til arbejdet med computationel tænkning i skolen. Det er således også en overvejelse værd om ovennævnte perspektiver på computationel tænkning meningsfuldt kan og skal danne afsæt for vurdering og videreudvikling af Coding Class projektet eller for den sags skyld for et fokus på computationel tænkning i skolen? Vi har i denne afrapportering valgt at fokusere på Coding Class projektet som en vej til at åbne for diskussioner af arbejdet med computationel tænkning i skolen. Når vi i senere afsnit åbner for de empiriske udfoldelser af forskellige tematikker, så vil vi derfor gøre dette med blik for både de udfoldede og uudnyttede potentialer i Coding Class aktiviteterne, som vi ser. Det er væsentligt at anerkende Coding Class projektets status som et pilotprojekt, der først og fremmest har haft til formål at dagsordensætte it som vidensdomæne og fag i skolen.

Da Coding Class projektet ikke har ekspliciteret hvilke læringsmål, der har været i fokus, mener vi, at det bedst giver mening at fokusere på, hvad der empirisk har udfoldet sig som læring i projektet - for elever og lærere. Herudover, så har vi valgt at perspektivere erfaringerne fra Coding Class med eksisterende viden om og internationale erfaringer med arbejdet med computationel tænkning i grundskolen.

Der eksisterer en mængde åbne spørgsmål (Committee for the Workshops on Computational Thinking, 2010, s. 33), når det gælder computationel tænkning som vidensdomæne. Et projekt som Coding Class projektet har den væsentlige funktion, at projektet bidrager til at åbne op for nogle af disse spørgsmål fx

1. Hvordan identificerer vi kompetencerne, der former computationelle tænkere?
2. Hvad er den bedste pædagogik til undervisning med fokus på computational thinking? Og ser den forskellig ud på forskellige klassetrin og i relation til forskellige fagområder i skolen?
3. Hvordan skal de videnskabelige miljøer informatik / datalogi's roller og funktioner være i forhold til arbejdet med computationel tænkning i skolen?
4. Findes der en struktur i folkeskolen for arbejdet med computationel tænkning?
5. Hvad er relationen mellem teknologi og computationel tænkning?

Larry Snyder, peger i samme rapport på, at metoder til evaluering af læring i undervisning med fokus på computationel tænkning er underudviklede. Snyder fremhæver, at der for eksempel ikke er nogen garanti for, at Brennan og Resnicks (2012) fokus på elevernes kreative udvikling af egne interaktive medier og på, at eleverne lærer at udtrykke sig, også er faciliterende for elevernes arbejde med problemløsning (som Wing har fokus på). Omvendt er der heller ikke nødvendigvis sammenhæng mellem arbejdet med computationel tænkning med fokus på problemløsning (hvilket ikke behøver at involvere digital teknologi)

understøtter, at eleverne bliver mere kreativt udviklende med egne interaktive medier. Brennan og Resnick peger da også selv på, at de via forskellige evalueringsmetoder har opdaget, at elevernes projekter (produkterne) ikke nødvendigvis i sig selv er repræsentative for elevernes læring. Fx har Brennan og Resnick via samtaler med eleverne afdækket, at nogle elever nok har løst komplekse programmeringsudfordringer og udviklet fremragende spil, hvor eleverne har brugt mange begreber og avancerede funktioner i Scratch, men at de har gjort det ved at kopiere en sekvens af kode fra et andet sted, remixet det ind i egen sammenhæng, uden at have forstået, hvad sekvensen egentlig gør. På denne baggrund viser Brennan og Resnick, at evaluering af elevernes læring og arbejde med computationel tænkning bør funderes i (løbende) samtaler med eleverne om deres læring og arbejdet med forskellige begreber, praksisser og perspektiver.

Den foregående gennemgang af computationel tænkning vil vi inddrage i den resterende del af rapporten, hvor vi vil fordybe os i empiriske analyser af, hvordan Coding Class projektets formål er blevet indfriet. Som nævnt i starten af rapporten, har projektet haft fire hovedformål:

1. at fremme elevernes forståelse for den digitaliserede verden, der omgiver dem nu og i fremtiden (læringsudbytte)
2. at fremme elevernes interesse for it (motivation)
3. at igangsætte undervisning, der fremmer elevernes evne til selv at arbejde mere kreativt og skabende med it i grundskolen (pædagogik)
4. at sætte fokus på it, kodning og computationel tænkning som vidensdomæne i grundskolen (de videre perspektiver)

Vores arbejde med at dokumentere og evaluere Coding Class projektet knytter an til disse fire formål og ambitionen om, på baggrund af Coding Class aktiviteterne, at vurdere de læringsmæssige og pædagogiske gevinster af denne form for undervisningsaktiviteter, samt udpege de udfordringer og muligheder der eksisterer, når det gælder ambitionen om videre udvikling og udbredelse til alle skoler i Danmark. På denne baggrund har vi struktureret den næste del af rapporten omkring fire temaer, der knytter an til de fire formål. Før vi udfolder de fire temaer vil vi imidlertid først sætte scenen ved at beskrive eksempler på, hvordan Coding Class aktiviteterne konkret har udfoldes sig i praksis i de cases vi har undersøgt.

Coding Class i praksis - et par cases

Coding Class aktiviteterne er som tidligere nævnt implementeret meget forskelligt på skolerne i projektet. I dette afsnit har vi valgt at gå relativt dybt ned i detaljen for at illustrere, hvor forskelligt Coding Class aktiviteterne og arbejdet med elevernes kreative it-kompetencer kan lande ude på skolerne. Vi har valgt at præsentere to markant forskellige cases fra samme kommune. De to cases illustrerer forskellige ståsteder og forudsætninger for Coding Class aktiviteter - både set fra lærer- og elevsynspunkt. De to cases illustrerer også, hvor væsentligt det er at tale om læringspotentialer set i forhold til de udfoldelsesmuligheder, der har været i de forskellige sammenhænge. Vi benytter de to cases til at udfolde og diskutere arbejdet med de læringsmæssige og pædagogiske perspektiver i Coding Class projektet.

Eftersom at Coding Class må betragtes som ny praksis i grundskolen, betragter vi det som væsentligt at udfolde specificiteterne af disse praksisser. Det er først ved at kigge på specificiteterne, at vi kan opnå bevidsthed om, hvordan elevernes kreative it-kompetencer kan udfordres og engageres. I det følgende tillader vi derfor også længere interviewcitater, for at lade skolernes stemmer komme til orde. Det er væsentligt både at udfolde nuancerne i aktiviteterne og give afsæt for at kunne diskutere muligheder og udfordringer - set i forhold til bredere skoleudbredelse.

De lærere, der har deltaget i Coding Class fra de to skoler, har haft meget forskellige ståsteder og forudsætninger for at arbejde med elevernes skabende it-kompetencer, Scratch, programmering og kodning. Dog er det fælles for begge skoler, at Coding Class er introduceret som en aktivitet, der berører bestemte klasser og lærere, og ikke hele skolen. Hverken de to skoler eller kommunen har i udgangspunktet defineret et langsigtet formål med at deltage i projektet.

Skole K - “nybegynderne”

Skole K har haft fire sessioner (hele skoledage) med Coding Class instruktørerne, som har organiseret forløbet og alle sessionerne i dialog med lærerne. Forløbet strakte sig over en måned i perioden fra 10. november - 20. december 2016. Vi besøgte skolen den sidste session med Coding Class instruktørerne, 20. december. Både lærere og Coding Class instruktører beskriver Skole K, som en skole, hvor de lærere, der blev tilknyttet Coding Class projektet - men også skolen som helhed - startede mere eller mindre på bar bund i forhold til arbejdet med Scratch, at programmere og kode samt udvikle computerspil. Skolen valgte, at Coding Class skulle omfatte hele 6. årgang, som bestod af tre klasser - i alt ca. 60 elever, som blev samlet til fire Coding Class dage. Udover de to Coding Class instruktører var årgangens tre matematiklærere tilknyttet forløbet. Ingen af de tre lærere havde tidligere arbejdet med kodning, programmering og spiludvikling. Både lærere og elever fortæller, at den ene lærer, Jytte[21], er en lærer der “aldrig bruger it i undervisningen”. Den anden lærer, Emma, bruger it en del i sin undervisning. Emma og Jytte deltager i interview om Coding Class aktiviteterne. De fortæller, at det var uklart fra begyndelsen, hvilke fag der ville være

[21] Alle navne er ændret for at sikre respondenternes anonymitet

oplagte at knytte an til. Dansklærerne fandt indledningsvist projektet interessant, men det endte hos matematiklærerne.

> *Jytte: tænkte bare, at når det var noget med computeren og at eleverne skulle bruge computeren, så ville det være godt for dem, for det er jo også den verden vi er i. Og når jeg nu ikke er den mest ferme til at sætte dem ind i det.*
>
> *Emma: jeg var med til intromødet på kommunen [sammen med Coding Class instruktørerne]. Der kan jeg huske, at jeg havde sådan en følelse af, at det var meget bredt. At det kunne mange ting. Jeg tænkte det kunne noget med problemløsning og at tænke kreativt i matematik. At det var en måde at få det på skemaet – som ellers nogle gange kan være lidt svært. Også det med et lidt større projekt. En måde man kunne arbejde sammen i grupper i matematik også [i dansk har man jo sådan et større projekt] om et fælles produkt. Det synes jeg ellers, kan være svært i matematik at lave den form for gruppearbejde. Jeg synes først, det var, da vi var til introdag på Skole M med Coding Class instruktørerne, at det var der, jeg egentlig fik indtryk af, hvad vi skulle. Indtil da havde det været meget høje tanker og snak.*
>
> *Jytte:: vi var også på kursus [introdag]. Så lavede vi vores eget projekt - det blev vores projekt.*
>
> *Emma: vi prøvede selv at lave musespil i scratch*
>
> *Jytte: vi kørte det ind i Min Uddannelse, med tidsplaner for hvad vi skulle. Men det er jo ikke vores projekt på den måde. Vi lægger jo elever til. Og så er det jo Coding Class instruktørerne, der skulle komme med oplæg, og så skulle vi gå ind i det.*
>
> *Emma: vi blev enige om, at vi kom fra ret forskellige baggrunde [de tre lærere], så vi var ikke der, hvor vi kunne sige præcis, hvordan tingene skulle være. Vi havde hverken tid eller overblik til at gennemskue, hvad programmet kunne. Vi så en plan Coding Class instruktørerne havde lavet – tror jeg – til en skole i København, og så tror jeg, vi sagde, at hvis det var ok med dem, så ville vi egentlig bare køre noget lignende. Det ville være en alt for stor opgave for os at specificere, hvad vi ville bruge det til, når vi kom fra så forskellige baggrunde.*
>
> *Jytte: så fik vi struktur og et billede på, nårhh det er den her vej, vi skal.*
>
> *Emma: så de [Coding Class instruktørerne] er kommet meget med idéerne, sådan rent indhold, og så har vi trukket didaktik ned over det. De har ikke samme baggrund for at have en idé om, hvor lang tid tingene tager. Så det synes jeg, at hver gang vi har haft det, så er vi kommet ind på sådan en god vekselvirkning i fht. hvad de kan tilbyde og hvad vi kan tilbyde i fht. struktur på dagen. De første par dage, de sidste timer*
>
> *Jytte: ja, de var rimeligt kaotiske*
>
> *Emma: de var rimeligt hårde, fordi der skulle eleverne lige pludselig til at lave virksomheder og være kreative over middag, og det kunne man se på nogen, at det var måske ikke lige der, at de skulle til at tænke nye og kreative tanker. Så hver gang er vi kommet lidt nærmere at få stykket noget sammen, der er struktureret*
>
> *Jytte: og passer til elevernes energi, og hvornår de er kørt trætte*

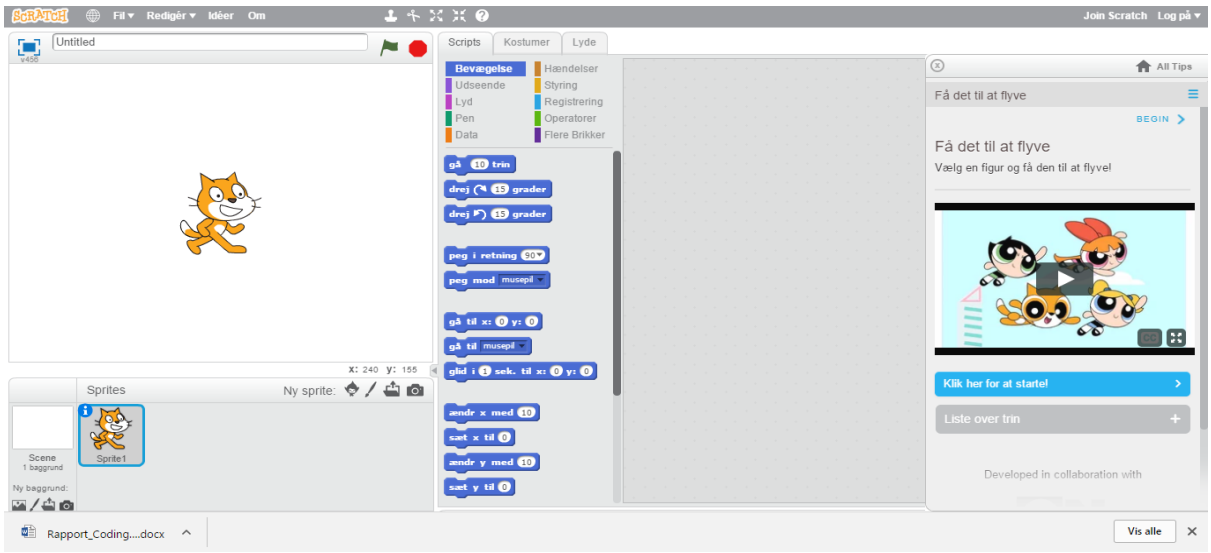
Eleverne skulle etablere spil-virksomheder i grupper (med navn og logo), hvor de arbejdede sammen om at udvikle et computerspil via Scratch. Grupperne fordelte selv fire

ansvarsområder og roller imellem sig, således at opgaverne blev delt ud:

-Spil-mekaniker
-Grafik
-Lyd
-Projektledelse

10/11 - Dag 1 “Opstart”

Den første dag var fokus på introduktion til Coding Class projektet samt til kodning. Eleverne skulle starte med at udvikle et flyvespil baseret på en af Scratch instruktionerne[22]. Guiden til at udarbejde et flyvespil rummer grundlæggende introduktion til Scratch med afsæt i at forstå programmeringsmiljøet Scratch.



Efterfølgende skulle eleverne udvikle et muselabyrintspil[23], som er et lidt mere kompliceret spil. Eleverne blev først bedt om at følge instruktionsvideoen udviklet af en Coding Class instruktør og lave et spil. Dernæst blev eleverne opfordret til at lave spillet “federe” - udvikle nye baner, bedre grafik og lave lyd til eller finde lyde på nettet. Dernæst skulle eleverne oprette deres spilvirksomheder, og de fik præsenteret den spiludfordring, som de skulle arbejde med: lav et simpelt spil.

Den instruktionsvideo fra kreakode.dk, som eleverne blev bedt om at følge, er egentlig en video som er målrettet lærere. Under videoen ses hvilke læringsmål og tekniske mål, der fx kan være i fokus i forløbet:

I forløbet arbejdes med følgende begreber:
– Algebraisk tænkning

[22] Se https://scratch.mit.edu/projects/editor/?tip_bar=fly
[23] Se www.kreakode.dk/kurser

– logisk opbygning
– abstrakt tænkning
– koordinatsystemet
– negative tal
– variable
– problemløsning
– vinkler og grader

I forløbet bliver man introduceret til følgende funktionaliteter i Scratch:
– visuel kodesyntaks
– Sprites
– Baggrunde
– if-statements
– keybindings
– tilfældige tal
– kodning af farver
– loops

(Kilde: kreakode.dk)

17/11 - Dag 2 “Idégenerering”

Den anden dag arbejdede eleverne med at generere spil-idéer. I grupperne skulle eleverne finde på syv spil-idéer. Den ene idé skulle ligge til grund for deres spil. Eleverne skulle huske, at de udviklede et simpelt spil, med én spilmekanik. Derpå skulle elevgrupperne besvare fire spørgsmål:

- *Hvad er målet?*
    - *Samle mønter, skyde monstrer, hoppe højest osv.*
- *Hvem er spilleren?*
    - *En prinsesse, en troldmand, en soldat?*
- *Hvad styrer man med?*
    - *Musen, tastaturet, én knap eller flere knapper?*
- *Er spillet til én eller flere spillere?*

Grupperne blev bedt om at lave udkast til deres spil på papir, og Coding Class instruktørerne understregede, at det var vigtigt, at grupperne gemte deres tegninger og tanker. I plenum gennemgik Coding Class instruktørerne forskellige spilgenrer:

- *Forhindringsspil*
- *Flappybirdspil*
- *Platformshopspil*
- *Omvendt hop spil*
- *Flyvende grise spil*
- *Endless runner - stampede*
- *Katapult spil / angry birds*
- *Minestryger*

Derpå fik eleverne en kort introduktion til "Rygsækken" i Scratch, Scratch Studiet og Piskel[24] (pixel-art). Så blev eleverne sluppet løs og kunne gå i gang med at udvikle spillene. Dagen blev afsluttet med gruppernes præsentationer af deres spilkoncept - med afsæt i de fire spørgsmål. Herudover skulle grupperne vise deres idéer og tegninger, fortælle om problemer de var stødt på undervejs, hvordan de løste dem, og elevgrupperne præsenterede noget af den kode, de havde lavet i Scratch.

13/12 - Dag 3 "Spilprogrammering"

Den tredje dag havde fokus på elevernes udvikling af samt præsentation for lærere og Coding Class instruktører med mulighed for sparring om spiludviklingen. Elevgrupperne startede med at have 10 minutter, hvor de talte om spillet i gruppen:

- *Hvor langt er I?*
- *Hvad mangler I?*
- *Hvad er den vigtigste opgave i dag?*
- *Uddel opgaver til hver person i gruppen - minimum 1 opgave pr. person!*

Derpå gik eleverne i grupper i gang med at arbejde med spillet. Når lærere og Coding Class instruktører havde mulighed, præsenterede elevgrupperne spillet for dem med udgangspunkt i følgende spørgsmål:

- *Hvad er spil-mekanikken?*
- *Hvad er historien?*
- *Hvad kunne gøres bedre ved spillet?*

Eleverne blev mindet om, at præsentation og respons ikke var en bedømmelse af spillet, men en måde at få gode idéer og råd. Eleverne skulle arbejde på spillet både før og efter pitchen. Efter pitchen blev grupperne bedt om at lave en fem minutters opsamling i gruppen ud fra disse fire spørgsmål:

- *Hvor langt er I nået, og hvad skal I nå inden dagen er omme?*
- *Skal der uddeles nye opgaver?*
- *Spil spillet sammen, så alle ved hvor langt i er.*
- *Har I husket lyd-effekter?*

Som afslutning på dagen skulle eleverne individuelt vælge og gennemføre en af tre tutorials på scratch[25]:

- *Hide and seek*

[24] www.piskelapp.com/
[25] se https://scratch.mit.edu/go

- *Race to Finish*
- *Catch*

Derpå skulle elevgrupperne give spillet et juletwist for eksempel med et rensdyr, en julemand, nisser, risengrød eller et juletræ.

20/12 - Dag 4 “Afslutning”

Den fjerde dag var afslutning på Coding Class forløbet på Skole K, både fordi en lærer skulle på barsel, og fordi de ikke havde fundet virksomheder i kommunen, som eleverne kunne besøge og præsentere deres spil for. Den sidste dag begyndte med at grupperne i ti minutter talte om spillet i gruppen for at planlægge dagen. De tog udgangspunkt i følgende spørgsmål:

- *Hvor langt er I?*
- *Hvad mangler I?*
- *Hvad er den vigtigste opgave i dag?*
- *Uddel opgaver til hver person i gruppen - minimum 1 opgave pr. person!*

Udover at færdiggøre spillene skulle grupperne endvidere lave en plakat, hvor de præsenterede spillets elementer (se nedenstående liste) og gruppens arbejde med spillet:

- *Gameplay*
- *Controls*
- *Karakterer*
- *Mål*
- *etc.*

Grupperne skulle på plakaterne beskrive og reflektere over, hvilke problemer grupperne var stødt ind i under udviklingen af spillet.

Eleverne kunne igen vælge individuelt at arbejde med en af tre tutorials på Scratch:

- *Hide and seek*
- *Race to Finish*
- *Catch*

Med afsæt i det juletwist grupperne havde valgt, skulle de nu lave en endless-runner med for eksempel julemanden, rensdyr og gaver! Til sidst skulle grupperne besøge hinanden og afprøve hinandens spil i en spil-expo - en fremvisning af deres spil.

Skole M - de mere erfarne

Skole M har 0.- 6. klasse. Hvor Skole K's forløb varede en måned, så strakte Skole M's forløb sig over et halvt år.

Skole M's lærer (Leo) underviser i natur og teknik og matematik i den Coding klasse, som han har deltaget i projektet med. Han har integreret Coding Class aktiviteterne med begge fag, hvilket betyder at de har haft 7-8 lektioner til rådighed hver uge til at være "Coding Class". Leo har haft klassen fra 4. klasse. De går nu i 5. klasse. Fra 1.-3. klasse havde klassen en anden matematiklærer. Og hun arbejdede altid i bogen. Leo underviser i matematik, natur og teknologi, idræt, kodning og animation. Kodning består af to obligatoriske lektioner for 6. klasserne. De har også innovation på skolen for 6. klasserne[26].

Leo har været i fuld gang med at arbejde med en skabende tilgang til it og digitale medier i hans undervisning før deltagelsen i Coding Class. Leo fortæller, at han inden Coding Class var begyndt at at bruge Scratch og andre programmer. Han havde nogle elever, som havde svært ved at lære koordinatsystemet, og så valgte han at de skulle bruge Scratch, så de fik styr på x- og y-aksen. De brugte også Scratch i den understøttende undervisning, til at blive bedre til matematik. Og det var egentlig hovedsageligt, fordi den understøttende undervisning på det tidspunkt mest blev brugt til lektiecafé, og det fandt Leo var spildtid.

Scratch og andre computerprogrammer blev således indgangen til at arbejde eksplorativt i samarbejde med eleverne om de pædagogiske gevinster ved forskellige programmer. Leo fortæller, at han og eleverne også har prøvet programmer, som de har droppet, fordi de ikke havde så stor succes grad med dem. Leo fortæller, at før Coding Class havde han således lært fx TinyTap[27] og Sploder[28].

Udover at have inddraget natur og teknik samt matematik, så peger Leo på, at undervisningen også i høj grad rummer engelsk, uden, at de har engelsk.

> *L: Mange af programmerne er engelsksprogede, så der skal de bare ind og læse på engelsk. Det er jo bare en sidegevinst i mit hoved.*

Leo har introduceret sine elever til en hel palette af multimodale repræsentationsformer, som de nu kan navigere i og på tværs af, og agere kreativt og skabende med. Det betyder, at eleverne kan arbejde kreativt med forskellige former for it programmer i og på tværs af fag, hvis lærerne i de andre fag tillader det. Det har endvidere betydet, at Coding Class instruktørerne på Skole M har skullet bidraget med noget ekstra, idet Leo allerede kunne Scratch før starten af Coding Class projektet.

Leos Coding Class klasse har formidlet og deltaget i en del aktiviteter for eksempel "Danish Entrepreneurships Awardships", de har holdt oplæg for lærere og elever på en af

[26] Skolen har to lærere, der deltager i Edison med klasserne. Se: http://www.ffe-ye.dk/media/39465/Alt-om-Edison.pdf
[27] Se: https://www.tinytap.it/activities/
[28] Se:http://www.sploder.com/

kommunens andre Coding Class skoler, og de har holdt oplæg for alle Pædagogiske LæringsCentre (PLC'er) i Syddanmark. Leo fortæller, at hans elever er vant til at præsentere, og de har en forventning om det. Klassen vandt i marts 2017 den landsdækkende matematikkonkurrence "Kænguruen", hvor 4.-6. klasser kunne tilmelde sig. De blev Danmarks bedste 5. klasse. I den forbindelse, fortæller Leo, var der en, der spurgte om han havde forberedt noget - nu hvor de vandt. Leo svarede, at det ikke var det store, han havde forberedt. Leos elever fortæller ligeledes, at Leo bare havde sagt "lav de der opgaver". Leo ved ikke selv om den måde, de arbejder kreativt med it på bidrager til, at eleverne har klaret sig så godt, men han finder det tankevækkende.

Leo er ikke en lærer, der på stående fod kan sætte pædagogiske begreber på sin tilgang til undervisningen. Men den er grundlæggende eksplorativ, og frem for et fokus på undervisningsteknologier[29], kan man sige, at Leo har fokus på eksplorationsteknologier[30].

> *Vi startede med Scratch og så som Coding Class. Så havde vi tid til at finde ud af, hvad vi ville. Så sagde jeg, at jeg ville arbejde med virksomheder. Det har jeg prøvet tidligere i matematik. Virksomhederne skulle have et budget. Det ville [eleverne] finde tilfældigt via nogle terninger. Stjålet fra mit matematikforløb. Så skulle de kunne tjene nogle penge på at løse opgaver. Og nogle gange var det de bedste der fik og andre gange var det alle der fik et grundbeløb. Hvis nu de laver undervisningsforløb i Tinytap fx Så sætter jeg kr. 1000,- ind på deres konto. De glemmer det nogle gange. Men penge er ikke det vigtigste, det er konkurrencen. Og så er vi i gang. Jeg ville gerne have de kunne lave computerspil. Først og fremmest. Det var der, jeg tænkte, vi skulle ind og kode. Og for at have 21st skills med, så ville jeg have, at de også skulle lave merchandise. Hvor vi kunne 3D printe eller stickers eller hvad det måtte være.*
>
> *Så måtte vi jo så se, hvilke programmer der kom hen ad vejen.*

På Skole M har Coding Class instruktørerne kørt et særligt forløb, hvor de har besøgt Leos klasse tre gange. Dette forløb er markant anderledes og - ifølge Coding Class instruktørerne - en hel del mere avanceret end det forløb, som Coding Class instruktørerne har kørt via fire sessioner på Skole K. Der er tydeligt forskel på forudsætningerne for 6. klasserne på Skole M og 5. klassen på Skole K , samt på deres tilgang til, at arbejde skabende med it og digitale medier, herunder på relationen og forventninger mellem lærere og elever.

7/12 - Dag 1 "Introduktion"

Leo og Coding Class instruktørerne vurderede, at Coding klassen på skole M var så langt allerede, at de kunne starte med PlayCanvas[31], der er et udviklingsmiljø ligesom Scratch, blot mere avanceret. I PlayCanvas er elementerne i 3D. Coding Class instruktørerne startede dagen med at introducere et designdokument, som elevgrupperne skulle oprette og arbejde med som redskab. Et dokument, der hjælper med at:

[29] Som fx didaktiserede læremidler - digitale såvel som analoge
[30] Med inspiration fra Paulo Blikstein, er eksplorationsteknologier, platforme der kan benyttes som afsæt for elevernes udforskning af fx matematiske begreber, processer eller noget andet.
[31] Se: https://playcanvas.com/

- *Holde styr på Jeres idéer*
- *Alle ved hvad de skal lave (hvis I har skrevet det i dokumentet)*
- *I kan se hvordan spillet udvikler sig*

Derpå fik eleverne en PlayCanvas introduktion med fokus på, at:

- *Elementerne i PlayCanvas er i 3D*
    - *Dvs. at I kan lave 3D-modeller i Tinkercad[32] og indsætte i Jeres spil*
- *I kan alle sammen arbejde på spillet på samme tid, fra forskellige computere*
- *Billeder/Baggrunde laves i piskelapp*[33]

Coding Class instruktørerne fortalte derefter, at de havde lavet fire spil-'templates' (skabeloner til spillene Flappy Bird, Pong, Rytmespil, Endless runner) og skulle bruge elevernes hjælp til at gøre spillene færdige. Noget af det spillene mangler er: Fede 3D-figurer, lyd-effekter og musik, bug-fixes og rettelser i koden. Eleverne blev bedt om at prøve de fire spil[34] og aftale efterfølgende i grupperne, hvilket spil de gerne ville arbejde med. Grupperne blev endvidere bedt om at tænke over, skrive ned og tegne i deres designdokument. Opgaveformuleringen var som følger:

- *Hvordan skal karakteren(e) se ud?*
- *Hvilke lyde skal, der bruges?*
- *Er der noget, der mangler i spillet?*
- *Fordel roller og opgaver:*
    - *Hvem laver 3D? Hvem er projektleder? Hvem står for programmeringen?*
    - *Hvilke opgaver skal vi nå i dag? Hvilke opgaver er vigtigst?*

    HUSK: Det kan være en god idé at tegne de ting man vil have med i hånden/på computeren inden man laver 3D-modeller!

    Sæt igang! Gå ind på det spil I har valgt:
    Log ind oppe i højre hjørne med de brugernavne og passwords vi har givet jer. 'Fork' projektet ved at trykke på ikonet med en kniv og en gaffel på. Inviter din gruppemedlemer til dit forkede projekt.

Endvidere fik elevgrupperne besked på at øve sig i programmering inde på CodeCombat[35].

[32] Se: https://www.tinkercad.com/
[33] Se: http://www.piskelapp.com/
[34] Rytmespil: bit.do/skibetrytme, Pong: bit.do/skibetpong, Endless runner: bit.do/skibetrun, Flappy bird: bit.do/skibetflap
[35] Se: https://codecombat.com/

8/12 - Dag 2 “Gruppe opstart”

På anden dagen blev grupperne bedt om at holde et opstartsmøde, hvor de åbnede deres design dokument og talte om:

- *Nåede I det, I skulle i går?*
- *Hvilke opgaver er de vigtigste at nå i dag?*
- *Spil jeres spil på en computer, og tal om, hvad der er ‘hot’ og ‘snot’ ved det*
- *Fordel minimum 1 opgave til hver person i gruppen*

Herefter fik alle elever til opgave at skabe en 3D-figur:

Tal sammen i gruppen omkring hvilke 3D-figurer der mangler:

- *Skal I bruge snefnug?*
- *Et rensdyr?*
- *Julemanden?*
- *En kage?*
- *En taco?*
- *Lasere og lyn?*

Tegn figuren på et stykke papir/på computeren først

Når I har tegnet figuren i TinkerCad gemmer I den ved at gå op i venstre hjørne og trykke på ‘Design’, vælg 3D-print og efterfølgende ‘.OBJ’.

Indsæt 3D-figurerne i PlayCanvas!

1. *Højreklik nede i ‘Assets’ vinduet i bunden af skærmbilledet*
2. *Gå op i toppen af menuen og tryk på ‘Upload’*
3. *Find jeres 3D-figur og vælg den*
4. *Nu er figuren inde i PlayCanvas og kan kodes*

Efter arbejdet med 3D-figurerne skulle eleverne arbejde med kodeskrivning - ‘Få ting til at falde’-script. Eleverne skulle lytte til Coding Class instruktøren, som præsenterede, og skrive det der står på skærmen:

- *Alt med // foran er kommentarer, det bliver ikke læst som kode*
- *Update funktionen opdaterer spillet 1 gang pr. frame*
- *Vi flytter snefnugget 0.1 skridt ned (af y-aksen) hver frame*
- *Hvis snefnuggets position bliver UNDER 0, ændre vi y-positionen til 10.*

Herefter skulle elevgrupperne gå i gang med deres spil.

- *Lav de opgaver I har besluttet i gruppen!*
- *Hvis man mener man ikke har noget at lave, går man ind på www.codecombat.com og øver sig*

- *Hvis gruppen mener deres spil er helt færdigt, viser man det til Leo eller Coding Class instruktørerne, der så kommer med kritik*
- *ALTERNATIVT: Spillet bliver vist for klassen eller til en anden gruppe, som så kommer med idéer til ting der kunne gøre spillet bedre!*

22/2 - Dag 3 "App Inventor"

Den tredje dag blev eleverne præsenteret for App Inventor[36]:

- *Et program I kan bruge til at lave jeres egne apps!*
- *Fungerer lidt ligesom Scratch*
- *Hjemmeside: appinventor.mit.edu*

Gennemfør Katte-opgaven:
http://explore.appinventor.mit.edu/ai2/hellopurr

Vi skal lave virksomheds Apps.

Jeres App skal indeholde:

- *Præsentation af jeres virksomhed*
  - *Hvad laver I, hvad har I lavet osv.*
- *Information omkring 'Om Os'*
  - *Hvem er med, hvad er jeres titler, slogan osv.*
- *Regnskab*
  - *Hvor mange penge har I, hvad har I brugt pengene på osv.*

I slutningen af dagen skulle eleverne vise deres app frem og fokusere på:

- *Hvad er målet med jeres app - hvordan har I gjort den spændende, sjov, anderledes?*
- *Hvad kunne I godt tænke jer at få med?*

Som vi har illustreret via de to cases ovenfor, så er det er meget forskelligt, hvordan design-baserede læreprocesser og computationelle tænkemåder er udtrykt i Coding Class aktiviteterne. Coding Class instruktørerne har været forskelligt medskabende i forhold til at ramme- og indholdssætte aktiviteterne. Når vi kigger på de to beskrevne forløb, så er det tydeligt, at de repræsenterer et fokus på især design-baserede læreprocesser. Herudover er det tydeligt, at begge forløb rummer fokus på kodning og programmering, på at lære forskellige (mini-)spilgenre at kende, og på spiludvikling. Baseret på vores interview og observationer, kan vi konstatere, at Informatik og computationel tænkning imidlertid ikke har været så tydeligt talt frem af Coding Class instruktørerne (og heller ikke af lærerne). Heller ikke, hvordan Coding Class aktiviteterne har knyttet an til faglige læreprocesser.

[36] Se: http://appinventor.mit.edu/explore/

På Skole K har Coding Class instruktørerne været dagsordensættende, og de har været de primære designere af og drivkraft for såvel det faglige indhold i forløbet, som de arbejdsmetoder elevgrupperne har mødt og været engageret i. De fire sessioner har udgjort Coding Class på Skole K. På Skole K har Coding Class været en projektaktivitet 6. klasses lærere og elever deltog i over fire sessioner, hen over en måneds tid.

På Skole M, derimod, har Coding Class instruktørerne suppleret det allerede igangværende i Leos aktiviteter og 5. klasse. Leos klasse *er* en kode-klasse, og det er ikke en projektaktivitet, som de i en kortere periode har deltaget i. Det er snarere en identitet de har taget til sig. I Leos Coding klasse og undervisning kan vi tale om at arbejdet med Coding Class aktiviteterne er blevet en mulighed for at styrke og videreudvikle de allerede igangværende it-pædagogiske greb med fokus på arbejdet med elevernes skabende it-kompetencer som vej til både elevernes digitale og faglige dannelse. Disse greb har Leo har udviklet og arbejdet med i klassen siden 4. klasse. Men det er ikke greb og en tilgang til undervisningen, som Leo bare kan beskrive. Leos måde at være lærer på og undervise er i høj grad implicit og ikke fyldt med en bunke etablerede og udtrykte tegn på, hvordan Leo får sin undervisning til at fungere, og hvordan han formår at motivere eleverne i klassen og skabe højt elevengagement og elevtrivsel i klassen.

Tema 1: Elevernes læring i Coding Class

Indledningsvist har vi gennemgået, hvordan forskere internationalt har forbundet computationel tænkning med dels problemløsning og dels en særlig deltagende tilgang. Derpå har vi fordybet os i to cases, der illustrerer hvor forskellige Coding Class aktiviteterne har udfoldet sig i forskellige skolesammenhænge i henholdsvis én 5. klasse og tre 6. klasser på to forskellige skoler. I det følgende vil vi uddybe, hvilke læreprocesser og læringsudbytter vi har iagttaget, at Coding Class forløbene i særlig grad har udfoldet. Disse observationer er afhængige af det lokale setup i forhold til, hvordan elevernes arbejde med programmering via spiludvikling er blevet rammesat. De identificerede læreprocesser illustrerer, hvordan arbejdet med kodning, programmering, elevernes kreative it-kompetencer via spiludvikling og computationel tænkning og deltagelse kan udfordre og styrke elevernes kompetencer.

Som tidligere nævnt, peger Brennan og Resnick (2012) på, at eleverne kan arbejde med udvikling af et program uden at have reflekteret over og bevidstgjort betydningen af processerne, og de begreber, som er involveret i deres skabende arbejde med it. Det er langt fra givet, at begreber, designpraksisser og det skabende mindset bliver udfoldet og bevidstgjort gennem elevernes arbejde med fx Scratch. Arbejdet med at udvikle spil og samtidig skabe bevidsthed om centrale computationelle tænkemåder, designpraksisser og udviklingen af et skabende mindset forudsætter, som også Brennan og Resnick peger på, løbende metakommunikation, dialog og processuel evaluering.

Vi har flere gange i denne rapport peget på, at arbejdet med informatik og computationelle tænkemåder i Coding Class projektet ikke har været tydeligt defineret og udtrykt fra begyndelsen af projektet. Det er heller ikke noget, vi har kendskab til er blevet markant tydeligere på baggrund af projektet ude på skolerne. Der er imidlertid ikke tvivl om at både lærere og elever står et andet sted i dag, i forhold til arbejdet med kodning og programmering og spiludvikling. I gennem vores interview med Coding Class instruktørerne, de kommunale koordinatorer, lærere og elever, og vores observationer i undervisningen på skolerne, er det blevet tydeligt for os, at tegn på design-baserede læreprocesser, elevernes læring af informatik og computationelle tænkemåder samt elevernes faglige læring i forløb som Coding Class, har brug for at blive identificeret, talt frem, og begrebsliggjort. Eleverne har arbejdet med design-baserede læreprocesser i Coding Class, hvilket har været mere indirekte udtrykt end egentligt peget ud af instruktører og lærere, såvel som IT-Branchen. Coding Class aktiviteterne har haft informatik og computationel tænkning som referenceramme. Det har mere været en antagelse, at elevernes arbejde med spiludvikling, kodning og programmering ville være lig med, at eleverne lærer fx matematiske begreber og opnår matematisk forståelse på nye måder, at arbejde designbaseret, og at eleverne tilegner sig informatik og computationelle tænkemåder som almendannende elementer af undervisningen.

De Coding Class aktiviteter vi har kigget på, har ikke haft særligt fokus på fag, men på elevernes arbejde med design-baserede læreprocesser, spiludvikling, programmering og virksomheder. Flere lærere og de kommunale Coding Class koordinatorer nævner imidlertid, at det kunne være et væsentligt næste skridt i arbejdet med Coding Class aktiviteter at få fremhævet koblingsmuligheder til flere forskellige fag (også de humanistiske og musisk-

kreative) og faglige læringsmuligheder.

I de følgende afsnit vil vi med afsæt i vores samtaler med de forskellige Coding Class aktører og besøg på skoler sætte fokus på eksempler på, hvordan Coding Class aktiviteterne har involveret læreprocesser og læringsudbytter for eleverne, der knytter an til blandt andet informatik og computationel tænkning, faglig forståelse, skabende og kreativt arbejde med it, it-interesser, samarbejde og design-baserede læreprocesser.

Informatik

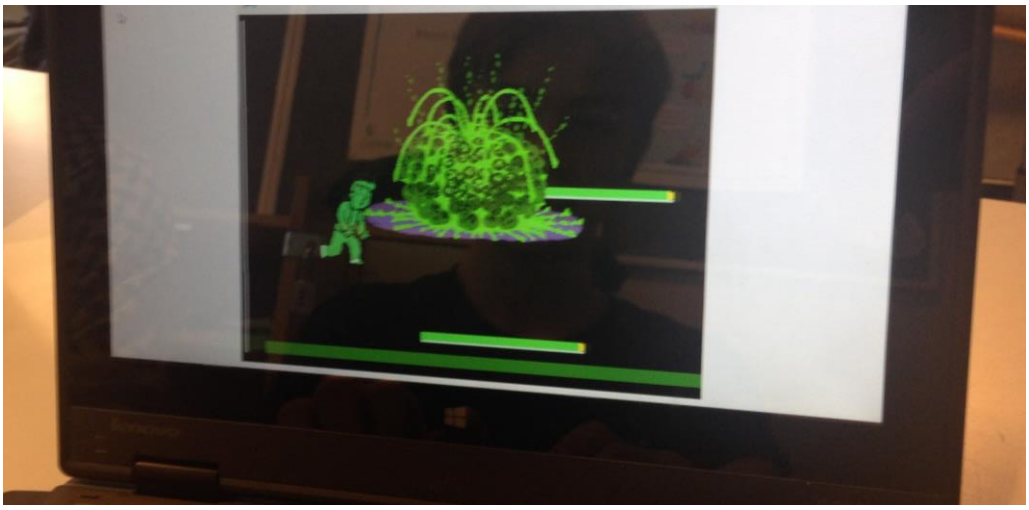
For at kunne udvikle et computerspil, skal eleverne skrive kode[37] og bygge et program op. I arbejdet med at udvikle et program skal eleverne blandt andet kunne identificere program sekvenser og fokusere på centrale koncepter - programelementer - som programmet bygges op omkring. De skal forstå tænkningen knyttet til computere. Computationel tænkning er blandt andet forbundet med abstraktion, generalisering, dekomposition, algoritmisk tænkning og debugging (fejlfinding og korrektion af fejl). Iteration betragtes inden for computationel tænkning som en væsentlig arbejdsmetode til udvikling af computationelle artefakter[38]. Flere af de lærere vi har talt med nævner, at eleverne har arbejdet med at videreudvikle deres produkter, og det ses også, at netop iteration har været væsentligt aspekt af Coding Class instruktørernes måde at tilrettelægge arbejdet med elevgruppernes programmering og spiludvikling. Grupperne har arbejdet med design af deres spil og med udviklingen af selve programmet ad flere omgange. Men hvordan ser iterativ spiludvikling ud, set fra et elevperspektiv?

I Coding Class projektet har eleverne arbejdet i grupper med om spiludviklingen og programmeringen. En af de interviewede elevgrupper fra en af de skoler, der ikke er uddybet ovenfor, er meget bevidste om, at de via gruppens iterative arbejdsprocesser har fundet frem til en inkrementel opbygning af deres program, hvilket betyder, at programmet gradvist vokser og er opbygget i små sammenhængende trin. For hver ændring eleverne laver, tester de, hvordan det virker og vurderer resultatet. Gruppen arbejder på spillet "Vault Boy Adventures"[39], der er inspireret af spillet Fallout, hvori hovedpersonen har et device - en såkaldt "pip-boy" (Personal Information Processor - PIP - se billedet). Pip-boy er en digital figur i spillet og han eksisterer i et grafisk univers, der er grøn-sort og pixeleret - ifølge eleverne er stilen "retro".

[37] At skrive kode betragter vi som en del af programmeringsarbejdet. Det at programmere rummer kodning, men fx også debugging aktiviteter. Eleverne skriver og arbejder blandt andet med kode, når de programmerer - laver et program, udvikler et spil.
[38] Computationelle artefakter er her de produkter, der baserer sig på computerteknologi.
[39] Se projektet her: https://scratch.mit.edu/projects/146381789/

Fra Fallout 4 - Pip-boy eller Vault Boy

Eleverne har taget udgangspunkt i at lave en “endless runner” dvs et spil, hvor avataren - her den grønne mand - ser ud til at løbe fra venstre mod højre, indtil spilleren mister sit liv, ofte kan avataren skyde noget undervejs, samle point og skal undgå forhindringer. I elevernes spil skal Pip-boy hoppe på ’bjælker’ og skyde mod kakerlakker, hvilket giver flotte eksplosioner (se billedet). De fortæller, at man styrer avataren på z - ligesom på gameboy spilkonsollen.

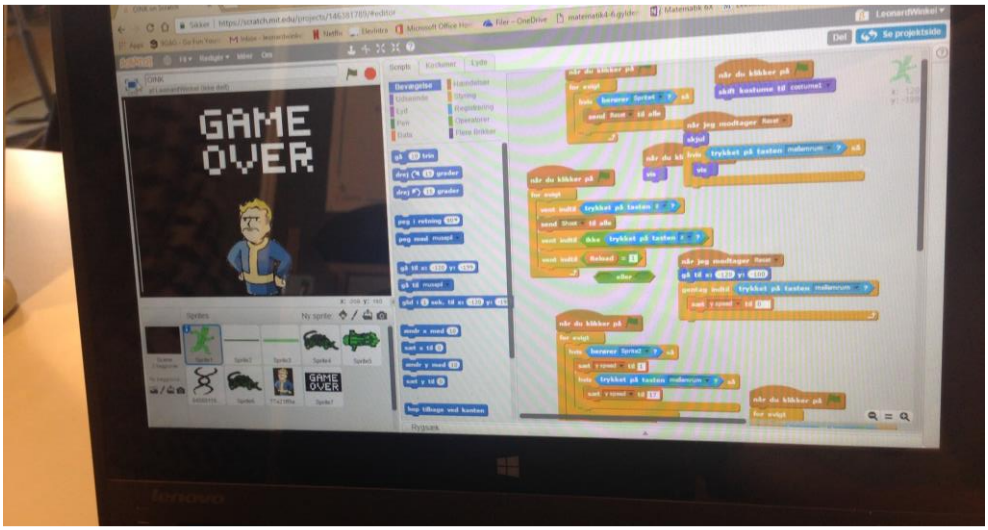


Eleverne har programmeret avataren til at kunne bevæge sig langs y-aksen (hoppe mellem bjælker) og skyde ved tastetryk. De har programmeret bjælker og kakerlakker til at komme ind på skærmen løbende langs x-aksen. Når en kakerlak rammes af et skud, eksploderer den. Når avataren rammes af en kakerlak, dør man.

Eleverne forklarer, at de er begyndt med at bygge “basics op” (som fx at få avataren til at flytte sig på skærmen). De har derefter bygget videre på spillet, gjort det sværere og arbejdet

med grafikken, med flere slags våben og power-ups til sidst. De mener selv, at de er nået langt på grund af denne inkrementelle arbejdsmåde. For at kunne arbejde på denne måde har gruppen skabt et overblik over, hvad der udgør et runner-spil, hvilke elementer, der er centrale, og de har bygget spillet op omkring disse grundfunktioner. Via dette arbejde har eleverne identificeret generelle spilfunktioner, og de har arbejdet med at abstrahere, det vil sige forstå visse egenskaber ved et spil, isolere det til en funktion og se bort fra andre. For så at kunne forstå denne delfunktion i én kontekst og oversætte (programmere) dets grundkoncept til en anden.

Endvidere forudsætter elevernes arbejde med at opbygge spillet, at eleverne kan nedbryde relevante problemstillinger til mindre dele (dekomposition). Det gælder, når de skal finde ud af, hvordan en kompleks hændelse nedbrydes i programdele, såsom: Avataren skyder og kakerlakken eksploderer, spilleren får point og eksplosionen forsvinder. Disse dele skal sættes sammen til en meningsfuld sammenhæng for computeren, så hændelserne sker, hver gang de rette betingelser er til stede (algoritmer). Eleverne arbejder derfor også med at have overblik over de enkelte elementer og deres gensidige påvirkning og sammenhænge.

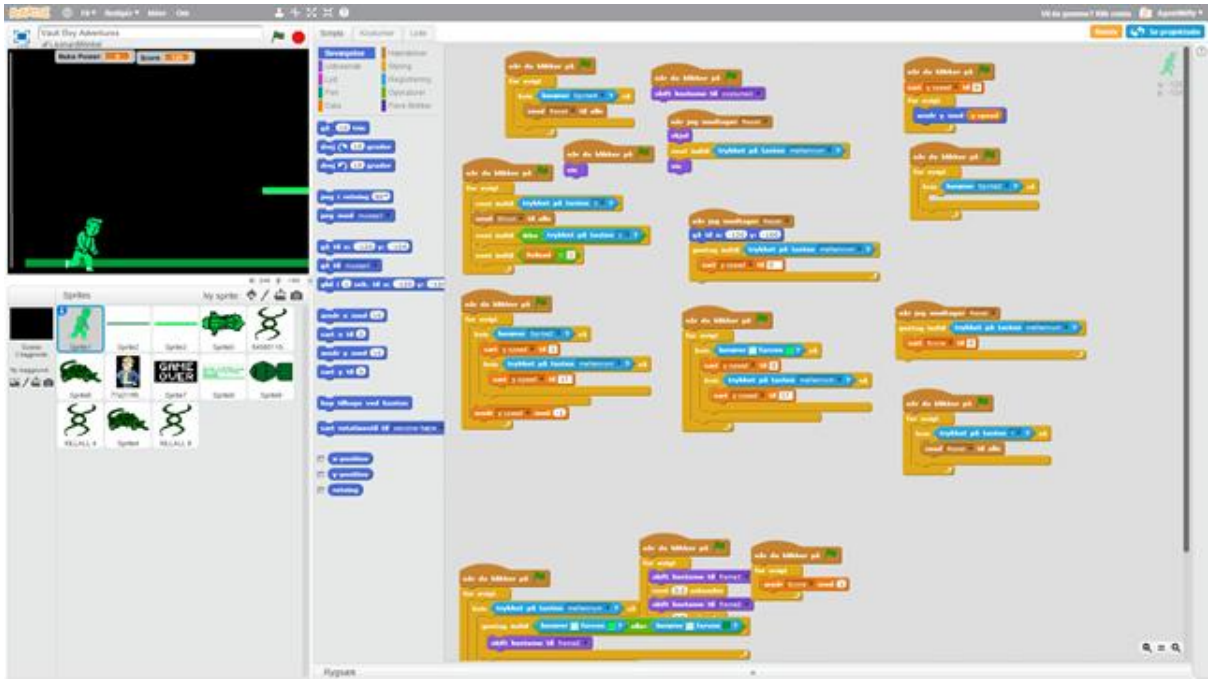


Her ses kode knyttet til ét af elementerne i PiP-boy spillet

Flere elever (ikke bare i denne gruppe) anvender en puslespilsmetafor om det at tænke i sammenhænge, fx:

Man skal være god til at have et overblik og man skal være god til at sammensætte ting. Det er lidt som et puslespil, det hele skal gå op i hinanden.

Denne metafor bruges også visuelt fx når man arbejder med spiludvikling via Scratch, hvor eleverne sammensætter koder ved at sætte blokke sammen - blokprogrammering. Her kan de teste deres hypoteser ved at sætte elementer sammen og køre koden for at observere konsekvensen.



Eksempel fra programmeringsmiljøet Scratch, der viser, hvordan blokprogrammeringen visuelt bygger på en puslespilsmetafor. Her vises koden knyttet til avataren i Vault Boy Adventures.

Elevernes logik kommer også på prøve, når der opstår fejl i koden. Dette fremhæver flere elever fra de skoler, vi har besøgt, som særligt svært. Eksempelvis svarer en af eleverne bag *Vault Boy* spillet på spørgsmålet om, hvad der er sværest ved programmering:

> *Elev: Der er bugs i scratch. Nogle gange virker det ikke. Fx så vender min kakerlak nogle gange på hovedet. Og jeg har gået igennem det, men det er ikke til at finde fejl i scripts*
>
> *Interviewer: Så det er svært at fejlfinde*
>
> *Elev: ja, for man tror, at man har lavet en fejl, når der er bugs. Og måske tror man, det er bugs, når man har lavet en fejl.*

Fejlsøgning kræver en dybere forståelse af afhængighederne mellem de forskellige funktioner, variable, operatorer mv, altså programmeringssprogets syntax, og det kræver udholdenhed og systematik. Netop denne del af kodearbejdet kan virke demotiverende på nogle af eleverne.

Problemløsning, hjælp og samarbejde

I vores interview med eleverne og observationer af deres arbejde med iteration og inkrementel udvikling af spillene, har vi fået indblik i elevernes og lærernes forskellige måder at håndtere især arbejdet med de-bugging og problemløsning. Et andet eksempel på en elevgruppes måde at organisere sig omkring problemløsning er en gruppe fra Skole M, hvor det ene gruppemedlem (en dreng) identificeres af de andre elever som særligt stærk i kodning og programmering - allerede før projektet:

Elev 1: det der Play Canvas, der skulle man selv skrive koderne. Alle de der engelske koder.

Elev 2: det var jeg den eneste, der kunne

Elev 1: hvis man glemte et stort bogstav eller punktum, så gik det hele i udu.

Interviewer: er det så kun dig Elev 2, der har siddet med det?

Elev 1: nej, for Elev 2 er sådan rigtig god til at huske at lære os andre det.

Elev 3: han lærer os

Elev 1: han er god til at sige, hey kom lige herover. Nogle gange viser han. Nogle gange spurgte han mig om hjælp.

Vi har set flere eksempler på elevgrupper, hvor der er en eller et par af eleverne som på forhånd er meget interesserede i kodnings- og programmeringsarbejdet. Fx var der ved Skole K en gruppe, hvor to af pigerne i gruppen havde haft medievalgfag året før. De to piger havde derfor allerede prøvet at kode og programmere i et andet program end Scratch. Håndteringen af den situation, at der kan være en eller flere elever i en gruppe med særlige it-interesser og -forudsætninger udfolder sig imidlertid meget forskelligt i grupperne. I ovenstående dialog med elevgruppen fra Skole M bliver det tydeligt, at Elev 2 er meget interesseret i og god til at kode og programmere, og han er samtidig god til at medinddrage de andre gruppemedlemmer, så gruppen lærer og udvikler i fællesskab (kollaborativt).

Gruppesamarbejde er en krævende proces, og den kan være lærerig. Men læringspotentialerne i gruppesamarbejde afhænger i høj grad af i hvilken udstrækning grupperne formår at samarbejde kollaborativt. Som man ofte ser ved gruppesamarbejde, så har vi også i forbindelse med Coding Class aktiviteterne set at nogle elevgrupper samarbejder mere kooperativt, hvor de fordeler opgaver imellem sig, således at nogle elever i processen programmerer meget, hvor andre tegner, finder på, skriver tekster mv. Alle disse aktiviteter er en del af den kreative proces og dermed vigtige elementer, men elevernes læring i gruppen afhænger i høj grad af, hvilke dimensioner af arbejdet den enkelte elev kommer i berøring med.

Det er, som altid i gruppearbejde, en udfordring at skabe grupper, hvor eleverne supplerer hinanden i forhold til kompetencer og samtidig beriger hinanden. Vi så i nogle grupper, at elever, der oplevede at have færre kompetencer, overlod programmeringsarbejdet til dem, der turde mere eller havde mere erfaring. Vi har også observeret elever, der i høj grad udveksler viden undervejs i produktionsprocesserne. Det er dog forskelligt, i hvor høj grad de har en kultur for at hjælpe hinanden, og hvor gode de er til at hjælpe hinanden frem for fx at løse problemerne for hinanden. En elevgruppe fra Skole K fortæller om, hvordan de har hentet hjælp hos en anden elev i klassen, men uden selv at blive klogere på løsningen:

Interviewer: er der nogle ting, der ikke har fungeret, som de skulle…

Elev 1: ja-ja… solen og at få point… at ramme stjernerne. Så ville den ikke tælle point, og så ville vi sætte sådan noget lyd ind, og det ville den også helt forkert.

Elev 2: ja, hvis man fx ramte en stjerne og så kom lyden ud et halvt minut efter, og det lød bare dumt. [griner]

Interviewer: hvordan har I så løst de udfordringer?

Elev 1: ja, hvordan løste vi dem?

...

Elev 3: vi spurgte Sofia, som havde fået det ordnet. Hun havde et spil, der mindede en lille smule om det. Jeg gik hen til hende og spurgte om hun kunne gøre noget, og så fik jeg det tilbage igen, og så virkede det. [griner]

Interviewer: okay, så I ved ikke, hvad hun lavede?

Pigerne: nej

Udover at hjælpe hinanden, så handler problemløsning også om at etablere en klassekultur for at hente hjælp til løsning af problemer. Eleverne i 5. klasse fra Skole M og Leo taler samstemmigt om, hvordan de arbejder med problemløsning i klassen:

Elev 1: man skal ville det og interessere sig for det. Man skal være tålmodig.

M: kan man hente hjælp nogen steder?

Elev 2: YouTube

Elev 3: og man kan også spørge sine klassekammerater om hjælp.

*Elev 1: man kan også altid undersøge på Google – Leo siger, at Google har svaret på alt. *griner**

Leo og eleverne i 5. klasse er enige om, at den sidste man spørger, er læreren. Først gør man selv et forsøg på at finde løsning på det problem, man sidder med, og hertil kan det være en fordel at søge svar på internettet, så benytter man sig af forskellige livliner i klassen. Til sidst går man til læreren. Her er der tale om en meget deltagende, kollaborativ og netværksbaseret tilgang til arbejdet med problemløsning, fejlfinding og samarbejde.

Coding Class instruktørerne har introduceret eleverne for instruktionsvideoer, som en anden indgang til at hente hjælp til at forstå, hvordan man koder og programmerer fx via Scratch. Både elever og lærere refererer til instruktionsvideoer som en meget stor hjælp og en god måde at lære programmet at kende indledningsvist.

Som Brennan og Resnick (2012) peger på, så må evalueringen af elevernes læring tage afsæt i samtaler med eleverne om deres refleksioner over eget arbejde og udvikling. Det er ikke nok at kigge på elevproduktet, idet det ikke giver en indikation af den enkelte elevs læring. Via ovenstående samtaler, får vi et indblik i, hvor væsentligt det er at tydeliggøre og arbejde med at udvikle en særlig samarbejdskultur for arbejdet med elevernes design-baserede læreprocesser og computationelle tænkning.

Læring i fællesskab

Det er et væsentligt aspekt af den deltagende tilgang til computationel tænkning, som blandt andet Brennan og Resnick repræsenterer, at elever og lærere agerer som et lærende fællesskab fx i arbejdet med problemløsning. At agere som et lærende fællesskab betyder, at de traditionelle skolehierarkier mellem lærere og elever overskrides. Læring i fællesskab kan både finde sted i undervisningen og på tværs af undervisningen i skolen, men også udenfor skolen. Når eleverne medinddrages som kompetente aktører i undervisningsaktiviteter kan både de elever, der får og påtager sig en vejledende rolle, og de elever der bliver vejledt, blive styrkede. Vi har set flere forskellige måder at medinddrage elever som kompetente ressourcer på skolerne i Coding Class aktiviteterne. Netop denne medinddragelse og oplevelsen af at deltage i reelt lærende fællesskaber er ifølge de involverede lærere og elever lærende, motiverende og engagerende. Fx har 6. klasserne fra Skole K efterfølgende undervist 5. klasserne i spiludvikling, programmering og kodning via Scratch. Vi har allerede nævnt, hvordan elever fra Leos klasse på Skole M har undervist PLC vejledere fra Syddanmark. På Skole M har de også oprettet et Google dokument, hvor lærere kan ønske appudvikling til deres fag, og så kan eleverne fra skolen byde ind på opgaverne. To af eleverne fra Leos klasse fortæller, hvordan de på eget initiativ udvikler spil til både yngre søskende og skolens lærere i fritiden.

I de fleste tilfælde ser vi, at det med at trække på elevernes ressourcer og agere et kollektivt lærende fællesskab placeres i bestemte tidsrum i relation til bestemte aktiviteter (fx en enkeltstående dag) og organiseringer (fx mediepatruljer). For Leo er det et pædagogisk greb, der gennemsyrer al hans undervisning.

> *Elev 1: de fleste timer bruger vi til at kode og arbejde med vores firmamatematik.*
>
> *Elev 3: Det gør vi også i natur og teknik.*
>
> *Interviewer: taler I om matematik?*
>
> *Elever: hmm*
>
> *Interviewer: eller lærer I det bare ved at kode og programmere?*
>
> *Elever: altså…. Ja..*
>
> *Elev 3: ja det gør vi så også*
>
> *Elev 1: men i grundbogen har han valgt nogle sider, hvor vi så lærer meget ved bare at lave et par opgaver af dem.*
>
> *Elev 2: Leo gider ikke, at vi skal lave hele bogen. Hvis han finder noget, han synes er spændende, så tager han det.*
>
> *Elev 1: I natur og teknologi, der har vi også bygget nogle robotter, hvor man skal sætte en motor ind og få dem til at køre. Det er også rigtig rigtig sjovt.*
>
> *Interviewer: har I også en bog der?*

Elever: nej

Elev 1: han siger bare, hent nogle batterier, ledning og en motor, så er I igang. Vi skulle engang lave en robot ud af en kop.

Elev 2: han sagde, at vi fik et batteri, to ledninger, en motor og en kop. Så måtte vi finde ud af.

Elev 1: og nu har vi så en gang mere til at bygge den færdig. Så skal vi ud og lave kapløb om, hvem der kommer først af dem. Vi skal så se, hvad for en gruppe, der har programmeret den bedst.

...

Elev 1: …. De fleste opgaver vi får af Leo, de er, hvor vi bare skal prøve os frem. Vi får ikke nogen instruktioner eller noget. Det er ikke særligt tit, vi får en opgave, hvor det er lige præcis DET, man skal gøre. Det er altid bare "prøv dig frem".

Elev 2: Leo siger, I får lov til at arbejde med det her i 30 minutter. I skal prøve jer frem. Leo har arbejdet med det i fem minutter. Så prøver vi ting af, så han ikke selv skal gøre det derhjemme.

Elev 3: Så får han os til at gøre det for ham

Leo forklarer selv sin tilgang med følgende ord:

Leo. Det med at være kreativ og skabende gennemsyrer al undervisningen [Leo's undervisning]. Det er lettere at gøre det aktuelt for eleverne, når man arbejder med, at de skal være skabende, i stedet for at de skal løse en eller anden opgave, hvor der står alle mulige faste ting i forvejen. Det var noget af det, som jeg… man måtte kunne åbne op for opgaverne. Så de blev nødt til at have sig en viden for at kunne løse opgaven. Tegn en grund på 275 kvm. Den må ikke have rette vinkler. Så er de nødt til at gå ind og kigge på, hvordan arbejder man med parallelogram eller en trekant eller en cirkel. Og her stiller jeg en formel op, der løser den ligning. Det ville vi nok bruge computeren til i dag. Det gjorde vi ikke til at starte med. [5m]

De arbejdsmåder, som eleverne og Leo beretter om og praktiserer, repræsenterer i høj grad arbejdsmåder og deltagelsesformer, der knytter an til arbejdet med computationel tænkning i lærende fællesskaber. Dette arbejde handler om at få styr på nogle begreber fx "scripts", men det mest væsentlige er at elever og lærere arbejder med de praksisser, der knytter sig til det at arbejde computationelt tænkende. Disse deltagende praksisser læres ikke, hvis afsættet for undervisningen fx er de traditionelle hierarkiske strukturer mellem lærere og elever, hvor lærerne er de vidende og kompetente, mens eleverne er dem, der skal instrueres. De deltagende computationelle undervisningspraksisser er i høj grad beslægtede med det, vi kender fra kollaborativt problemorienteret gruppeprojektarbejde, hvor lærere og elever samarbejder om elevernes interessedrevne projektidéer, aktiviteter tager afsæt i reelle problemstillinger, arbejdet med at indkredse problemstillinger, og måder at tænke kreativt med at skabe viden. I Coding Class projektet har vi imidlertid iagttaget, at det langt fra er intuitivt for alle lærere og elever at engagere sig i disse fælles eksplorative praksisser. Vi har iagttaget lærere, der har svært ved at stille udforskende spørgsmål til eleverne, og lærere, der hurtigt overtager problemløsningen for eleverne, snarere end at guide dem i

teknikker til at kunne problemløse. Årsager til disse praksisser kan findes i såvel lærerens tilgang til undervisningen, egen rolle i undervisningen, og helt konkret om de har kompetencerne til at engagere sig i praksisser såsom fælles eksploration af fejlfinding i et spil, fælles idegenerering mm.

Undervisning med fokus på design-baserede læreprocesser og computationel tænkning ud fra det deltagende perspektiv forudsætter, at elever og lærere engagerer sig i det, vi vil kalde for 'indirekte undervisning', fordi elever og lærere lærer i fællesskab med de problemer, der opstår. Den helt store udfordring med denne form for undervisning er, at den forudsætter meget viden, der kan trækkes på ad hoc hos læreren, som løbende skal understøtte elevernes læreprocesser ved at forbinde elevarbejdet med fagsprog og –mål. Herudover viser eksemplerne fra Skole M, at denne fælles eksplorative form for undervisning helt grundlæggende anerkender elevernes kreativitet og eleverne som ressourcefyldte bidragydere i den fælles læring i klassen. Leo arbejder hele tiden selv med at blive dygtigere til at kende og bruge teknologi i undervisningen, men han gør det i høj grad i fællesskab med eleverne, og han opsøger den særlige form for læring om brug af teknologi i undervisningen, som han skaber sammen med eleverne.

Vi vil kalde Leos tilgang for en *empowerment tilgang*, hvor det væsentlige er at kunne åbne for meningsfuld og relevant læring, der viser sig at være behov for i forløbet, snarere end at fokusere på fx "i dag skal vi alle lære om loops". Leo er i Coding Class regi en særlig lærer, og hans 5. klasse står frem som en særlig klasse. De bliver af gode grunde fremhævet som særlige i Coding Class projektet, både af Coding Class instruktører og den kommunale Coding Class vejleder. Men som vi har forsøgt at tydeliggøre her, så er Leo og hans klasse ikke repræsentative for de mange lærere og elever i Coding Class projektet. For de fleste er både arbejdet med spiludvikling, kodning, programmering og computationelle tænkepraksisser et nyt domæne, der - som Angeli et al. (2016) peger på - mangler at klargøre, hvilke teknologiske, pædagogiske, og indholdsmæssige kompetencer, praksisser og begrebsliggørelser, der er nødvendige for, at elever og lærere kan udfolde læringspotentialerne i praksis. Her ser vi således et markant potentiale for videreudvikling af undervisningspraksisser for undervisning med fokus på computationel tænkning.

Informatik i matematik og vice versa

Med reference til Caspersen (2017) har vi tidligere nævnt, at det både er relevant at tale om informatik som fag og informatik i fag. Informatik i fag har som tidligere påpeget ikke været i særlig grad i fokus i Coding Class aktiviteterne. Men især i forhold til matematik, har aktiviteterne aktualiseret det at kode og programmere samt udvikle spil, som en vej til at få et andet fokus på fag. Når det særligt er matematik, der træder frem, er det netop fordi, en del af lærerne i Coding Class projektet har været matematiklærere.

Når eleverne programmerer, så skaber de produkter i matematiske rum. Det vil derfor ofte være en hjælp at have styr på matematiske størrelser som variable, koordinatsystem, operatorer og geometri, for eksempel når man skal forstå, hvorfor kakerlakken, som tidligere nævnt, drejer om egen akse. En deltagende matematiklærer betoner dog, at det er systematikken og problemløsning, der især trænes igennem programmering:

Sådan noget drilsk matematik, hvor man ikke bare skal tænke vanetænkning

Derudover trænes algoritmisk tænkning i programmering. Eksempelvis arbejder pip-boy programmørerne med en betingelse, når kakerlakken eksploderer: Hvis kakerlakken rammes af ammunitionen, så skal den "skifte kostume" til grafikken "eksplosion".

Algoritmisk tænkning og indkapsling af delproblemer er ligeledes kompetencer, der er påkrævet i matematik. En lærer udtrykker:

> *...hele 'hvis så' tanken går videre i matematik, hvis man skal løse en større problemstilling, så er du også nødt til at pille den ud i tre – fire – fem enkelte elementer og sige: "nå men, hvis jeg lægger de her to enkelte elementer sammen og så ganger den med den der, så må det betyde, at jeg har fået resultatet for det der".*

Eleverne bliver altså med programmering udfordret på at nedbryde problemstillinger i enkelte dele og behandle dem hver for sig (se også Ejsing-Duun & Misfeldt, 2016).

Vi har endvidere set, at elever igennem programmering kan få øjnene op for, at matematik ikke blot kan bruges til at løse problemer - og få et facit - men også til at skabe med. Programmering kan altså forbindes med kompetencer, der relaterer til anvendt matematik.

Faglig læring og skabende tilgang til it

Lærere og elever peger på faglige potentialer i forløbene, der relaterer til elevernes skabende arbejde med it. Leo fra Skole M har inddraget mange forskellige produktions- og repræsentationsteknologier i sin undervisning. Da vi besøger Leo og Coding Class klassen, er eleverne ved at gøre klar til at kunne præsentere deres virksomheder og spiludviklingsarbejde for en handelsskole i Løvens Hule, hvor eleverne får respons på deres arbejde. Som en del af forberedelsen skal eleverne forberede en blog eller hjemmeside, hvor de linker til deres produkter. Leo har en liste med eksempler på de produktions- og repræsentationsteknologier eleverne kan arbejde med:

Tinytap spil (programmeringsmiljø)
Sploder (programmeringsmiljø)
Kodu (programmeringsmiljø)
Scratch (programmeringsmiljø)
Powtoon (udviklings af animerede videoer)
Tinkercad (3D modellering til merchandise)
Inkscape (grafikprogram)
Stikbot (udvikling af stop-motion videoer)

Som det fremgår af listen, så kan Leos Coding Class klasse vandre hjemmevant i og på tværs af mange forskellige teknologier, som de kan agere skabende og formidlende med. Gruppen Electro's hjemmeside er illustrativ for, hvordan Leo og eleverne har koblet arbejdet med de mange forskellige programmer, og det kreative og skabende arbejde med it i Coding Class aktiviteterne.

Gruppen Electro har lavet en hjemmeside, der præsenterer deres virksomhed og en blog, hvor de blandt andet linker til forskellige spil, gruppen har udviklet som en del af Coding Class aktiviteterne.

På hjemmesiden præsenterer eleverne de programmer, som de anvender:

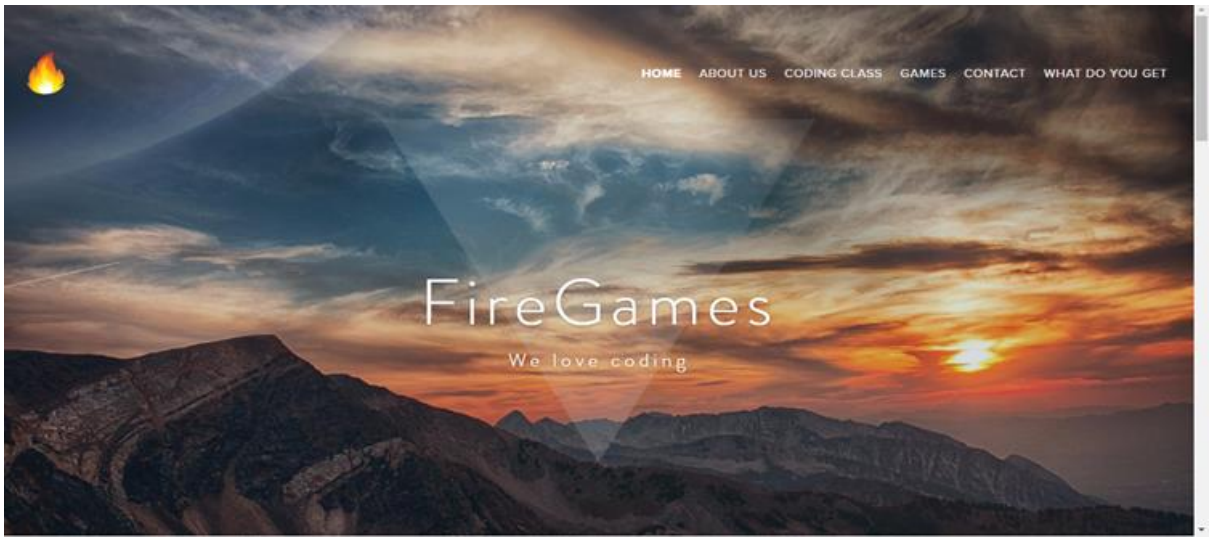
Leo forklarer, at han har mange forskellige faglige mål knyttet til sin undervisning:

> *Jeg synes, der er flere matematik-mål at nå, end der er natur- og teknikmål. Og endnu flere danskmål. Jeg blev interviewet på et tidspunkt af en, der spurgte, hvilke mål jeg lagde ind i det, og så, når jeg kiggede i Min Uddannelse, så kunne jeg se, at der var flest danskmål. Så det ved jeg bare, at der er i mit forløb. Men jeg har også planlagt det i tre år. Bl.a. det med at*

lave hjemmesider, det ligger inde som en danskfaglig del. Variabler og sådan noget, det er selvfølgelig matematik. Teknologi og når vi bygger robotter også sådan noget, så er det nok mere natur og teknologi. Selvom vi har mange variable med også der. …

Men underviser jo også i engelsk uden at have engelsk. Mange af programmerne er engelsksprogede, så der skal de bare ind og læse på engelsk. Det er jo bare en sidegevinst i mit hoved.

Leo identificerer en række fag, som han kan se blive bragt i spil i den skabende og kreative undervisning med it. Et eksempel på, hvordan engelsk kan blive bragt i spil uden for engelskundervisningen, er gruppen Firegames. Denne gruppe har selv valgt at skrive hjemmesiden på engelsk.



Det ligger, som allerede nævnt, ikke inden for dokumentations- og evalueringsprojektets rammer at vurdere elevernes produkter og læring i arbejdet med produkterne. Når vi refererer til disse gruppers produkter, så gør vi det illustrativt for at beskrive, hvordan elever og lærere ser muligheder for at arbejde med koblinger mellem fag, og det at arbejde skabende og kreativt med it i Coding Class aktiviteterne.

For Leos Coding Class klasse har det ikke været nyt at arbejde kreativt og skabende med it. Det gjorde de allerede. Men det var nyt for dem, da de fik Leo som lærer. To elever fra Leos klasse udtaler:

Elev 1: Jeg synes, det har gjort en kæmpe forskel. Før var det eneste jeg kunne finde ud af at bruge skærmen til, det var at sætte mig ned og se på et eller andet. Blive underholdt af det. Jeg kunne ikke selv. Jeg tænkte aldrig over det der med, at jeg selv kunne producere ting, som andre kunne bruge.

Elev 2: Jeg tænker meget det samme som Elev 1. Jeg kan ikke tegne. Så kan man lige hente et program til sin iPad, og så kan man tegne. Det er meget nemmere. Og man laver spil til andre og andre kan spille det.

Flere elever og lærere betoner kreativiteten og den skabende tilgang til it som noget af det de har lært i Coding Class forløbet. Men vi ser stor variation i, hvor tydelig erkendelsen er for

eleverne. For eksempel forklarer en elev fra Skole K:

> *Elev A: skabende? At internettet og Google drev ikke bare er, at man kan skrive, men at man faktisk kan lidt mere med det.*

Elev A er fra en elevgruppe, der består af fodbolddrenge uden særlig interesse for it. Drengene forklarer også, at de ikke er 'gamers' ligesom nogle af de andre elever i klassen, der er rigtigt gode til at programmere og kode. For denne elevgruppe har det været svært at finde engagement, og de har været udfordret på, at de fik slået et alt for komplekst spiludviklingsprojekt op. Til trods for at drengene ikke havde den store interesse og succes med deres projekt, så har Coding Class aktiviteterne alligevel åbnet for nye erkendelser for det udfordrende ved at skabe med it fx når man udvikler spil:

> *Elev A: det er meget sjovt at lave sit eget spil, tænker man. Men det er meget svært. Meget sværere end man tror.*
>
> *Elev B: man skal ligesom ind på bagsiden af det, før man sådan finder ud af det. Nogle siger, at det er easy, vi skal bare lave vores eget spil. Men det er det faktisk ikke, for man skal LAVE det og ikke bare kigge på det.*
>
> *Elev C: tænk på dem, der har lavet FIFA. Det må være svært med publikum der jubler, og de rammer jorden altid.*

For mange elever i projektet, har Coding Class givet anledning til at bruge computere til andet end at konsumere musik, film og spil, nemlig til at skabe digitale produkter herunder spil og programmering af robotadfærd. Via de skabende og kreative Coding Class aktiviteter opnår eleverne en anden adgang til erkendelse af, at de kan bruge computere til at udtrykke sig med. Pip-boy udviklerne brugte eksempelvis meget tid på at finde billeder på nettet og modificere dem i programmer for at tilpasse dem den grafiske stil (farve billederne grøn/sorte, pixel for pixel, samt ændre på på Pip-boys udtryk). De brugte også deres genrekendskab og det repertoire de har inden for spil til selv at skabe et spil. Deres designpraksis tager således udgangspunkt i genbrug af materialer, som de finder (et andet spil i Scratch, billeder på nettet, genren 'runner', narrativet om Pip-boy) og remixer til deres eget projekt.

I det ovenstående citat betoner Elev 2 også det særlige i, at eleverne ikke skaber for at skabe - men skaber til nogle, og Elev C peger på, at det de konsumerer, er skabt af nogen. Derved aktualiseres elevernes kritiske blik på egne og andres produktioner, hvilket kan fremme elevernes forståelse for kommunikative processer,skærpe de kreative processer, og forståelsen for det bagvedliggende - som Elev B udtrykker det.

Kreativiteten handler også om at skabe løsninger på komplekse problemer, som eleverne selv skal lære at rammesætte. Det er både en udfordring mange har værdsat, men også har virket overrumplende på nogle. En observation er, at når elever og lærere arbejder med programmering, må de gøre op med facitkulturen, som en af Coding Class instruktørerne fortæller:

> *Sådan er kodning også – der er ikke én løsning der er den rigtige – hvis du skal kode en figur til at hoppe, så kan du gøre det på 1000 måde*r.

Det er imidlertid ikke alle elever, der finder det let at agerere inden for disse friere rammer, hvor der ikke er eksakte løsninger og en opgave ikke nødvendigvis er færdig, fordi den er lavet. Produktet kan gøres bedre og fordybelse er påkrævet for at kunne løse vanskelige udfordringer. En elev fremhæver, at vedholdenhed derfor er en central kompetence for at lære at kode:

> *...hvis man koder, skal man [...] ikke give op. Hvis man giver op og ikke har en fuldstændig instruktionsbog, så går det ikke. Man skal fortsætte og ikke give op. Det er noget af det, man skal være god til for at kunne kode. Man skal kunne holde ud, at der er noget, der ikke går, som det skal.*

Kode og afkode: Sproglige kompetencer

Programmering er at skrive en forskrift til computere på et sprog, de forstår. Det er et skabende sprog med sit eget vokabular, sin egen grammatik og udsigelseskraft. Som Brennan og Resnick (2012) fremhæver, så knytter der sig nogle computationelle begreber til computationel tænkning.

Eleverne betoner vigtigheden af at forstå de enkelte sprogs syntax (strukturen for udsagn) for at skabe en betydningssammenhæng, der udmønter sig i noget computeren kan forstå. Computerne tillader ikke nogen særligt stor fejlmargin, for glemmer man et *"bogstav eller punktum, så gik det hele i udu".* Dette gælder for tekstprogrammering, men også ved blokprogrammering er overholdelse af blokkenes rette sammenhæng en vigtig del af det at skabe et program, computeren kan forstå.

Eleverne skal også lære at nedbryde og formulere deres problemstillinger i et sprog, som computeren forstår. Således erkendte elever fra Skole K, at koden både kan skrives og 'læses' som en meningsfyldt sammenhæng:

> *Hvis man skal have noget til at gentage sig selv, så skal man indsætte sådan en "for evigt" klods. Det er også lidt svært at finde ud af, hvornår man skal bruge den. Så havde vi også lige hende læreren Mette, der forklarede det og læste det tydeligt op for os. Hvad der stod på de der klodser. Når du gør det… så for evigt… hvis den rører farven sort, så hopper den tilbage for eksempel. Så man skal også lige forstå det lidt. Sætte sig tilbage og have tid til at tænke over det og kunne læse det for sig selv.*

Eleven beskriver her, hvordan hun lærte at afkode sin kode ved at læse og reflektere over den. Skal eleven afkode koden, kræver det en dybere forståelse af, hvad koden kan, og hvornår den kan bruges meningsfuldt. Det er netop svært at verbalisere sammenhængene i programmeringen, hvorfor det kan være svært for de dygtigere elever at hjælpe deres kammerater. En lærer forklarer:

> *...børnehjælpere er fint, men til en hvis grænse. Der er også i den anden klasse, hvor der er en [...], der er vildt dygtig til det. Han har siddet og forklaret, hvad han gjorde, men han har bygget et vildt kompliceret spil op. Og så spørger jeg: hvorfor er det ham, der har lavet det – så siger de: ja men han forklarede alt, hvad han gjorde undervejs – men nu kan vi bare ikke huske det – nu kan vi ikke forstå det længere.*

I dette tilfælde har eleverne endnu ikke en kultur for og sprog til at kunne hjælpe og støtte hinanden meningsfuldt. Mange elever har været mere optaget af, at deres spil skulle lykkes, end den læring, de opnår. Som tidligere nævnt peger Brennan og Resnick (2012) på, at elevernes projekter (produkterne) ikke nødvendigvis i sig selv er repræsentative for elevernes læring. Forfatterne understreger, at evaluering af elevernes læring og arbejde med computationel tænkning derfor bør funderes i (løbende) samtaler med og mellem eleverne om deres læring og arbejdet med forskellige begreber, praksisser og perspektiver.

Ovennævnte eksempel, hvor læreren har talt om og læst koderne sammen med eleverne, er en praksis med potentialer, som vi har set, men den har ikke været udtalt i projektet. Vi ser et uudnyttet potentiale i at skabe rum for, at eleverne og lærerne undervejs taler og læser kode sammen for at øve forståelse af programmering som betydningssammenhæng. Det kan gøres som en del af introduktionsforløbet, men også i forbindelse med feedback og feedforward sessioner mellem elever undervejs i forløbet med fokus på koden. Vi har observeret, at lærere har arrangeret en feedbacksession, hvor eleverne øvede sig i at præsentere deres idéer og fik respons på deres arbejde i forløbet, men disse sessioner har ikke haft fokus på elevernes læsning og skrivning af kode.

Flere af de elever vi har talt med, har ikke oplevet, at de har lært matematik i Coding Class aktiviteterne. De lærere vi har talt med udtrykker ligeledes at det matematikfaglige ikke har været eksplicit i fokus. Via samtaler med eleverne kan vi se, at det at arbejde med Scratch i sig selv ikke nødvendigvis er understøttende for elevernes matematiske forståelse af fx koordinatsystem og x- og y-akser. Det er nødvendigt eksplicit at etablere de faglige relationer. I den forbindelse ser vi uudnyttede potentialer i også at tale om, hvordan det at arbejde med kode og læse kode også hænger sammen med matematiske begreber.

Kode og afkode: Teknologiforståelse og dannelse

Arbejdet med informatik, computationel tænkning, design-baserede læreprocesser og elevernes skabende og kreative it-kompetencer handler i høj grad om arbejdet med elevernes teknologiforståelse. Teknologiforståelse handler både om, hvad it kan bruges til, men også om en forståelse af, hvordan de programmer, som elever selv bruger, er skabt. Arbejdet med programmering kan rumme en dybere forståelse for de logikker (årsag-virknings-sammenhænge), som programmerne opererer efter. I eksemplet ovenfor med elevgruppen fra Skole K, der erfarede, at de fik indblik i “bagsiden af spillet”, ses at eleverne foretager en ny læsning af velkendte produkter (FIFA), som de er forbrugere af. Denne nye læsning vidner om et nyt refleksionsniveau og anderledes forståelse i relation til teknologi. Udover at opnå konkrete færdigheder i at håndtere programmer, så opnår eleverne via arbejdet med spiludvikling et nyt kendskab til deres egne muligheder for at arbejde computationelt. Eleverne opnår nye muligheder for at identificere sig som skabere af computationelle artefakter.

Denne dimension styrkes yderligt i Coding Class projektet for de klasser, der har haft et samarbejde med en it-virksomhed om at præsentere og få respons på deres spiludvikling i

slutningen af projektet. I Coding Class aktiviteterne er elevernes nysgerrighed på it, spiludvikling, programmering og kodning, blevet stimuleret både via virksomhedsbesøg og samarbejde med Coding Class instruktørerne, som har bragt særlige it-specialistkompetencer ind i klasserne.

En lærer understreger også elevernes teknologiske dannelse som en central faktor og et væsentligt argument for at have et it-fag i folkeskolen. Læreren argumenterer for, at eleverne lærer "hvad der ligger bagved" systemerne, hvilket gør det muligt for dem at deltage i demokratiske beslutninger, og det er en kerneopgave for folkeskolen at gøre eleverne i stand til dette. Læreren peger endvidere på, at vi ikke ved, hvordan programmering ser ud om 10 år, hvilket betyder, at vi ikke skal arbejde målrettet mod at give eleverne programmeringsfærdigheder, men snarere kompetencer inden for computationel tænkning. Dette synspunkt stemmer overens med Kafais udsagn, som vi tidligere har citeret: "*Today, reading code is about reading the world. It is needed to understand, change, and remake the digital world in which we live*."

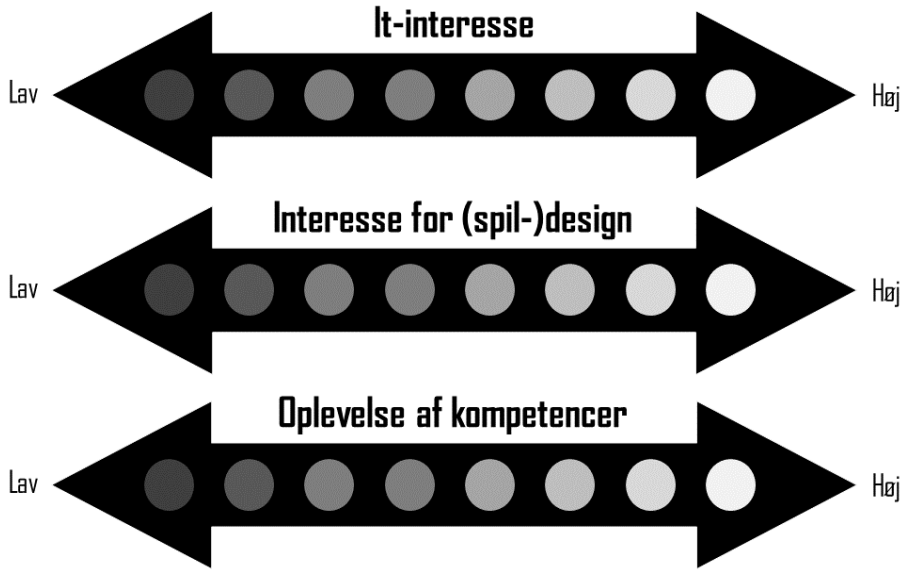
Centralt for elevernes dannelse er især, at eleverne forstår, at det er vigtigt, at vi via udviklingen af computationelle artefakter med- og nyskaber vores samfund. Via de to illustrative cases viste vi, hvordan fokus i Coding Class projektet primært har været på elevernes (spil-)produktioner. Diskussioner og udforskning af eller igennem computationelle artefakter i verden, har hverken været et tydeligt spor i interviewene med Coding Class instruktørerne, de kommunale koordinatorer, lærere eller elever. Vi betragter diskussioner og udforskning af, samt gennem computationelle artefakter i verden som væsentlig for både elevers og læreres forståelse af, hvordan og hvorfor arbejdet med informatik og computationel tænkning i og som fag er væsentligt i grundskolen. Her ses altså endnu et uudnyttet potentiale for at styrke elevers og læreres interesse såvel som blik for det væsentlige i Coding Class aktiviteterne.

Tema 2: Elevernes interesse og motivation

Det har været et erklæret mål i Coding Class projektet at fremme elevernes "interesse for it". Midlet har, som anbefalet af Aho (Committee for the Workshops on Computational Thinking, 2011, s. 36-37) været at arbejde "med kreative programmeringsprojekter som afsæt for at udvikle computationel tænkning", hvilket måske med tiden kan "motivere elever til at søge videre uddannelse og arbejde med fokus på informatik / datalogi". Vi har ikke fulgt de samme elever igennem længere tid og kender dem ikke i forvejen, derfor kan vi ikke vurdere, hvorledes deres motivation i projektet og interesse for it har ændret sig. Vi må derfor støtte os til deres egne og lærernes udtalelser.

En grundlæggende udfordring for lærere, der underviser med fokus på computationel tænkning, er som nævnt at lade elevernes interesse være i centrum for arbejdet med problemløsninger (Angeli et al., 2016). Coding Class har taget udgangspunkt i at lære at skabe spil igennem programmering. Spilinteresserede elever er således imødekommet gennem valg af emne.

I projektet har vi oplevet, at elevernes it-interesse, interesse for (spil-)design og oplevelse af egne faglige og tekniske kompetencer har betydning for elevernes deltagelse i og udbytte af Coding Class aktiviteterne. De tre elevforudsætninger kan placeres på et kontinuum mellem høj og lav:



Igennem interviewene med lærere, kommunale Coding Class koordinatorer og Coding Class instruktørerne er det ligeledes blevet tydeligt, at elevforudsætningerne har stor betydning. Vi har set, at lærere og Coding Class instruktører retrospektivt har reflekteret over dette, men ikke at de har medtænkt dette eksplicit i undervisningsforløbene. Elevforudsætningerne kan både være motivations- og modstandsskabende. Ligger interessen i elevens arbejde med it

eller/og i den skabende proces af et givent produkt (og i så fald hvilket)? Det er væsentligt at koble elevernes forudsætninger med Coding Class aktiviteterne. Læreren skal have indholdsviden i forhold til computationel tænkning (CK_{CT}) og kunne engagere elever med forskelligartede forudsætninger i aktiviteterne.

Læreren skal kunne pejle efter, hvordan den enkelte elev oplever forholdet mellem egne kompetencer (fagligt og teknisk) i forhold til de opgaver, eleven sætter sig for at løse. Det kræver, at læreren har viden om elevernes forudsætninger i relation til fx computationel tænkning (LK_{CT}). Denne læringsviden er en væsentlig forudsætning for at håndtere de ting, eleverne kan have svært ved fx, når de programmerer.

Er eleven for ambitiøs grundet en høj interesse for at skabe fx et godt spil, så er det en opgave at reducere kompleksiteten. Det kan fx gøres ved at opfordre eleverne til at arbejde inkrementelt og iterativt med opgaven, hvilket som tidligere nævnt kræver, at læreren har pædagogisk viden om computationel tænkning (PK_{CT}): Læreren skal kunne hjælpe eleverne med at prioritere, hvad der er vigtigst, og så udføre dette først. Det er også vigtigt, at især de elever, der forventer, at de ikke har de påkrævede kompetencer, hurtigt oplever succes - som eksempelvis gennem arbejdet med en indledende tutorial. En del elever giver udtryk for, at de hurtigt får udviklet noget via arbejdet med Scratch tutorials, og de udtrykker begejstring for denne form for individuelt arbejde med at lære at programmere og kode i Scratch .

Det relationelle - forholdet mellem lærere og elever - spiller væsentligt ind. Interessefællesskaber kan eksistere, men også skabes i klassen, som vi fx har set i tilfældet med Leo og hans Coding Class klasse. En elev, der var meget fanget af programmering, gav også udtryk for, at det var stærkt motiverende at dele sin interesse med klassekammeraterne. Eleverne oplever også, at samarbejdet i designprocesserne har indflydelse på deres motivation. Når samarbejdet er godt, stiger motivationen, og når det er skidt, så falder den. Elever giver endvidere udtryk for, at deres lærers interesse og motivation for emnet har en afsmittende effekt på elevernes engagement i undervisningen. Denne motivation omhandler ikke kun - men relaterer sig også til - lærerens interesse for den tekniske viden.

It-interesse og relevans

Flere elever gav udtryk for at have mødt projektet med negative forventninger grundet deres lave interesse for it:

> *I starten lød det rigtig kedeligt og jeg syntes ikke, det var særlig fascinerende, men nu er jeg helt vild med det og det troede jeg ikke, jeg ville være i starten. Jeg er glad for, at vi lærte om det.*

Eleven giver udtryk for, at det har været godt at blive introduceret for. Eleven ville ikke selv have valgt at beskæftige sig med programmering på egen hånd. Der er tegn på, at it-interessen rent fagligt er vakt hos nogle elever som ellers ikke ville have beskæftige sig med

programmering. Andre elever har mest betragtet det som en sjov form for undervisning, der ikke rigtigt relaterer til dem:

> *Det var sjovt at lave her i skolen, men jeg er ikke sikker på, at det er noget, jeg vil bruge så meget tid på i min fritid. Jeg er ikke så meget en it- type…...*

Der var på den anden side også elever, der gav udtryk for at have lært meget og at kunne se videre perspektiver i forhold til deres fremtid:

> *Det er altså mega sjovt Coding Class. Jeg vidste intet om at kode, og det har jeg lært, og er blevet ret god til det. Der er stadig nogle problemer, som jeg ikke kan løse uden hjælp, men man skal jo lære i skolen, så altså jeg tror, det er et af de sjoveste forløb i matematik.*

> *Hvis man nu keder sig derhjemme, begynder man tit bare at spille, men så når vi har lært at lave spil, kan vi jo selv lave et og bestemme, hvordan det skal være, eller når man bliver voksen, kan man blive programmør.*

Dog var der flere elever, der ikke kunne se fremtidsperspektiverne efter forløbet:

> *Jeg kan ikke rigtig se, hvad jeg skal bruge det til i fremtiden. Jeg vil nok mere tænke, at det var noget, man lavede i fritiden og ikke rigtig noget fagligt.*

Disse elever har forbundet forløbet med at være et spiludviklingsforløb snarere end programmeringsforløb, og de har forholdt sig til, om de vil være spiludviklere:

> *Det er meget sjovt nu, men jeg ved ikke, hvor meget jeg kommer til at bruge det i fremtiden. Jo, selvfølgeligt hvis du gerne, når du bliver stor, vil være et eller andet spille-noget eller sådan noget, så er det jo meget brugbart. Men ellers så ved jeg ikke lige sådan helt, hvad man skal bruge det til.*

Via interviewene med eleverne kan vi se et potentiale for at arbejde mere med transfer-aspektet af elevernes læring. Det har for de fleste været sjovt at udvikle *spil*, men dette har ikke nødvendigvis affødt, at de laver koblingen til, hvordan og hvorfor Coding Class aktiviteterne har været væsentlige for deres læring, og muligvis også for deres eget liv og fremtid.

Coding Class er landet meget forskelligt på skolerne. På Skole M har Coding Class aktiviteterne klare referencer til arbejdsmåder, eleverne allerede er familiære med. Leos Coding Class klasse har endvidere arbejdet med Coding Class knyttet til fagene natur og teknik og matematik i mere end et halvt år. Leo har - som Kolodner (Committee for the Workshops on Computational Thinking, 2010) anbefaler - gjort elevernes skabende og kreative arbejde med it til en aktiv del af skolehverdagen og en arbejdsmåde eleverne møder i flere meningsfulde sammenhænge på tværs af fag. Dette kan fremme læringstransfer, ifølge Kolodner. Derudover kan det hjælpe på transfer, hvis eleverne får lejlighed til at anvende og artikulere deres nyerhvervede kompetencer på tværs af sammenhænge. Endvidere kan lærerne udfordre eleverne til at tænke på andre situationer, hvor de kunne anvende de samme kompetencer og analytiske tænkemåder. Alt sammen praksisformer, som Leo entrerer med. Det kan meget vel være derfor, at eleverne fra Leos klasse i så høj

grad synes engagerede i og kan se perspektiverne i det at være skolens Coding Class klasse. En elev i Leos klasse sætter specielt ord på, hvordan hun personligt har oplevet udvikling og stærkere skole- samt fagligt engagement gennem arbejdet med en mere skabende og kreativ tilgang til it, herunder kodning og programmering i undervisningen, siden 4. klasse:

> *Elev 1: For mig personligt, så har det her arbejde fået mig til at få meget mere lyst til at komme i skole om morgenen. Nu forstår jeg, hvorfor jeg går i skole. Det giver meget mere mening. … vi var i Billund, hvor vi skulle undervise 50 mennesker, som var sådan rigtigt voksne. Vi skulle undervise dem. Det her Coding Class projekt har også fået mig til at turde at stille mig frem foran mange mennesker og sige, hvad jeg mener og føler uden*
>
> *Elev 2: at nogen griner*
>
> *Elev 1: ja. Det er bare virkelig. Det har været en stor oplevelse for mig, og jeg ved det vil fortsætte det næste år. Det er bare rigtigt dejligt….Jeg er sådan helt over, hvor meget det fylder, det der med at når jeg står op, så ved jeg, at jeg skal hen og lære noget nyt, i stedet for det der, neee,j jeg gider ikke. [21m] Det fylder bare virkeligt, virkeligt meget. Det er virkeligt dejligt.*
>
> *Interviewer: i dag er det i virkeligheden bare to timer, I har haft med det her?*
>
> *Elev1: ja, men det gør bare hele dagen meget bedre, fordi vi får lov til at… der er ingen begrænsninger for, hvad vi må lave. Jeg sad og skrev lidt på vores blog. En anden fra klassen lavede en hjemmeside...*
>
> *Interviewer: jeg kunne godt tænke mig at høre lidt mere om, hvad I synes, at I lærer, når I har de her Coding Class aktiviteter?*
>
> *Elev 1: før vi havde det, vidste jeg ikke engang, hvad kodning var. For mig har det taget logisk tænkning på et lidt højere plan. Jeg kunne slet ikke finde ud af, hvis jeg ikke havde en helt lukket opgave med én måde at løse den på, og jeg vidste, hvordan jeg løste den. Nu er det mere logisk tænkning, så man skal selv finde en måde at løse opgaven. ...og en computer er bare blevet meget mere for mig, end den var før i tiden… Jeg har lært i forhold til i 3. klasse, der sad jeg ved det dårligste matematikbord, og så har jeg virkelig lært, at jeg er blevet meget meget bedre til hovedregning. Jeg er bedre til hovedregning end min far. Det har bare lært mig matematik på en meget sjovere måde. Og det synes jeg bare, er rigtigt dejligt.*

Arbejdet med informatik og computationel tænkning kan både udvikle og knytte an til elevernes it-interesser. Hansbøl (2015) illustrerer, hvordan it-interesser og fremtidsdrømme kan se forskellige ud, og det influerer engagementer i it-undervisningen. Nogle drømmer fx om at udvikle fremtidens professionelle it-systemer, andre om at udvikle fremtidens forretningsmodeller i computerspil, og for nogle er det at blive it-support, der reparerer fremtidens hard- og software et ønskescenarie. I Coding Class projektet har den eksplicitte kobling fra Coding Class aktiviteterne til fremtidige studie- og karrieremuligheder ikke været i fokus. Her ser vi endnu et uudnyttet potentiale.

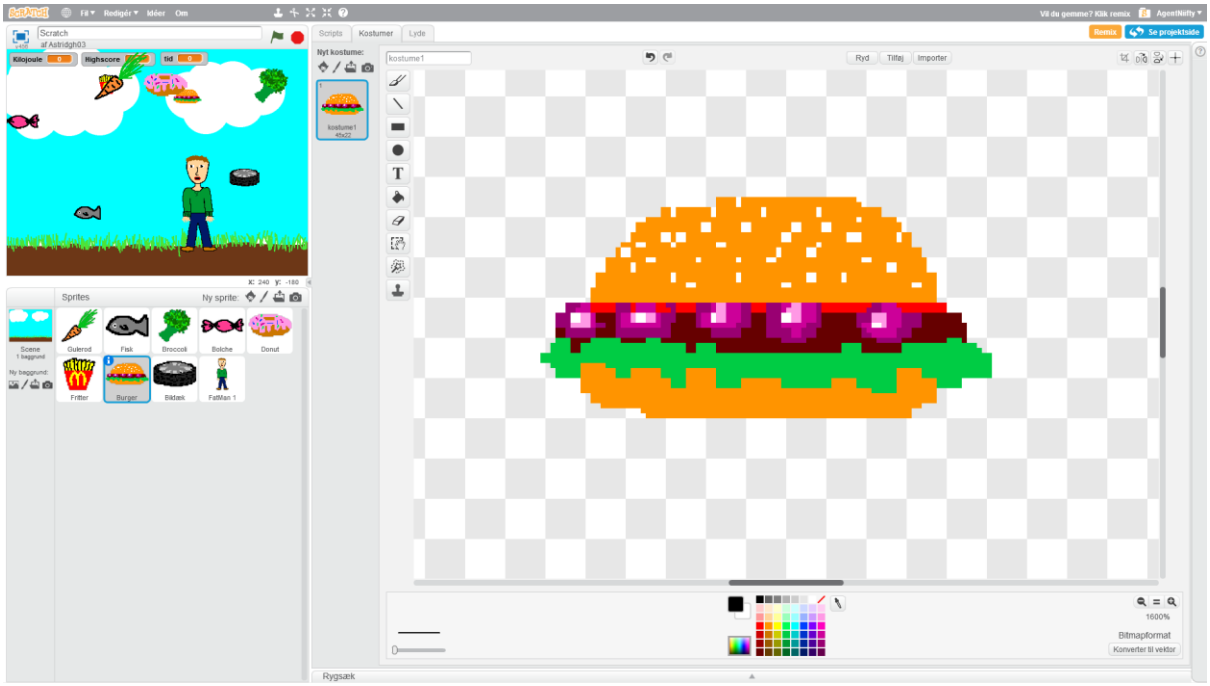
Interesse i (spil)designprocesser

For nogle elever er særligt (spil)designprocessen motiverende. Flere elever fremhæver, det at kunne arbejde skabende, på egen hånd med at lave spil som særligt motiverende:

Det her med, at det er et spil. Vi er jo børn. Hvis det var noget andet, så ville vi måske ikke være ligeså [motiverede]… men når det er et spil, vi skal lave…

Det er dog svært at udvikle et velfungerende spil og for nogle lykkes det ikke.

En gruppe har udviklet spillet "Fat man". Avataren skal fange sund mad og undgå usund mad, der falder ned fra himlen. Eleverne får aldrig spillet til at fungere, men er derimod meget optagede af at tegne spillets elementer i Scratchs grafiske værksted (se billedet).



Her ses en burger, eleverne har tegnet i Scratch til spillet Fat man.

Man kan argumentere for, at eleverne har været tændt af at skabe med it, omend de ikke var motiveret af eller i stand til at programmere. En lærer betoner, at flere af hendes svagere elever er blomstret op i Coding Class projektet, netop på grund af dets designmæssige, legende og æstetiske dimensioner, der kan motivere eleverne, men også tilbyder andre læringsmuligheder pga. de anderledes udtryksformer og muligheder for interaktivitet:

Lærer: ja. Altså (navn på elev) - hvis jeg havde sat det her op med matematik tal, tegn – hvis jeg kunne lave det hele, hvor det ikke var pænt og lækkert og det var tal, så ville han ikke finde ud af det

Interviewer: er det det æstetiske, der betyder noget eller det legende?

Lærer: Der er helt klart noget med det legende og bevægelse og sjov for ham. Og jeg tænker, det er det samme, der gør sig gældende her. Og tror du ikke også, at den lækkerhed, der er i det, måden det er sat op på, også at det er puslespilsbrikkerne – det giver mening, man kan sætte den der sammen med den der. Man behøver ikke tænke, man kan også bare prøve, nå, så virkede det.

Den sidste del af lærerens udsagn peger på, at den umiddelbare feedback, der understøtter, at eleven kan afprøve hypoteser, kan virke motiverende. Dette er en form kendt fra computerspil, men forholder sig også til elevens oplevelse af kompetencer og retning mod et mål.

Oplevelse af egen kompetence og motivation

Det kan synes skræmmende at kaste sig ud i programmering, på grund af den forventede høje sværhedsgrad:

> *Jeg har aldrig prøvet kodning før, og jeg har altid troet, at kodning kun var noget med lange komplicerede sætninger med tal, men så, når man kom i gang med det, blev det sjovt og udfordrende. Det, at vi gjorde det i Scratch, gjorde det også meget sjovere og nemmere.*

Idet eleverne kan arbejde med blokprogrammering og på andre legende måder, kan sværhedsgraden sænkes, uden at det dog bliver nemt. Derfor er det også vigtigt, at eleverne er vedholdende, især, når der ikke er hjælp at finde. Eleverne oplever alle, at de sætter noget kode sammen, der ikke virker som intenderet. Men det kan være svært at lokalisere fejl og løse dem, for hvad er 'bugs' og hvad er elevens egen manglende formåen? Netop denne del af kodearbejdet kan som tidligere nævnt virke demotiverende på eleverne, ifølge dem vi interviewede. En lærer påpeger, at det især var en udfordring for nogle af de skarpe elever i klassen. Ifølge læreren giver nogle af de skarpe elever op, når de møder modstand og udfordringer:

> *Lærer: Så har jeg faktisk nogle rigtigt skarpe drenge, der har givet lidt op – der ligesom har slået for stort op og så er det blevet lidt for svært. Og de er vant til at kunne det hele. Det der med at skulle virkeligt til at knokle og pille lidt fra hinanden igen og lave noget systematisk kedeligt arbejde med at prøve klodserne af.*

Det kan virke som om, de har været for ambitiøse i, hvad de ønsker at udvikle - en problemstilling Coding Class instruktører og flere lærere er opmærksomme på, idet de opfordrer til at tænke i simple løsninger. Lærerens kommentar peger imidlertid også på, at de skarpe elever kan blive demotiverede på grund af den store udfordring:

> *Lærer: eller måske er rollerne skiftet og så var de lige pludseligt ikke de bedste, og det er ikke særligt rart, når man er vant til at være den bedste.*

Elevernes oplevelse af egen formåen hænger sammen med elevernes motivation i Coding Class aktiviteterne. For nogle elever styrker Coding Class aktiviteterne deres oplevelse af egen formåen, mens andre elevers oplevelse af egen formåen bliver svækket. Elevernes motivation og oplevelse af egen formåen hænger imidlertid også sammen med omfanget af Coding Class aktiviteterne og hvilke aktiviteter eleverne arbejder med på forskellige tidspunkter. Flere af de elever og lærere vi har interviewet, fremhæver at det eleverne oplever det som hårdt eller kedeligt at have Coding Class en hel dag. Både fordi de arbejder så koncentreret med programmeringen, gruppesamarbejdet, og fordi de kigger meget på en skærm. En del elever blev meget trætte efter frokost. Udover, at det er relevant for læreren at have blik for, hvordan elevernes oplevelse af egen formåen udvikler sig i Coding Class

aktiviteterne, så er det endvidere hensigtsmæssigt, at læreren overvejer, hvordan undervisningsdagene varieres, samt hvor lange (massive og intensive) de skal være, for at eleverne får optimalt udbytte af undervisningen.

Tema 3: Lærere og pædagogik

Lærerne med om bord

De deltagende lærere, vi har interviewet, har meget forskellige udgangspunkter for at undervise i computationel tænkning, spiludvikling, kodning og programmering, men er alle motiverede for at deltage. De deltagende lærere har primært været matematiklærere. Én er pædagogisk it-vejleder på skolen. To af de interviewede lærere har allerede prøvet at arbejde med programmering i klassen inden forløbet. De andre lærere deltager for første gang i programmeringsforløb.

Coding Class har været et undervisningsforløb, hvor lærerne har deltaget med deres klasser. Lærerne har indledningsvist fået en kort introduktion til Scratch, men Coding Class projektet og Coding Class instruktørerne har haft fokus på at udvikle elevernes kompetencer. Intentionen med Coding Class har alligevel været, at Coding Class instruktørerne gradvist skulle slippe styringen i forløbet, så lærerne kunne overtage og skolerne selv kunne videreføre Coding Class aktiviteterne. Dette har ikke kunne lade sig gøre fx på Skole K, og der, hvor vi har set det lade sig gøre fx Skole M, der sker det især: 1. når læreren i forvejen har været interesseret i arbejdet med elevernes kreative og skabende it-kompetencer, spiludvikling, kodning og programmering og it-understøttet undervisning, og 2. når læreren har engageret sig i kompetenceudvikling som led i hans/hendes deltagelse i projektet. En af de deltagende lærere betoner netop, at forløbet har været godt, fordi hun allerede forstår projektet og er interesseret i det:

> *...jeg har lært mig selv at kode HTML, og der er nogle ting, der går igen. …fordi jeg er ...matematiker, at tankegangen i Scratch den passer ind i mit hoved – hvor jeg har en anden kollega, der har været oppe at se noget undervisning, fordi hun godt kunne tænke sig at lære scratch lidt at kende og hun siger, at hun forstår det ikke ... og jeg tror ikke, hun ville kunne have lavet den samme undervisning med den samme indgang til det, som jeg kan.*

Samme lærer har introduceret andre lærere til især intro-forløbet, og de kan nu køre forløbet på egen hånd på skolen. Den fælles introduktion igennem et enkelt spil, hvor alle eleverne er med, har vist sig at være rigtig værdifuld for de deltagende lærere. Denne introduktion præsenterer grundbegreber og muligheder ved Scratch og programmering samtidig med, at den reducerer kompleksiteten både for lærerne og for eleverne.

Idet forløbene havde spiludvikling som omdrejningspunkt, så stiller det krav til lærernes forståelse af spilgenren. En lærer fremhæver netop dette som en kompetence, Coding Class instruktørerne bød ind med:

...han (deltager fra Coding Class red.) kunne ligesom overskue, hvad det var for nogle spil, [eleverne red.] lavede – det der var et runner spil og det der var… – og det tog jeg vildt mange noter til. Og så tog han sådan en lille session på 10 minutter, hvor alle de grupper, der lavede et runnerspil, de skulle komme op til ham og så forklarede han lidt om et runner-spil og lavede et eksempel på tavlen. Der tog jeg sindssygt mange noter og så fik jeg linket til den scratchfil – det han lige havde siddet og lavet - så jeg havde det på min computer.

Læreren skal altså ikke kun sætte sig ind i Scratch, men også have en viden om design af spil og kreative processer. Det kan hjælpe at afgrænse til én slags spil og så identificere hvilke elementer, der skal til for at lave et fx et spil inden for denne genre. Som Coding Class instruktørerne gør med labyrintspillet, hvor de faste elementer er:

- En labyrint
- En avatar som brugeren styrer - typisk med piletasterne (musen)
- Et mål (osten)
- Et "winning state" (at få fat i osten - eller evt. tid der tæller ned eller op)

Eventuelt:

- En eller flere fjender - som gerne bliver hurtigere eller flere med tiden, der tager liv eller point fra brugeren (kan være nogle med en adfærd, en musefælde eller labyrintens vægge)
- Et antal liv
- Evt powerups (dvs noget der beskytter eller giver brugeren liv)

Læreren kan inddrage børnene i at identificere disse elementer ved at lade dem studere lignende spil, som de finder interessante og så selv imitere dem (Ejsing-Duun & Tosca, 2017)

Lærerne skal altså ikke blot have styr på Scratch, kodning og programmering, men også have designmæssige kompetencer, grundlæggende it-forståelse og spilgenreforståelse for at kunne engagere Coding Class aktiviteter i undervisningen.

Pædagogiske greb

Lærerne har forskellige strategier til at erhverve sig øgede kompetencer inden for kreativ produktion med it. Læreren i ovenstående citat følger aktivt undervisningen og bruger lejligheden til at skabe sig et arkiv og repertoire til selv at kunne stå med undervisningen ved at tage gode noter samt gemme dem med scratch-projekter. Jytte fra Skole K kaster sig over Scratch, når hun har chancen:

> *Jeg sad selv og prøvede at lave nogle spil, for ligesom at komme ind i det der. Og vi har sådan nogle studietimer, hvor de sidder og arbejder selv. Der sad jeg og bøvlede med mine spil, fordi de bare blev ved med at drille. Og i løbet af fem minutter stod der fem [elever] omkring mig, der sagde, "har du prøvet… hvis du nu fjerner den der for evigt. Og du skal have den til at starte på et andet x." ...det var bare et nyt sprog, de havde lært i løbet af to gange. Og det var ikke engang deres spil. De kommer lige ind fra højre og siger: ahhh, jeg kan godt se, hvordan får vi lige. Og de tog den på sig. Det, synes jeg, er vildt fedt at mærke.*

Læreren tør her at kaste sig ud i at arbejde med programmet, og lader eleverne udfolde deres viden. Hun bliver begejstret over, at eleverne er blevet så kompetente, at de kan hjælpe hende. I begge disse to eksempler - læreren der tager noter og læreren der selv engagerer sig i arbejdet med at programmere via Scratch - positionerer lærerne sig selv som lærende, der er åbne overfor at tilegne sig nyt stof, hvor en væsentligt del af det handler om at opnå teknologisk viden. Leo har ligeledes, som vi har set tidligere, en tilgang, hvor det at lære computationel teknologi at kende er en del af hans udvikling af undervisningen. Både Jytte og Leo har endvidere en tilgang, der involverer, at lærere og elever i fællesskab udforsker og lærer teknologien at kende.

Denne tilgang hænger også sammen med et pædagogisk mindset, hvor der er mere fokus på at rammesætte end formidle et stof. Jytte og Leo søger her at tilegne sig TK_{CT} - teknologisk/teknisk viden om Scratch og kreativ programmering - undervejs i undervisningen, sammen med eleverne. De giver eleverne mulighed for at vise, hvad de har styr på og måske også, hvad de synes, der er svært - hvilket kan give læreren Learner Knowledge (LK_{CT}). Ved at gå ind i processen selv og arbejde med at bygge et program, får læreren endvidere viden, der kan hjælpe ham/hende med at skabe Pedagogical Knowledge (PK_{CT}) - viden om, hvordan arbejdsprocesserne kan stilladseres og hvilke problemløsningsmodeller, der giver mening i kontekst af programmering.

Der er ikke nogen tvivl om, at det at få andre professionelle ind i skolehverdagen i form af Coding Class instruktørerne, har været livgivende i forhold til at igangsætte aktiviteter på skolerne, der ellers ikke havde været mulige. Vi har observeret, at Coding Class instruktørerne har lavet korte oplæg – relateret til mål og fagligt indhold samt problemer eleverne er stødt på undervejs i projektet. De har i deres vejledning været undersøgende sammen med eleverne og søgt ved at spørge ind til elevers intentioner, metoder, og ved at interessere sig for hvilke hypoteser eleverne arbejder ud fra, og hvad eleverne allerede har forsøgt, at understøtte elevernes refleksioner (in-action/on-action). Derudover har de givet eleverne små 'puzzles', hvor kompleksiteten er reduceret - eksempelvis tager en af instruktørerne fire klodser ind på canvas og beder eleverne samle dem, så de producerer et bestemt resultat. Eleverne får så lov til at teste deres hypoteser, idet de umiddelbart derefter kan se resultatet.

En af Coding Class instruktørerne beskriver sin egen tilgang:

> *...lærerens rolle er anderledes i den her proces, fordi det netop er en opdagelse, læreren gør sig sammen med eleverne i stedet for en ren undervisning af materialet. Det er det, jeg også siger hver gang til lærerne, fordi den måde jeg er gået til det på, når jeg har haft mit eget valgfag, det er, at jeg har sagt, jeg kan ikke alt i Scratch og jeg kan ikke alt i andre kodningssprog, hvis det er det, vi arbejder med, men jeg er villig til at finde ud af det sammen med eleverne.*

Han betoner her, at elever og lærere samskaber agendaen og den teknologiske viden, der er behov for via fælles engagementer i teknologien i undervisningen TK_{CT}. Altså, er det ikke lærerens viden og prædefinerede mål, der styrer arbejdet, men derimod elevernes og lærernes fælles interesse og de problemer, de i det lærende fællesskab oplever som

relevante. Det sætter læreren i den situation, at de kan komme til kort, når eleverne ikke selv kan løse problemerne. Men det bør læreren kunne rumme ifølge Coding Class instruktøren:

> *Så det er også noget, jeg præsenterer for lærerne i Coding Class, at det kan godt være eleverne har en idé til noget i spillet, som jeg ikke lige har en løsning på, men derfor behøver du ikke at sige: "det kan du ikke lave, for det er for svært" – så kan man sige: "lad os prøve at finde ud af at løse det" – og hvis man så ikke kan tænke sig til det, så kan man google det og google det sammen med eleverne. ...Så man lærer noget som lærer, eller også kan eleverne lære noget af en. Og - den proces - den der facilitatorrolle - tror jeg, der er mange lærere, der synes, er fed at tage, når vi har Coding Class.*

Denne tilgang betyder dog ikke, at undervisningen ikke er rammesat - en anden af Coding Class instruktørerne udtaler:

> *Det har så været vigtigere, at alle børnene ikke har kørt hver deres spor. Fordi hvis man skal starte op med at lære noget grundlæggende omkring det her program, så er man også nødt til, at eleverne i begyndelsen følges ad for senere hen at blive sluppet fri og blive mere kreative.*

Når eleverne må arbejde rimeligt frit med egne projekter inden for den fastsatte ramme, så lægges ansvar over på børnene. For at de får kompetencer til at kunne håndtere ansvaret bør lærerne gennem en eksplorativ samtale arbejde med ikke at vise eleverne, hvordan man gør, men at give dem idéer til, hvad de kunne gøre. En Coding Class instruktør giver et eksempel på, hvordan man kan tale med eleven:

> *Det er at sige "så skal du have ændret noget, du skal have den til at gå opad" – så kan det være, eleven siger: "Nå ja, så skal jeg have ændret på y-aksen". For det har de lært, at det er opad. Okay – vi skal have ændret noget, så den kan bevæge sig – så klikker de på koden og så siger de måske noget. Og hvis de siger noget, så tager man fat i det. Og ellers så giver man nogle forslag – selvfølgelig – der er forskel på elever… det er en didaktisk/pædagogisk kompetence at kunne mærke på dem, hvor de ligger – om de er frustrerede – om de har brug for mere hjælp, eller om de skal have frirum til at klare sig selv*

Coding Class instruktøren peger her på, at læreren skal have, hvad Angeli et al. (2016) kendetegner som Learner Knowledge (LK_{CT}), altså viden om, hvor elevernes forståelse for og problemer med computationel tænkning kan ligge, og bruge det som udgangspunkt.

Coding Class instruktørerne og flere af lærerne betoner dog, at denne slags viden har man ikke fra begyndelsen - ligesom man ikke har Technology Knowledge (TK_{CT}). Technology Knowledge er vigtigt, at have i den forstand, at lærere og elever udforsker værktøjerne. De opfordrer lærerne til at komme i gang med programmeringsprojekter, med tillid til børnene og egne didaktiske kompetencer, så de får noget spillerum, og så kommer viden gennem erfaringerne. Omend det ville være at foretrække, at lærerne havde fået noget af denne viden med sig. Leos klasse anvender, som vi allerede har set, flere forskellige strategier til at få adgang til (teknologisk) viden i klassen:

> *Interviewer og når nu I skal gøre noget med noget I aldrig har set før. Hvad gør I så?*
>
> *Elev 1: går ind på og klikker på noget.*

Elev 2: Så kan det være der er en, der lige har fundet på noget… I vores klasse kan vi godt lide at stjæle fra hinanden.

Elev 3: ideer

Elev 2: Hvis der er en der har brugt en knaldgod idé, så stjæler vi andre den [griner]

Elev 3: Så tager man noget fra de andres, men der er måske også nogen der har taget noget fra vores.

….

Interviewer: men hvordan er det så. Når Leo siger I skal have om isbjerge, er det så jer der undersøger og underviser hinanden?

Elev 1: research på nettet

Elev 2: han siger ikke et ord. Bare lav en fremlæggelse.

Elev 1: han siger måske, der er Clio Online, den skal I bruge. Og så er det bare at gå i gang. For det meste skal vi lave en fremlæggelse, hvor vi selv må vælge hvilket program vi vil gøre det på. Om vi vil gøre det på Google slides, docs eller om vi gør det med en planche.

I Leos klasse er det en væsentlig strategi for både Leo og eleverne at udforske viden i klassen som et lærende fællesskab. Det betyder, at både Leo og eleverne har udviklet mange forskellige praksisser for at gå til ny viden i klassen sammen. Med afsæt i Coding Class instruktørernes praksisser og erfaringerne fra fx Skole K (Jytte) og M (Leo) vil vi pege på, at der i videreudviklingen af Coding Class aktiviteter med fordel kan være mere fokus på lærernes kompetenceudvikling af disse særlige eksplorative videnspraksisser i lærende fællesskaber med eleverne i undervisningen.

Tema 4: Videre perspektiver i skolen

Coding Class projektets fjerde mål er at sætte fokus på kreativ brug af it og computationel tænkning som vidensdomæne i grundskolen. Her ser vi på, hvordan de deltagende lærere, elever og Coding Class instruktørerne vurderer muligheder og potentialer, samt hvad vi har peget ud i rapporten i relation til dette tema.

Computationel tænkning i skolen

Aho fremhæver som nævnt fem udbredte argumenter for at inkludere computationel tænkning i skolens curriculum (Committee for the Workshops on Computational Thinking, 2011). At computationel tænkning:

1. har indflydelse indenfor mange fagområder
2. kan forebygge samfundsmæssige og menneskelige risici
3. kan skabe mulighed for at innovere måder at skabe, forstå og manipulere repræsentationer.
4. kan motivere elever til at søge videre uddannelse og arbejde med fokus på informatik / datalogi.
5. når det sættes i relation til nye områder kan bibringe nye metoder og problemløsningsprocesser.

Coding Class bygger, som tidligere nævnt, ikke i særlig grad på computationel tænknings betydning på tværs af fag (punkt 1), men har mere haft fokus på informatik og computationel tænkning som et særligt vidensdomæne - et it-fag (punkt 3 og 5), og Coding Class projektet har taget afsæt i et grundlæggende behov for at vække elevernes interesse for it (punkt 4). Vi har allerede berørt, hvordan Coding Class aktiviteterne kan påvirke elevernes motivation og interesser. Men hvordan forholder eleverne sig til fremtidsudsigterne for at kunne arbejde kreativt med it i skolen? Hvordan forholder lærerne, de kommunale aktører og Coding Class instruktørerne sig til fremtidsudsigterne for at introducere computationel tænkning i skolen?

Eleverne er delte i deres holdninger: En del forbinder forløbet med at lære om spiludvikling og kan ikke se en umiddelbar anvendelse for dem i fremtiden, andet end måske som en del af deres legekultur. Som tidligere nævnt er Coding Class kun ét forløb, og vi antager at eleverne vil få en øget fornemmelse af faget, hvis det implementeres. En anden gruppe elever - Leos elever, som har haft mere undervisning i kreativ brug af it - har allerede fået en fornemmelse for fagets betydning for dem:

> *Elev 1: for det første er det sjovt. Og det er rigtig rigtig lærerigt, når man skal til at have job. At man kan finde ud af at kode. Så ved man, at han er rigtig god til matematik, og han er god på det logiske plan.*
>
> *Elev 3: man kan bruge kodning til hvis man skal have et job, hvor man skal bruge computere meget. Fx hvis man skal lave noget med at printe ud fra en 3D printer. Hvis man har lært det, så kan man blive en af dem, der er god til det.*

> *Elev 2: jeg kunne rigtig rigtig godt tænke mig at blive sådan en, der lavede spil til Appstore, Playstation eller… jeg kender en, der har det som arbejde at lave spil.. eller hjemmesider.*
>
> *Elev 3: hvis man er ligesom Elev 2, så kan man jo blive sådan en som Coding Class instruktørerne, hvor man kan komme ud på skoler og lærer folk at programmere.*

Nogle af eleverne har allerede idéer om hvilke typer af jobs, Coding Class giver dem kompetencer til. De deltagende lærerne er meget interesserede i at knytte computationel tænkning til skolens fag. En lærer argumenterer for, at nogle elever vil kunne få øjnene op for en interessant faglighed:

> *Vi har håndarbejde og billedkunst og vi har sløjd og så har vi engelsk – nej håndværk og design – og vi har engelsk, tysk og fransk. Det gør vi, fordi hvis [eleverne] vil læse videre, så har de en basisviden – og samfundsmæssigt har de en basisviden, der er en dannelse og viden rent samfundsmæssigt. Og der synes jeg bare, kodning og computer literacy er blevet en basisting.*
>
> *Og hvem siger ikke, at jeg har nogle piger nu, der har siddet i timerne og tænkt: "Gud, det er fedt det her – den tankegang det er noget for mig – helt sikkert skal jeg indenfor den branche, når jeg skal uddanne mig videre eller jeg skal lege lidt mere med det". … Det er vores pligt, at vi i folkeskolen at sørge for at eleverne blive introduceret for basis, så de har et ordentligt beslutningsgrundlag og en vis dannelse - og det synes jeg, kodning er blevet – og i hele taget kunne agere i en computer…*

Netop fordi en del af eleverne ikke opfatter kodning eller programmering som værende relevant for dem, er det vigtigt ifølge læreren at introducere til denne særlige faglighed. Læreren fremhæver også, at det er væsentligt, at lærerne uddannes til det og læringsmålene skal omfatte ikke kun kodning, hvilket vil være at gribe faget for "instrumentelt" an, men i det hele taget til computationel tænkning som en skabende tilgang til verden. Det handler om, at give eleverne *"digital dannelse"*, så de ved, hvad "*der ligger bagved [systemerne]*" og kan "*Deltage i demokratiske beslutninger omkring det.*"

Hvor i skolelivet passer arbejdet med aktiviteter som Coding Class? Flere elever og lærere peger på, at man ikke kan lægge timer til skoledagen, men må tage timerne fra andre fag. Helst allerede tidligt i skoleforløbet. En kommunal aktør udtaler:

> *Det skal puttes ind i fag fra begyndelsen af skolegangen. Vi arbejder jo allerede med robotter i daginstitutioner og dagtilbud. Så kan det jo ikke nytte noget, at de først møder det igen som valgfag i udskolingen.*

Argumentet bygger på kommunens ønske om kontinuitet og progression i arbejdet med elevernes digitale dannelse fra 0-18 års området. En lærer argumenterer også for at det skal lægges ind allerede i indskolingen og mellemtrinnet (3. -5. klasse), fordi eleverne her er "medgørlige, mere fokuserede" – modsat i de senere klasser, hvor eleverne ifølge ham har dannet sig en idé om, hvordan man gør skole - herunder hvilke fag og fagligheder der er væsentlige. Flere lærere og elever understreger også at forløbet skal være obligatorisk/for alle, så nogle af de elever, der ikke troede, arbejdet med computere var noget for dem, kan

blive introducere for det.

Når informatik og computationel tænkning indføres - som fag og/eller på tværs af fag - så udfolder der sig en central udfordring, som en kommunal aktører inden for skoleområdet understreger:

> *Det helt store problem med kodning er, at det er ikke noget lærerne har fået med fra læreruddannelsen. Så hvis du kan finde personer i skolerne, der har de her kompetencer, så er det fordi det er personer der kan det i forvejen. Måske har de det som personlig interesse, eller de har en baggrund, hvor de kan kode.*

Det er et lotteri for børnene, om de får lærere, der faktisk kan undervise dem inden for fagligheden. For at lærerne kan få fagligheden, mens de er i praksis, skal de have ressourcer:

> *Det er en hindring, at man som lærer er alene i klasseværelset, for man skal være opmærksom på mange ting. Og når man ikke er mere sej til kodning – det er også rimeligt nyt for mig – så er det svært at koncentrere sig om at hjælpe, når koden bliver lidt svær. Jeg kan ikke lukke hele klasseværelset ude og sætte mig ned for, at bruge en halv time for at hjælpe nogen videre. …Jeg er helt sikker på, at der er mange af tingene, jeg kunne løse selv. Men ikke når jeg ikke kan sidde og fordybe mig og koncentrere mig om det. Det er fordi, jeg ikke fagligt er stærk nok til at kunne ryste det ud af ærmet.*

Læreren giver her udtryk for at mangle faglighed til på nuværende tidspunkt at kunne løfte Coding Class aktiviteter alene i undervisningen. På trods af, at hun er åben for at kaste sig ud i forløbet, giver eleverne plads til at vise deres kompetencer, og er it-vejleder. Uddannelse af lærerne og prioritering af ressourcer til lærernes kompetenceudvikling er helt grundlæggende.

I Coding Class har det fælles afsæt været Scratch, der kan bruges til at udvikle en række forskellige programmer, og der findes mange eksempler lærere og elever kan arbejde videre med i skolen - især spileksempler. Forløbet ville have knyttet an til et helt andet indhold og måske også fagligheder, hvis eleverne havde programmeret en LEGO Mindstorms robot, eller fx programmeret Ozobots bevægelse med tuscher. Dette er også centralt at nævne. Valg af program og aktiviteter har betydning for mulighederne for at knytte an til elevernes computationelle tænkning og digitale dannelse i undervisningen. Fx ville det også have givet andre muligheder, hvis Coding Class aktiviteterne havde taget afsæt i reelle problemstillinger og innovations- og entrepreneurskabspædagogiske tilgange til undervisningen, frem for at vælge en spildidaktisk tilgang.

De elever, vi har talt med i Coding Class projektet, har samstemmigt peget på, at det helt centrale for videreførslen af Coding Class aktiviteter i skolehverdagen er lærernes interesse, engagement og - ikke mindst - kompetencer.

Konklusion

I denne rapport har vi fremlagt vores evaluering og dokumentation af aktiviteterne i Coding Class projektet. Vi har ikke dækket alt, men foretaget kvalitative nedslag gennem observation og interviews med projektets forskellige deltagere på udvalgte skoler i både København og Vejle. Vi har forholdt erfaringerne til international, nyere forskning, med fokus på arbejdet med computationel tænkning, for at sætte ord på erfaringerne samt pege på, hvad vi kan lære af projektet, men også for at identificere udviklingspotentialer og hindringer for det videre arbejde med informatik og computationel tænkning i grundskolen.

Coding Class projektet er et pilotprojekt, og vores dokumentation af Coding Class projektet omfatter et relativt lille og kvalitativt orienteret empirisk materiale. Samtidig er der igennem Coding Class skabt forsøg med praksisser, der ellers kun eksisterer i ringe grad i en dansk skolekontekst, og derfor har projektet produceret et vigtigt afsæt for den videre diskussion og implementering af arbejdet med informatik og computationel tænkning i den danske grundskole. Herunder, hvilke lærerkompetencer det forudsætter og hvilke pædagogisk-didaktiske greb, der især er væsentlige at have for øje. Konklusionerne i nærværende afsnit skal ses i lyset af projektets størrelse.

Konklusionen på projektet deles ind i rapportens forskellige bidrag:

- Teoretiske bidrag
- Praksiserfaringer
- Vurdering af de fire formål i Coding Class med afsæt i teori og praksiserfaringer

I denne sidste del af rapporten zoomer vi altså ind på, hvad vi har lært på baggrund af Coding Class projektet. På hvilke måder har Coding Class projektet understøttet arbejdet med at udvikle elevernes computationelle tænkning? Hvilke særlige læringsmæssige og pædagogiske perspektiver er udfoldet i relation til Coding Class projektet? Hvilke udfordringer og muligheder for at udbrede arbejdet med elevernes kreative it-kompetencer og computationelle tænkning har vist sig i relation til Coding Class projektet?

Første bidrag: Teoretiske bidrag

Vi har haft behov for at ridse scenen op og tydeliggøre vores ståsted for at dokumentere og evaluere Coding Class aktiviteterne, samt bidrage til diskussionen af vidensdomænet informatik computationel tænkning i grundskolen.

Vi har redegjort for informatik og computationel tænkning som et nyt vidensdomæne, der knytter an til andre vidensdomæner. Wing (2006) definerer computationel tænkning som en problemløsende tilgang til verden og dens computerrelaterede problemstillinger, der er særligt knyttet til STEM fagene. Brennan & Resnick (2012) og Kafai (2016) udvider Wings forståelse af computationel tænkning til også at involvere 'arts', og dermed betoner forskerne de sociale, kreative og skabende elementer ved vidensdomænet. I forlængelse heraf opsummerer Caspersen (2017), at computationel tænkning (informatik) giver nye udtryks-, erkendelsesmuligheder og radikalt nye sociale og fællesskabsmuligheder. Computationel

tænkning omfatter ifølge Angeli et al (2016) abstraktion, generalisering, dekomposition, algoritmisk tænkning og debugging (fejlsøgning og korrigering af fejl). Brennan og Resnick (2012) er mere specifikke – men reducerer også kompleksiteten – idet de beskriver, hvordan computationel tænkning omfatter computationelle begreber, design- og deltagelsespraksisser, samt computationelle perspektiver og et skabende og udforskende mindset.

For at kunne undervise i det skal læreren ifølge Angeli et al (2016) have Content Knowledge (CK_{CT}) (viden om computationel tænkning og tilknyttede kompetencer); Learner Knowledge (LK_{CT}) (viden om, hvad eleverne kan have svært ved - læringsviden); Pedagogical Knowledge (PK_{CT}) (pædagogisk viden og pædagogiske greb knyttet til computationel tænkning); Technology Knowledge (TK_{CT}) (viden om og færdigheder til at kunne bruge, udvikle og tilpasse teknologier til undervisningen); samt Context Knowledge (CX_{CT}) (bevidsthed om værdien af computationel tænkning i skolen, for samfundet og for eleverne).

Computationel tænkning defineres ofte i meget overordnede begreber, men knyttet til de konkrete praksisser skaber de, som vist i rapporten, en øget forståelse for den kompleksitet, der er forbundet med at lære elever om denne tilgang til viden, udtryk og samvær.

Andet bidrag: Dokumentation af to cases

Vi har udarbejdet detaljerede beskrivelser af to forløb på forskellige skoler. Det har vi gjort for at dokumentere disse nye praksisser samt at give inspiration til videre arbejde med computationel tænkning. Med casebeskrivelserne har vi sigtet mod at give en praksisnær viden om, hvordan arbejdet med computationel tænkning kan udfoldes i et skolerum. De to cases har vist, at det for de uerfarne lærere kræver en del at gå ind i aktiviteterne omkring computationel tænkning, men også, at forskellen mellem mere erfarne og uerfarne lærere netop er erfaringen, modet og tilgangen til arbejdet med computere som eksplorative og skabende redskaber snarere end en skoling i fagområdet.

Vi så hos de deltagende lærere et meget lidt udviklet sprog for, hvad computationel tænkning og programmering er. Det har heller ikke været nævneværdigt i fokus i projektet via Coding Class instruktørerne, som mere har fokuseret på design-baserede læreprocesser, spiludvikling og kodning og programmering med fx Scratch. Informatik og computationel tænkning som indholdsdimension har snarere været implicit repræsenteret i aktiviteterne. Det samme gælder de pædagogisk-didaktiske greb, der understøtter arbejdet med vidensdomænet i undervisningen. Fokus i Coding Class projektet har været på elevernes kompetencer til at arbejde designorienteret med spiludvikling via kodning og programmering i Scratch, og som vi har vist i rapporten, knytter dette på mange forskellige måder an til vidensdomænet informatik og computationel tækning. Vi vurderer, at fokus på lærernes kompetenceudvikling inden for fagområdet er en nødvendighed for at kunne kvalificere arbejdet med aktiviteterne og sikre udbredelse af aktiviteterne i skolehverdagen fremadrettet. Herudover er der behov for at udvikle lærernes kompetencer til at knytte informatik og computationel tænkning an til de allerede eksisterende fag og faglige aktiviteter i skolen.

Tredje bidrag: Vurdering af de fire formål

Vores arbejde med at evaluere Coding Class projektet i relation til dets fire formål sigter mod at redegøre for de læringsmæssige og pædagogiske gevinster, samt pege på potentielle udfordringer og muligheder. Coding Class har haft til formål at skabe forandringer i praksis, idet projektet har sigtet mod:

1. at fremme elevernes forståelse for den digitaliserede verden, der omgiver dem nu og i fremtiden (læringsudbytte)
2. at fremme elevernes interesse for it (motivation)
3. at igangsætte undervisning, der fremmer elevernes evne til selv at arbejde mere kreativt og skabende med it i grundskolen (pædagogik)
4. at sætte fokus på it, kodning og computationel tænkning som vidensdomæne i grundskolen (de videre perspektiver)

Fremme forståelse for digitaliseret verden

Igennem Coding Class har eleverne stiftet bekendskab med vidensdomænet informatik, idet de har nedbrudt problemstillinger (dekomposition) og udtrykket enkeltdele af et komplekst hele (abstraktion), bygget sekvenser op, der eksekveres under rette betingelser (algoritmisk tænkning) og testet koden for at rette fejl (debugging). Eleverne har stiftet bekendtskab med sekvenser, loops, hændelser, parallelitet, betingelser, data og i nogen grad til operatorer. Alt dette er dog primært på et indirekte niveau, idet den kreative proces med at skabe primært spil har været i forgrunden for eleverne.

Eleverne har kæmpet med abstraktionen, idet de har haft svært ved at have overblik over koden, der også påvirkede deres fejlfindingsprocesser. Dette er dels fordi de er uerfarne, dels fordi canvas i Scratch er lille, og koden er distribueret på forskellige sprites.

Vi så, at nogle børn allerede gennem frivillige aktiviteter har gode kompetencer og nogle havde meget lidt at trække på. Dette gab var i nogen grad en udfordring i samarbejdet mellem eleverne, de steder hvor arbejdet med elevernes skabende og kreative it-kompetencer, kodning og programmering, var nyt. I de klasser hvor eleverne havde større erfaringer, var gabet dog mindre tydeligt. Samarbejdet hænger også meget nært sammen med læringskulturen etableret af lærere og elever, samt deres evne til at lære i fællesskab. Idet lærere og elever har forskellige kompetencer at byde ind med og på grund af den høje kompleksitet, lægger vidensdomænet op til en omstrukturering af de mere traditionelle skolehierakier. Vi har således set, hvordan lærere sætter sig i den lærendes sted med eleverne, men også udfordrer dem til selvstændigt at søge viden og ressourcer, hvorved elevernes tilgang til at skabe viden og erkende selvstændigt styrkes.

Det har ikke været nemt for alle lærere og elever at gå ind i disse friere rammer, med ændrede forventninger og tilgange. Vi har observeret lærere såvel som elever, der i stedet for at undersøge sammen overtager og løser problemer for hinanden. Det kan dels skyldes den eksisterende kultur aktørerne imellem, men også et manglende fagligt overskud, der

først kan oparbejdes gennem refleksion over praksis.

Coding Class har haft de kreative og skabende kompetencer i fokus. Derved har diskussioner om og udforskning af og gennem computationelle artefakter i verden ikke fået nævneværdig plads i forløbene. Vi vurderer som nævnt, at dette aspekt udgør et stort potentiale for at knytte an til relevansen for eleverne og styrke elevernes teknologiforståelse. Dette kan fx gøres ved at arbejde med andre produktionstyper end spil. Her er robotteknologi oplagt i skolen (Helms & Majgaard, 2015), idet robotterne omfatter et spejl for vores forståelse af menneske-maskin interaktion og indgår i banebrydende nye teknologier som Tesla bilen, og samtidigt berører en række diverse samfundsmæssige områder (sundhed, økonomi, undervisningssystemet, industri mv.).

Fremme elevernes interesse for it/computationel tænkning

Elevernes it-interesse, interesse for (spil-)design og oplevelse af egne faglige og tekniske kompetencer har stor betydning for elevernes deltagelse i og udbytte af Coding Class aktiviteterne. Elever der oplever store udfordringer, har moderat til lav it-interesse og er uinteresserede i spil, vil være demotiverede. Opgaven er da at tilpasse udfordringsniveau og finde ud af, hvad der indholdsmæssigt kan fange eleverne. Det er generelt set udfordrende for alle eleverne at programmere et spil, hvorfor pædagogisk viden knyttet til computationel tænkning og tilknyttede arbejdspraksisser, der hjælper til at at reducere kompleksitet (abstraktion, inkrementel og iterativ udvikling, samt et vokabular for området) er centrale.

Interessen for it stiger især, når det lykkes at skabe interesse- og praksisfællesskaber i klasserne lærere og elever imellem. Det relationelle aspekt har en afgørende betydning for elevernes motivation.

En del elever mødte projektet med en negativ forventning til fagområdet. De angiv forskellige grunde, såsom at det var svært, og at it ikke var noget for dem. Lærere og kommunale koordinatorer har dog overordnet oplevet at eleverne har været begejstrede for aktiviteterne og vi har vist, hvordan elever selv peger på, at det skabende og kreative arbejde med it, programmering og kodning i undervisningen, har ændret deres interesser og engagement. Dette viser vigtigheden af at dette arbejde bliver obligatorisk for alle elever, hvis alle elever via skolen skal have adgang til informatik og computationel tænkning som almendannende områder.

Dog var eleverne ikke entydigt overbeviste om, at de ville arbejde videre med faget, som de forbandt med spiludvikling. På samme måde havde nogle svært ved at se fremtidsperspektiverne. Vi har peget på et potentiale for at arbejde med transfer af viden knyttet til projektet til andre områder, der er betydningsfulde for eleverne i deres liv.

Ifølge Brennan & Resnick (2012) omfatter computationel tænkning også design- og deltagelsespraksisser. Mange elever var netop begejstrede for at få lov til at udfoldede sig gennem skabende praksisser, som de i høj grad fordybede sig i. Her ses også et potentiale for at involvere elever, der ikke tænder på teknologi i sig selv, men er begejstrede for de

muligheder, de har for at skabe nye udtryk igennem teknologien.

Idet programmering er krævende og udfordrende hænger interessen for aktiviteterne igen sammen med hvilke rammer der skabes omkring aktiviteterne. Intense forløb med høje krav trætter på en måde, der kan demotivere eleverne.

Igangsætte undervisning - skabe pædagogisk viden

Det har vist sig at være effektivt at køre et enkelt grundforløb (Musespillet) med eleverne, men også lærerne har kunnet lukrere på dette forløb. Forløbet er designet så det reducerer kompleksiteten: Alle elever arbejder med samme udfordring, spillet er enkelt og viser, frem for at gennemgå, hvad man overordnet kan i det anvendte program. Med denne introduktion i hånden kunne én lærer bringe forløbet videre til andre lærere og vurderer selv at kunne stå med et lignende forløb i fremtiden. Flere lærere vurderer dog også, at forløbet ikke er nok som introduktion til at køre programmering i grundskolen. Givet at computationel tænkning er et vidensdomæne i sig selv, er dette ikke overraskende. Lærerne er bekendt med at have en faglighed som de bygger deres undervisning op omkring, der skal derfor være en resonans mellem programmeringsaktiviteterne og lærerens faglige baggrund.

I det konkrete forløb viste det sig også at være vigtigt at have viden om spil som genre og spildesign. Dette kan imødekommes ved at indarbejde bevidst imitation af en genrerepræsentant (et udvalgt spil) i egen version. Derved har såvel lærere som elever et eksempel der kan tjene som grænseobjekt for deres forhandling. Denne udfordring vil lærerne møde også, når de arbejder med andre genre end spil.

Det har været en god strategi for lærerne at indgå i et lærende fællesskab med eleverne for derigennem i fællesskab at blive mere teknologividende og -erfarne. Den påkrævede viden om lærernes håndtering af læringsudfordringer; om pædagogiske greb og kontekstviden er gledet i baggrunden i disse korte forløb.

Coding Class instruktørerne har demonstreret en eksplorativ tilgang til vejledning af eleverne, der har støttet op omkring elevernes egne undersøgelsesprocesser, en tilgang som Leo med umiddelbar succes også anvender. Denne tilgang giver eleverne selvtillid og egen drift i undervisningen, hvilket igen skaber et godt læringsmiljø, hvilket er centralt, idet eleverne udfordres af aktiviteterne.

Fælles for de mere erfarne lærere og Coding Class instruktørerne synes også at være at undervisningen er mere problem-, process- og interesseorienteret end mål- og produktorienteret.

Der er et potentiale for at øge fokus på lærernes kompetenceudvikling af disse særlige eksplorative videnspraksisser i lærende fællesskaber med eleverne i undervisningen i det videre arbejde med Coding Class og tilsvarende undervisningsaktiviteter.

Sætte fokus på computationel tænkning i skolen

Der er et potentiale for at videreudvikle koblingen mellem computationel tænkning og de eksisterende fag ved involvering af disse fagligheder. Dette skal dog vægtes i forhold til at der også er behov for at skabe et grundlag for at informatik og computationel tænkning etableres som sit eget vidensdomæne.

Et vigtigt argument for at indarbejde computationel tænkning i skolens faglige grundlag har fra It-Branchens side været at udvikle elevernes interesse for it som et fremtidigt arbejdsområde for eleverne. For at kunne indfri dette potentiale, skal eleverne arbejde med teknologi på forskellige måder og knytte arbejdet til kontekstviden. Her ser vi et potentiale for videreudvikling af Coding Class og tilsvarende initiativer.

Der er argumenter, der går i retning af at placere undervisning i computationel tænkning og informatik tidligt i skoleforløbet både fordi det giver en kontinuitet, fordi elevernes læringskultur kan påvirkes i det tidlige skoleforløb og dermed kan de være mere åbne over for den eksplorative tilgang; og endeligt vurderes eleverne som værende mere åbne over for de store udfordringer tidligt i deres skoleforløb. Med eksemplerne fra Skole M (5. klasse) og K (6. klasse) kan vi se, at sværhedsgraden i aktiviteterne hænger nærmere sammen med elevers og læreres forudsætninger end det handler om klassetrin. Således arbejdede Leos 5. klasse med aktiviteter, der var langt sværere og mere komplekse end 6. klasserne fra Skole K.

En hindring i grundskolen, som den ser ud nu, er at der ikke er tid til faget i strukturen og at lærerne ikke er uddannet inden for informatik og computationel tænkning. Ingen aktører i projektet har nævnt økonomi som en forhindring, idet der eksisterer mange gratis tjenester, som har været omdrejningspunkt for nærværende interventioner. Ønskes arbejde med eksempelvis robotteknologi eller anden hardware vil et økonomisk aspekt dog også være relevant at overveje.

Kom godt i gang

Her er nogle ressourcer, der kan give en hjælp til at komme i gang:

På Code.org kan du finde en hel del kurser - på dansk:
https://studio.code.org/ /

På EMU: Scratch – kom i gang med programmering
http://www.emu.dk/modul/scratch-%E2%80%93-kom-i-gang-med-programmering /

Programmering Naturligvis: Ildsjæle fra Aalborg Kommune:
https://www.facebook.com/programmeringnaturligvis/

Lukket Facebookgruppe - primært for lærere: Scratch i undervisningen
https://www.facebook.com/groups/555918697893834/

EDU21.dk: Josefine Jack Eiby - har en del om programmering på sin blog:
http://edu21.dk/kom-gang-med-programmering-folkeskolen/ /

Grundig introduktion til Scratch:
http://elevakademiet.dk/kodning-boern-hele-familien-intro-scratch/ /…

Dansk side med tips og tricks i Scratch:
http://scratch-paa-dansk.nu/scratch%20i%20skolen.html /

Engelsksproget website for undervisere om brug af Scratch i undervisningen:
http://scratched.gse.harvard.edu/ /

Introduktion til Scratch og Musespillet, som blev brugt i Coding Class, ved Martin Exner:
https://www.youtube.com/watch?v=1-Jb8fKMaIc

Referencer

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. Educational Technology & Society, 19 (3), 47–57.

Brennan, K. and 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 (AERA), Vancouver, Canada.

Caspersen, M. E. (2017 forventet): 4.15 Informatik og Computational Thinking. I: Gymnasiepædagogik - En grundbog. Hans Reitzels Forlag. København.
Danmarks Vækstråd (2016): Rapport om kvalificeret arbejdskraft. København.

Committee for the Workshops on Computational Thinking. (2010). Report of a Workshop on The Scope and Nature of Computational Thinking. National Research Council. Washington, D.C.: National Academies Press.

Committee for the Workshops on Computational Thinking. (2011). Report of a Workshop on the Pedagogical Aspects of Computational Thinking. National Research Council. Washington, D.C.: National Academies Press.

Ejsing-Duun, S., & Misfeldt, M. (2016). Tema 1: Programmering af robotenheder i grundskolen. *Tidsskriftet Læring og Medier (LOM)*, *8*(14).

Ejsing-Duun, S., & Tosca, S. (2017). Design thinking and imitatio in an educational setting. *Digital Creativity*.

Hansbøl, M. (2015): Robot technologies, autism and designs for learning. I: Læring og Medier (LOM). Årg. 8, 14, s. 1-24 24 s.

Helms, N. H., & Majgaard, G. (2016). Tema 1: Robotter i skolen. *Tidsskriftet Læring og Medier (LOM)*, *8*(14).

Kafai, Y. B. (2016). From computational thinking to computational participation in K--12 education. *Communications of the ACM*, *59*(8), 26-27.

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

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

Scratch (2017). What is Computational Thinking:
http://scratched.gse.harvard.edu/ct/index.html (lokaliseret på www 26-05-2017)

Selwyn, N. og Hillman, T. (2017). Computerprogramming in schools… Can we avoid coding each other into a corner? Blogindlæg på bloggen Lärende og IT. Göteborgs Universitet.

Vækstpanel (2017). Danmark som digital frontløber. Anbefalinger til regeringen fra Digitalt Vækstpanel.

Wing, J. M. (2006). Computational thinking. Viewpoint. In: Communications of the ACM. March 2006/Vol. 49, No. 3. S. 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717-3725.

Forfatterne

Mikala Hansbøl

Mikala er ansat som Docent og forskningskoordinator af FoU-miljøet Digitalisering i Skolen (DiS) ved Institut for Skole og Læring (SoL) på Metropol. DiS rummer ca. 30 undervisere fra læreruddannelsen. Forskning og udvikling i DiS har fokus på samspillet mellem it, læring, fag, kreativitet, didaktik, teknologiforståelse og digital dannelse/medborgerskab. Herunder kompetenceudvikling, spredning af it-didaktiske innovationer, og samspillet mellem it i læreruddannelsen, lærerfaglig teknologiforståelse og it i folkeskolen.

Mikala har en ph.d. i uddannelsesforskning, og har beskæftiget sig med forskning i og udvikling af relationer mellem it, læring og uddannelse i ca. 20 år. Mikala er medlem af Undervisningsministeriets Rådgivningsgruppe for teknologi i undervisningen.

Stine Ejsing-Duun
Stine Ejsing-Duun er Lektor på Aalborg Universitet København ved Institut for Kommunikation og Psykologi, K-ILD, ILD-LAB. Hun undersøger, hvordan (teknologiske) designprocesser og design tænkning kan anvendes som undersøgelsesformer, der hjælper os med at skabe en foretrukken fremtid. Hendes ambition er at beskrive, hvordan teknologier giver os mulighed for at overskride os selv. Hendes forskning har på forskellige områder været forbundet med at være kreativ igennem spillende og legende processer med teknologi.

Stine har deltaget en række projekter med teknologi, leg og lærings som omdrejningspunkt de seneste 10 år. Blandt andet har Stine været en del af det to-årige forskningsprojekt Elevernes Egenproduktion og Elevlæring (et demonstrationsskoleprojekt), hvor digital produktion, herunder programmering var omdrejningspunkt for en række interventioner.

Stine Ejsing-Duun har en ph.d.-grad i HCI (Human Computer Interaction) fra DPU/AU. Projektet handlede om (interaktions-)design af lokationsbaserede spil.