Universiteit Leiden
The Netherlands

Opleiding Informatica

Blind leren programmeren:

Een studie naar de Computational Practices van kinderen met een visuele beperking bij het programmeren van muziek in Sonic Pi

Julia Bolt

Begeleiders:
Anna van der Meulen & Felienne Hermans

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

22/01/2021

Samenvatting

In dit onderzoek is gekeken naar welke Computational Practices kinderen met een visuele beperking laten zien wanneer ze leren programmeren met Sonic Pi. De participanten waren tien kinderen van 10-12 jaar oud, uit het speciaal onderwijs, met een visuele beperking. De kinderen werkten in paren aan een programmeeropdracht met Sonic Pi, waarbij zij werden gefilmd, zodat hun gedrag aan de hand daarvan kon worden geobserveerd. Verbale gedragingen zijn getranscribeerd en non-verbale gedragingen gecodeerd. De dataset is geanalyseerd door kwantitatief en kwalitatief onderzoek, met behulp van het model van vier abstractielagen. (De vier abstractielagen zijn de Probleemlaag, Codelaag, Designlaaag, Codelaag en Uitvoerlaag.) De probleemlaag kwam bijna niet voor, de designlaag kwam daarna het minst vaak voor, daarna de codelaag en de uitvoerlaag kwam het vaakst voor. Daarbij wisselden de kinderen veel en snel tussen de verschillende abstractielagen. Dat de programmeertaal van Sonic Pi in het Engels is blijkt een extra barrière te vormen voor de kinderen. De kinderen hadden veel begeleiding nodig. De slechtziende kinderen maakten weinig gebruik van hulpmiddelen en de blinde kinderen gebruikten vooral de screenreader. De doelgroep bleek divers te zijn, onder andere in hoeverre het lukte hardop te denken. Programmeren combineren met muziek maken lijkt de kinderen aan te spreken.

Trefwoorden: Sonic Pi, Computational Thinking, Computational Practices, Inclusief programmeeronderwijs, Kinderen met een visuele beperking

Inhoudsopgave

1 Introductie

'Waarom moeten blinde en slechtziende kinderen leren programmeren? Omdat alle kinderen moeten leren programmeren!' [Her] Snelle ontwikkelingen in digitale technologieën veranderen de vaardigheden die jonge mensen nodig hebben in de toekomst [Pet19]. Programmeeronderwijs is relatief nieuw. Daarom zijn er veel open vragen zoals: 'Wat is de beste leeftijd om te leren programmeren?', 'Welke concepten brengen kinderen het meest in verwarring?', 'Hoe moet een leraar programmeren leren als ze er zelf niet veel van weten?'. Dergelijke vragen naar programmeeronderwijs worden in huidig onderzoek beantwoord [Lab]. Programmeren is een uitstekende carrièreoptie voor blinde kinderen, én een manier waarop ze zichzelf kunnen uiten. Maar de bestaande programma's daarvoor zijn vaak niet geschikt voor blinden en slechtzienden [ea18a]. Om programmeeronderwijs ook toegankelijk te maken voor blinde en slechtziende kinderen, is het belangrijk om te onderzoeken of de bestaande programma's geschikt zijn voor deze doelgroep. Na deze inventarisatie kunnen er richtlijnen opgesteld worden waarmee leraren programma's kunnen beoordelen op inclusiviteit. Ook kunnen er voorstellen komen om programma's aan te passen, om vervolgens zowel door ziende als visueel beperkte kinderen die aangepaste programma's te laten testen. Het is belangrijk om specifiek te kijken naar talen die inclusief zijn voor blinden en slechtziende kinderen, in plaats van talen die voor hen ontworpen zijn, zodat alle kinderen samen kunnen programmeren, met programma's die iedereen kan gebruiken [oz].

Voor dit bachelorproject is ervoor gekozen om kinderen in de leeftijdsgroep van 10-12 jaar te onderzoeken en het materiaal Sonic Pi daarvoor te gebruiken. Sonic Pi is een programma waarmee muziek geprogrammeerd kan worden. Wellicht ten overvloede, Sonic Pi is niet speciaal ontwikkeld voor kinderen met een visuele beperking. Een verdere afbakening is, dat van de verschillende aspecten van het leerproces waar op gelet kan worden, in dit onderzoek de nadruk wordt gelegd op Computational Practices. Dit wordt verder uitgelegd in Sectie 2. Het doel van het onderzoek is om meer inzicht te krijgen in hoe kinderen met een visuele beperking leren programmeren. Daarvoor is Sonic Pi een interessant programma om te gebruiken, vanwege de auditieve output. Een waardevolle manier om te beoordelen hoe kinderen met een visuele beperking werken en leren met het programma, is door te kijken naar de Computational Practices die de kinderen laten zien. Daaruit volgt de volgende onderzoeksvraag:

Welke Computational Practices laten kinderen met een visuele beperking zien wanneer ze leren programmeren met Sonic Pi?

Voor het vormen van een volledig beeld, is gekozen voor een mixed-method, waarbij er kwantitatief en kwalitatief onderzoek gecombineerd wordt. Om de Computational Practices gestructureerd te kunnen observeren wordt gebruik gemaakt van het model van de vier abstractielagen [ea19]. In het theoretisch kader, Sectie 2, wordt dit model verder toegelicht.

Dit hoofdstuk bevat de introductie; Sectie 2 bevat het theoretisch kader, waarin bestaand onderzoek wordt besproken gerelateerd aan Sonic Pi, Computational Practices en programmeren voor kinderen met een visuele beperking; Section 3 beschrijft de onderzoeksmethode; Section 4 beschrijft de resultaten; Section 5 beschrijft de conclusies en discussiepunten.

Tot slot wil graag mijn begeleiders Anna van der Meulen en Felienne Hermans van het Leiden Institute of Advanced Computer Science [lia] bedanken voor de uitstekende wijze waarop ze mijn hele onderzoeks- en schrijfproces hebben begeleid. Ik heb er veel van geleerd en ik kijk ernaar uit om in de master me verder te verdiepen in Computer Science and Education.

2 Theoretisch Kader

2.1 Sonic Pi

Het programmeeronderwijs voor kinderen neemt toe en daarvoor zijn programma's nodig, ter ondersteuning. Er zijn al verschillende programma's ontwikkeld om kinderen te leren programmeren. Een veelgebruikt programma daarvoor is bijvoorbeeld Scratch. Bij Scratch kunnen kinderen met blokken slepen om een code te maken die uiteindelijk vaak een beeldende output geeft, zoals een poppetje dat over het scherm beweegt. Zo zijn er meer programma's, waarbij kinderen blokken gebruiken om te leren programmeren, maar ook zijn er programma's waarbij er gebruik gemaakt wordt van tekstuele code. Daarnaast zijn er veel verschillende soorten van output mogelijk.

Sonic Pi is ook ontwikkeld om schoolkinderen te leren programmeren, maar bij Sonic Pi gebeurt dat door muziek te maken [Aar16, Pet19]. De output is dus altijd auditief. Verder werkt Sonic Pi met tekstuele code. Veel leerlingen blijken te waarderen dat ze bij Sonic Pi te maken krijgen met tekstuele code [Aar16]. Voor Sonic Pi is er niet een nieuwe programmeertaal gemaakt, maar het programma gebruikt een variant van Ruby en de programmeertaal is in het Engels. Ruby is niet eerder gebruikt voor educatie. Wel is het een relevante programmeertaal, want bijvoorbeeld Twitter is in deze taal geïmplementeerd [Aar16]. Op deze manier wordt er dus een programmeertaal gebruikt in de klas die ook echt commercieel gebruikt wordt. Dat is bijvoorbeeld anders bij Scratch, waar er gebruik gemaakt wordt van felgekleurde blokjes puur bedoeld voor het onderwijs [Aar16].

Niet alles aan Sonic Pi is anders dan Scratch. Sonic Pi deelt namelijk het "low-floor and high-ceiling" ontwerp van bijvoorbeeld Scratch. Sonic Pi is ontworpen om iedereen de mogelijkheid te geven om te leren programmeren in een interdisciplinaire manier, door het coderen van muziek [Pet19]. Zo kan elke leerling, leerlingen met weinig tot geen ervaring, maar ook leerlingen met wat meer ervaring die wat meer uitdaging willen, met Sonic Pi werken.

De vraag is of Sonic Pi daadwerkelijk een goed programma is om kinderen te leren programmeren. Bij eerder onderzoek [Aar16] is muziek en live coding gebruikt om kinderen te motiveren en lessen te structureren. De pedagogische inhoud was gerelateerd aan computer science. Lessen waar er geen duidelijk muzikaal doel was, maar meer gestructureerd waren om een specifiek computer science concept over te brengen, waren veel minder effectief. Ook bleek dat leren programmeren door muziek het meest effectief was wanneer er samengewerkt en geïmproviseerd werd. In dat onderzoek, waar de kinderen in het Britse systeem in jaar 9 zaten, waren de reacties van de leerlingen erg positief [Aar16]. Deze kinderen zullen waarschijnlijk ongeveer 13 jaar oud zijn geweest. De bestaande studies werken vaak met live-coding, waar leerlingen muziek maken voor een publiek, als een optreden. Het past niet in de traditionele beoordeling van muziek, omdat het compositie, optreden en programmeren combineert. Dan staan studenten onder de druk dat ze een set van

vereiste vaardigheden moeten laten zien, door op te treden. Live-coding optredens kunnen een te stressvolle ervaring zijn voor een introductie in muziekcompositie en programmeren [Pet19]. Dit probleem kan ontweken worden door de focussen op muziek compositie en niet op live optreden.

Sonic Pi lijkt dus een zeer geschikt programma om kinderen mee te leren programmeren, op een interessante interdisciplinaire manier, door het gebruik van muziek. Een manier om dat te toetsen is door te kijken naar de Computational Practices die kinderen ontwikkelen wanneer zij werken met het programma.

2.2 Computational Practices

Om uit te leggen wat Computational Practices zijn, moet eerst gekeken worden naar Computational Thinking[ea17]. Voor Computational Thinking zijn allerlei verschillende definities geformuleerd. Er zijn ook weer definities geformuleerd op basis van meerdere verschillende publicaties gerelateerd aan de definitie van Computational Thinking. Computational Thinking wordt bijvoorbeeld beschreven als bestaande uit zeven aspecten[ea17]:

1. een denkproces
2. abstractie
3. decompositie
4. algoritmisch ontwerp
5. evaluatie
6. generalisatie
7. automatisering

The Netherlands Institute for Curriculum Development (SLO) heeft een eigen definitie van Computational Thinking gepubliceerd. Daarin worden de volgende termen toegevoegd: parallellisatie, simulatie, datacollectie, data-analyse en datarepresentatie [Ken19]. De eerste definitie is meer geworteld in academische gewoonten, maar de tweede definitie lijkt meer algemeen aangenomen. Beide definities hebben een aantal belangrijke overeenkomsten. Er kan daarom daaruit afgeleid worden dat Computational Thinking in elk geval gaat over abstractie, decompositie, algoritmisch denken of ontwerpen en automatisering. De literatuur suggereert dat programmeren gebruikt kan worden om Computational Thinking te onderwijzen [ea17].

Hieruit blijkt nog niet wat Computational Practices zijn. In ander onderzoek [BMR12], worden Computational Practices genoemd als onderdeel van Computational Thinking. Daaruit blijkt namelijk dat Computational Thinking bestaat uit drie dimensies:

1. Computational Concepts, de concepten waar ontwerpers zich mee bezig houden wanneer zij programmeren, zoals iteratie en parallellisme;

2. Computational Practices, de handelingen die ontwerpers ontwikkelen wanneer zij zich bezighouden met de computational concepten, zoals debugging projecten of het hergebruiken (remixing) van het werk van anderen;

3. Computational Perspectives, de perspectieven die ontwerpers vormen over de wereld om hun heen en over henzelf.

Computational Practices beschrijven de processen van de constructie, de ontwerppraktijken. Het focust op het proces van denken en leren, dat gaat dus verder dan wat je leert, het gaat om *hoe* je leert. Er zijn vier sets van practices geobserveerd [BMR12]:

1. "being incremental and iterative", het ontwerpen van een project is een proces van aanpassing, waarin het plan zou kunnen veranderen als reactie op het benaderen van een oplossing in kleine stapjes. Het is een iteratieve cyclus van bedenken en bouwen, eerst een beetje ontwikkelen, daarna uitproberen, daarna verder uitproberen, gebaseerd op de ervaringen en nieuwe ideeën.;

2. "testing and debugging", er zijn verschillende strategieën voor het testen en voor debugging. Hierbij kan gedacht worden aan door code heen lezen, code opnieuw schrijven, een voorbeeldcode zoeken die werkt of iemand anders het probleem uitleggen.;

3. "reusing and remixing", verder bouwen op de code van andere mensen ondersteunt de ontwikkeling van het kritisch lezen van code.;

4. "abstracting and modularizing", het bouwen van iets groots door een collectie van kleinere delen te combineren.

Computational Thinking is dus een metacognitieve vaardigheid die aangeleerd wordt aan leerlingen om ze voor te bereiden op de nieuwe vaardigheden die jonge mensen in de toekomst nodig gaan hebben door de veranderingen in digitale technologieën. Computational Thinking omvat de verschillende denkgewoonten van het oplossen van problemen voor computers.[Pet19].

Er zijn nog veel overwegingen over wat Computational Thinking precies omvat, maar daaraan gerelateerd is het een belangrijke vraag hoe we Computational Thinking (bij kinderen) kunnen beoordelen. De ideale leeftijd om te beginnen met het ontwikkelen van Computational Thinking lijkt rond de 11 à 12 jaar te zijn [Pet19]. Het ontwikkelen van Computational Thinking bij kinderen in kaart brengen is erg lastig. Om hierbij te helpen worden onder andere de volgende oplossingen aanbevolen: check-ins at multiple points [BMR12] en think-aloud [ea05]. De think-aloud methode wordt gebruikt om onderzoekers meer inzicht te geven in de cognitieve processen van deelnemers die een bepaalde taak uitvoeren, in plaats van dat de verzamelde data alleen bestaat uit het eindproduct. De deelnemers van een onderzoek worden dan gevraagd om alles hardop te zeggen wat in hen opkomt. Op die manier worden de denkprocessen zo expliciet mogelijk, tijdens het uitvoeren van een opdracht. Normaal gesproken werken de deelnemers individueel aan een opdracht, maar er is ook een variant van think-aloud, genaamd constructieve interactie, waarin er in duo's gewerkt wordt, die samenwerken op een computer. Kinderen kunnen namelijk meer moeite hebben met het volgen van de instructies voor een standaard think-aloud test, constructieve interactie wordt gesuggereerd als evaluatie methode wanneer er een onderzoek wordt gedaan met kinderen. Deze variant blijkt gepast te zijn voor kinderen, omdat op deze manier het hardop denken natuurlijker is

[ea05]. Het framework van Brennan en Resnick [BMR12] voorziet ons van een holistische manier om Computational Thinking te beoordelen voor het Sonic Pi platform [Pet19]. Er worden drie benaderingen genoemd voor het beoordelen van de ontwikkeling van Computational Thinking in jonge mensen, die zich bezighouden met ontwerpactiviteiten in Scratch. Er komt naar voren dat "process-in-action" de voorkeur heeft boven "process-via-memory" [BMR12]. Een voorbeeld van process-in-action zou kunnen zijn dat de kinderen zijn gefilmd tijdens de "actie" en met behulp daarvan worden geobserveerd. Process-via-memory zou zijn wanneer er bijvoorbeeld na afloop een interview gehouden zou worden. Er worden ook een aantal suggesties gedaan voor het beoordelen van Computational Thinking door programmeren. Een van de suggesties is om op meerdere punten in de tijd te kijken, de eerder genoemde "check-ins at multiple points", omdat het geen binaire staat is, het er is wel of niet op een bepaald punt in de tijd, maar het is in ontwikkeling. De benadering voor de beoordeling zou dus moeten beschrijven waar een leerling was, is en waar het heen zou kunnen gaan. Een andere suggestie is "illuminating the process" [BMR12]. Dat kan worden bereikt door heel precies van minuut tot minuut te observeren wat er gebeurt.

Tenslotte, een benadering die een concrete aanwijzing geeft om Computational Practices bij kinderen te observeren is het model van vier lagen van abstractie [ea19]. Abstractie wordt, zoals eerder is genoemd, gezien als een essentieel aspect van Computational Thinking [ea17, ea19]. Omdat abstractie refereert naar een cognitief proces, is het moeilijk om abstractie te operationaliseren in de zin van observeerbaar gedrag. De vier lagen die daarbij kunnen helpen zijn: de probleemlaag, de designlaag, de codelaag en de uitvoerlaag. Het verschil tussen de code- en designlaag is gebaseerd op het verschil in taal dat gebruikt wordt om de gesuggereerde oplossing te beschrijven. Op designniveau is de oplossing beschreven met behulp van menselijke leesbare taal, zonder een referentie naar een programmeertaal. Bij de codelaag is het design vertaald naar regels code. Door audio en video-opnames te analyseren, kunnen we verbaal en non-verbaal gedrag te categoriseren en een observationeel schema verkrijgen. Vervolgens kan er patroonanalyse gedaan worden, bijvoorbeeld om inzicht te krijgen over hoe leerlingen zich bezighouden in het proces van abstractie in de loop van de tijd en wat voor patronen ontstaan tijdens die tijd [ea19].

2.3 Programmeren voor kinderen met een visuele beperking

In Nederland zijn ruim 320.000 mensen slechtziend of blind. Je spreekt van een visuele beperking als iemand niet of slechts gedeeltelijk kan zien. Er zijn veel verschillende vormen van slechtziendheid en oogaandoeningen. Iemand is slechtziend bij een lage gezichtsscherpte, een verkleind gezichtsveld of bij overgevoeligheid voor licht. Sommige mensen zien allemaal donkere vlekken en andere zien alleen wat recht voor ze staat, maar niets aan de zijkant. Blind zijn betekent dat iemand niet of minder ziet dan 5% of dat het gezichtsveld is beperkt tot minder dan 10 graden. Blind zijn houdt is niet altijd in dat iemand helemaal niets meer ziet. Sommige mensen zien nog wel het verschil tussen licht en donker [Vis]. Doordat er zoveel verschillende vormen van slechtziendheid zijn, zijn er ook diverse voorkeuren in het gebruik van technologie. Het ene kind heeft mogelijk genoeg aan een zoomfunctie, terwijl het andere kind een screenreader nodig heeft bijvoorbeeld.

Programmeren kan uitdagend zijn om te leren en voor kinderen met een visuele beperking zijn er meerdere extra obstakels in het leerproces. Veel moderne programmeeromgevingen zijn ontoegankelijk voor kinderen met een visuele beperking. Ze zijn bijvoorbeeld moeilijk of onmogelijk toegankelijk

voor een screenreader. Een literatuuroverzicht [ea18a] heeft meerdere strategieën geïdentificeerd die zijn gebruikt om leren programmeren toegankelijk te maken voor leerlingen met een visuele beperking. Deze strategieën kunnen verdeeld worden in de volgende categorieën; auditieve en haptische feedback, op tekst gebaseerde talen toegankelijk maken, op blokken gebaseerde talen toegankelijk maken en fysieke artefacten [ea18a]. Het programma Sonic Pi werkt met een op tekst gebaseerde taal en met auditieve output, dus die twee categorieën zijn in combinatie met Sonic Pi het meest relevant, hoewel kinderen van basisschoolleeftijd vaker te maken krijgen met talen gebaseerd op blokken, als introductie op programmeren.

In het programmeeronderwijs aan kinderen met een visuele beperking zijn een aantal dingen van belang. Ten eerste is een gebruikelijk thema in de literatuur, de moeite die leerlingen met een visuele beperking hebben met het krijgen van overzicht over de structuur van hun code. Veel van het onderzoek in dit gebied focust op de evaluatie van interventies voor middelbare schoolleerlingen met een visuele beperking, met gelimiteerde aandacht voor het leerproces. De meerderheid van het onderzoek gaat over op tekst gebaseerde programmeertalen, terwijl de meeste inleidende programmeerlessen voor beginnende leerlingen gebruik maken van op blokken gebaseerde talen. Er is dus meer onderzoek nodig naar mogelijke strategieën voor het introduceren van kinderen met een visuele beperking in het basisonderwijs aan programmeren en de bijbehorende leerprocessen [ea18a]. Ten tweede is de keuze voor een taal belangrijk, wanneer onderwezen wordt met op tekst gebaseerde programmeertalen. Het is aan te raden een taal te kiezen die specifiek is ontwikkeld voor leerlingen met een visuele beperking, of een taal met simpele syntaxis en een gelimiteerd gebruik van niet-alfanumerieke karakters, zoals Ruby [ea18a]. Niet-alfanumerieke karakters zijn bijvoorbeeld haakjes en accolades. Die worden in sommige talen veel gebruikt, maar in het geval van Sonic Pi heeft de taal dat niet. Het programma Sonic Pi is niet speciaal ontwikkeld voor kinderen met een visuele beperking, maar de taal van Sonic Pi is gebaseerd op Ruby en dat is waarschijnlijk een voordeel voor kinderen met een visuele beperking. De programmeertaal die gebruikt wordt door Sonic Pi hij is wel in het Engels, wat een nadeel kan zijn voor Nederlandse kinderen ten opzichte van Engelstalige kinderen. Ten derde is het belangrijk een programmeeromgeving te kiezen die toegankelijk is en waar makkelijk in te navigeren is met bijvoorbeeld een screenreader. Als een passende omgeving niet beschikbaar is, kan een simpele tekstverwerker gebruikt worden, hoewel het een uitdaging kan zijn zonder de debugging tools [ea18a]. Het is niet geheel duidelijk of Sonic Pi toegankelijk is met een screenreader, maar er is een mogelijkheid om in een gewone tekstverwerker te typen en vervolgens de code te kopiëren naar Sonic Pi. Leerlingen met een visuele beperking hebben vaak moeite met het begrijpen van de structuur van code geschreven in op tekst gebaseerde talen, een manier om hen te ondersteunen is door voorbeeldcode in braille te verstrekken [ea18a]. Ten vierde kan het kiezen van een passend thema voor programmeeractiviteiten het toegankelijk en innemend maken voor leerlingen met een visuele beperking [ea18a]. Mogelijk spreekt het maken van muziek de leerlingen aan, meer dan bijvoorbeeld programma's waarbij de output visueel is.

Alles bij elkaar genomen zijn er dus verschillende redenen om meer onderzoek te doen naar het programmeeronderwijs voor basisschoolleerlingen met een visuele beperking. Daarvoor is Sonic Pi met name een interessant programma om te gebruiken. Een waardevol aspect om naar te kijken, om te beoordelen hoe kinderen met een visuele beperking werken en leren programmeren met het programma, is door te kijken naar de Computational Practices die de kinderen laten zien.

3 Methode

3.1 Participanten

We gebruikten een stukje data uit een groter project. Aan dat onderzoek hebben groepen 3 t/m 8 binnen het speciaal onderwijs aan kinderen met een visuele beperking, op scholen van Visio en Bartiméus meegewerkt. Zo konden er meer materialen onderzocht worden, naast Sonic Pi, die geschikt waren voor verschillende leeftijden. De leerkrachten waren geïnteresseerd in Computational Thinking in hun lessen, maar hadden geen achtergrond in IT of computer science. De scholen waren in verschillende plaatsen in het land, niet uit de allergrootste steden, maar ook niet uit kleine dorpjes. Er is een selectie gemaakt van 10 kinderen tussen de 10 en 12 jaar, met een gemiddelde leeftijd van 11 jaar. Het waren 4 jongens en 6 meisjes, van twee verschillende scholen, waarbij de visuele beperking sterk verschilt. De selectiecriteria bestonden uit: een representatieve verdeling van slechtziende en blinde kinderen, verschillen in niveau en algemene beleving, maar ook alleen paren bij wie er redelijk goed met het programma gewerkt kon worden, omdat er anders niet voldoende sprake is van Computational Practices. De paren waren voornamelijk gekoppeld op grond van welke leerlingen er goed met elkaar konden samenwerken volgens de docent.

Paar	**Leeftijd**	**Sekse**	**Visus**
1	12	m	slechtziend
	11	m	slechtziend
2	12	m	slechtziend
	11	m	slechtziend
3	11	v	slechtziend
	11	v	slechtziend
4	12	v	slechtziend/braille
	11	v	braille
5	10	v	slechtziend
	10	v	braille

Tabel 1: Achtergrondinformatie over de geselecteerde participanten

3.2 Procedure

De ouders van kinderen in de deelnemende klassen kregen een informatiebrief waarin werd uitgelegd dat de klas van hun kind programmeerlessen zou krijgen en dat er in het kader van die programmeerlessen werd gevraagd of een aantal kinderen aan het onderzoek wilden meedoen. Er werd toegelicht dat de hele klas mee zou doen aan de programmeerles, die zou worden gepresenteerd als een reguliere les waar de eigen leerkracht bij is, en dat het optioneel is dat een kind meedoet aan het onderzoek. Dit betekent dat er alleen data werden verzameld en verwerkt van kinderen van wie de ouders hier specifiek toestemming voor hebben gegeven. Om deze data volledig en nauwkeurig te verwerken werden er video-opnames gemaakt tijdens de programmeerles. Het werd aan de ouders uitgelegd dat deze opnames alleen gebruikt zouden worden om beelden van de deelnemende kinderen te bekijken en te verwerken, dat het wel mogelijk was dat een niet deelnemend kind in beeld kwam, maar dat

er niets met deze beelden gedaan zou worden. Verder werd uitgelegd hoe de data vertrouwelijk verwerkt zou worden, dat deelname vrijwillig was en er op elk moment gestopt zou kunnen worden met het onderzoek, zowel wanneer de ouder als het kind dit zou aangeven. Ouders werden gevraagd toestemming te geven voor de video-opnames die tijdens de programmeerlessen zouden plaatsvinden en werden gevraagd apart toestemming te geven voor deelname van hun kind aan het onderzoek. Het werd niet verwacht dat ouders bezwaar hebben tegen de programmeerles zelf aangezien deze gezien zou kunnen worden als een gewone les onder aanwezigheid van de eigen leerkracht. Als ouders bezwaar zouden hebben tegen de video-opname werd er eerst overlegd of ouders akkoord gingen als de beelden van hun kind direct vervaagd werden, en werd er anders in overleg met de school en leerkracht een alternatief voor de betreffende leerling gezocht. Achtergrondinformatie over de kinderen werd via de school opgevraagd, waarvoor de ouders toestemming gaven.

De kinderen zelf kregen aan het begin van de eerste programmeerles een brief met informatie over het onderzoek en een instemmingsformulier. De brief werd hardop voorgelezen door de onderzoeker wel de kinderen konden meelezen. Het werd uitgelegd wat de programmeerlessen inhielden en dat de onderzoekers aanwezig zouden zijn, omdat ze geïnteresseerd waren in hoe de verschillende programmeermaterialen werken. Ook werd aangegeven dat er gefilmd zou worden, zodat de onderzoekers dit later nog goed zouden kunnen nakijken. Het werd benadrukt dat het niet gaat om hoe goed de kinderen het zouden doen, maar hoe goed de materialen het zouden doen. Er werd ook aangegeven dat de ouders een brief hadden gehad, er werd gelegenheid gegeven tot vragen stellen en vervolgens konden de kinderen instemmen.

Deelnemende klassen kregen drie keer een les van een uur, verspreid over enkele weken. De programmeerles bestond uit een klassikaal gedeelte en een begeleid gedeelte waarbij de kinderen in paren aan een opdracht gingen werken.
Tijdens het klassikale gedeelte kregen de kinderen een introductie van Sonic Pi en muziek programmeren.
Vervolgens gingen de kinderen hun eigen stukje muziek maken. Daarbij werden de kinderen gefilmd. De onderzoeker en een aantal onderzoeksassistenten waren aanwezig. Zij hadden van tevoren een training gekregen van de onderzoeker in de verzameling en verwerking van observatiedata tijdens constructieve interacties van kinderen. Tijdens de programmeerles observeerden de onderzoeker en onderzoeksassistent de paren kinderen die deelnemen aan het onderzoek en maakten aantekeningen van opvallendheden. De kinderen konden in het duo zelstandig aan de slag, maar ze konden ook stap voor stap begeleid blijven worden door de testleider.

3.3 Instrumenten

3.3.1 Constructieve interactie

De interactie van kinderen met een visuele beperking met verschillende programmeermaterialen werd onderzocht met de methode van constructieve interactie, een vorm van de think-aloud methode. Door kinderen in paren met een programmeermateriaal aan het werk te zetten werd een natuurlijke situatie gecreëerd waarin de samenwerkingen interactie van de kinderen geobserveerd kon worden en waarin zij automatisch hard op naar elkaar hun gedachten en beslissingen zouden uitspreken. De onderzoeksassistenten stimuleerden dat ook.

3.3.2 Opdrachten

De programmeerles bestond uit een klassikaal gedeelte en een begeleid gedeelte waarbij de kinderen in paren aan een opdracht gingen werken. De testleiders, die de kinderen begeleidden, hadden van tevoren een training gehad daarvoor.
Tijdens het klassikale gedeelte kregen de kinderen een introductie van Sonic Pi en muziek programmeren. In deze les leerden de kinderen wat programmeren is en waar een muziekstuk allemaal uit bestaat, zoals bijvoorbeeld bepaalde instrumenten, noten, tempo en dynamiek. Eventueel werden ook herhalingen besproken. Verder werd er kennisgemaakt met de interface van Sonic Pi en enkele soorten code die gebruikt konden worden. Eventueel werd daarbij ook gebruikgemaakt van een tekstverwerker en screenreader ernaast.
Er waren twee varianten van de opdracht. De opdracht van niveau A begon met het spelen van noten van verschillende hoogte, waarbij vervolgens de noten tegelijk of na elkaar afgespeeld moesten worden. Daarna werd het tempo aangepast en als laatst werden ook instrumenten gekozen. De opdracht van niveau B sloot aan bij de les waarbij uitleg was gegeven over herhalingen, dan kregen de kinderen ook de opdracht herhalingen toe te voegen in hun stuk. In beide gevallen konden kinderen optioneel ook nog werken met "samples" en "bugs". De selectie van 10 kinderen voor dit onderzoek, hebben allemaal de opdracht van niveau A gedaan.

3.4 Data verwerking

3.4.1 Coderen en transcriberen

Tijden het maken van de opdracht werden de kinderen gefilmd. In de video waren beide kinderen zichtbaar, maar het beeldscherm niet. Na afloop van de lessen werden de videobeelden gepseudonimiseerd getranscribeerd. Er is hierbij gebruikgemaakt van een observatiechecklist. Zo konden we observaties omzetten in concrete gedragingen die gecodeerd konden worden. Deze gedragingen konden zowel herleid worden uit wat de kinderen doen als uit wat zij zeiden tijdens de hun interactie met elkaar en met het materiaal. De van tevoren gedefinieerde gedragingen werden aangevuld met andere open, niet geanticipeerde observaties. De samengevoegde ruwe data werden vervolgens verder verwerkt door middel van open, axiaal en selectief coderen. De observatiechecklist bestond uit 17 verschillende gedragscategorieën, zoals bijvoorbeeld:

1. Zelfstandig toegang tot het materiaal
2. Aanwezigheid samenwerking
3. Creativiteit en exploratie
4. Positieve ervaring materiaal

Om de onderzoeksvraag te beantwoorden, waren we alleen geïnteresseerd in de categorieën van de abstractie levels en de Computational practices. De overige categorieën waren van nut voor het grotere project. De codes voor gedragingen binnen de categorie *Computational practices* bestonden uit:

1. Schrijven code/bouwen programma

2. Uittesten stukje of hele code/programma
3. Lezen of volgen gemaakte code, auditief, visueel of tactiel
4. Opmerken fout tijdens uitvoer programma
5. Zoeken naar fout in code/programma
6. Aanpassen code/programma om fout te herstellen

De codes voor gedragingen binnen de categorie *Abstractie level: probleemlaag* bestonden uit:

1. Startpunt/uitgangspunt opdracht bespreken
2. Eindpunt bespreken

De codes voor gedragingen binnen de categorie *Abstractie level: designlaag* bestonden uit:

1. Kind beschrijft plan in termen van opeenvolgende stappen
2. Kind beschrijft aantal stappen dat gemaakt moet worden

De codes voor gedragingen binnen de categorie *Abstractie level: codelaag* bestonden uit:

1. Schrijven code
2. Lezen code
3. Code aanpassen

De codes voor gedragingen binnen de categorie *Abstractie level: uitvoerlaag* bestonden uit:

1. Code runnen
2. Output volgen
3. Kind relateert de uitkomst aan de voorspelde uitkomst
4. Kind voorspelt de uitkomst van (een stukje van) het programma door te redeneren hoe het programma geïnterpreteerd wordt, zonder programma al echt uit te voeren

3.4.2 Data analyse

Om de data verder te analyseren, maakten we gebruik van een mixed-method, een combinatie van kwantitatief en kwalitatief onderzoek. Voor het kwantitatieve deel gebruikten wij de methode die gebruikt is in het onderzoek [ea19] naar het model van de vier lagen van abstractie. Nadat het gedrag van de kinderen gecategoriseerd was in de verschillende lagen van abstractie en Computational Practices, analyseerden we de resultaten om te zien of er patronen geïdentificeerd konden worden. We verwachtten dat een analyse inzicht kon geven in de ontwikkeling van abstractie tijdens een algoritmische taak, uitgevoerd door jonge leerlingen. Voor het kwalitatieve deel gingen we met behulp van de transcripties proberen wat meer interpretatie te geven bij de resultaten van het

kwantitatieve deel. We maakten hierbij alleen gebruik van de gedragingen gecodeerd bij Computational Practices en de abstractielevels.

Specifiek hebben we eerst per duo gekeken naar hoevaak elke abstractielaag voorkwam en op welke manier de verschillende lagen zich afwisselden. Om dit duidelijk te representeren hebben we de verschillende lagen gekwantificeerd en lijngrafieken ervan gemaakt. Voor het kwantificeren hebben we Excel gebruikt en voor de grafieken SPSS. Door de verschillende grafieken naast elkaar te leggen konden we patronen en opvallendheden ontdekken.

Bij een duo viel de camera uit, terwijl ze nog ongeveer halverwege de programmeeropdracht waren. We missen daardoor een aantal minuten aan data. We hebben daarom besloten om voor dit paar twee grafieken te maken, een grafiek van het gedeelte voordat de camera uitviel en een grafiek van het gedeelte nadat de camera was uitgevallen. Om te voorkomen dat er een vertekend beeld ontstaat hebben we de grafieken ongeveer half zo breed gemaakt als de overige grafieken, zodat de breedte per tijdstap zo goed mogelijk overeen komt voor alle grafieken. Op deze uitzondering na hebben we voor elk duo één grafiek gemaakt.

Om het beeld waar de grafieken ons van voorzien verder te completeren zijn er ook een aantal gegevens overzichtelijk in een tabel weergegeven. Daarmee konden we namelijk bepaalde gegevens makkelijker aflezen, dan we konden met de grafieken. Bijvoorbeeld de exacte frequentie waarmee de verschillende abstractielagen voorkwamen. Om de verschillende paren goed te kunnen vergelijken hebben we ook een tabel gebruikt met de percentages waarin de verschillende abstractielagen voorkomen, in plaats van de absolute aantallen.

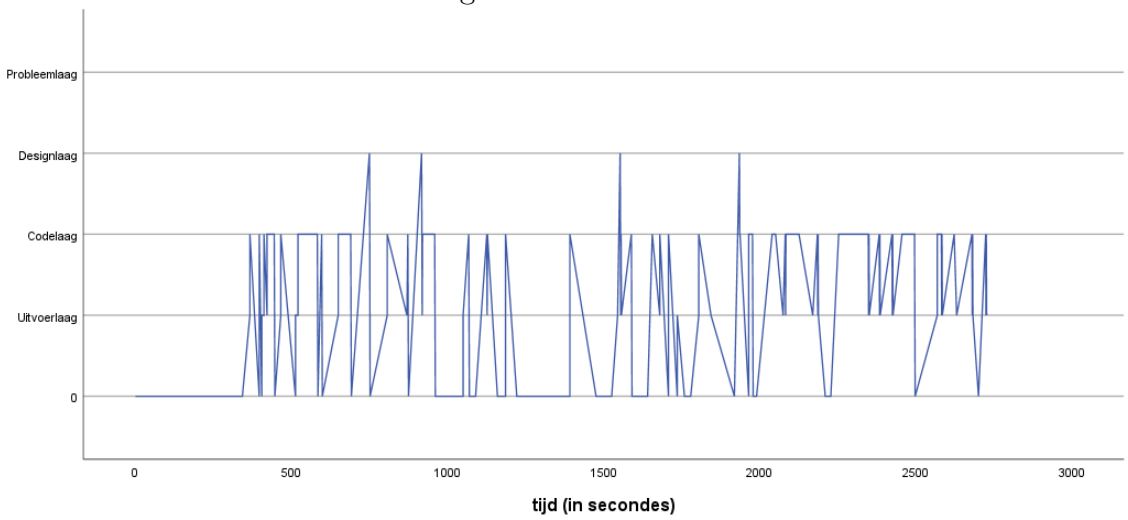
Met de verschillende grafieken en tabellen, waarmee de gegevens per paar duidelijk zijn weergegeven, hadden we genoeg informatie om verder te kunnen gaan met het kwalitatieve deel van de data analyse. Met behulp van de transcripties en coderingen bij de verschillende gedragscategorieën konden we de opvallendheden vanuit de kwantitatieve data verder onderbouwen en proberen er meer interpretatie bij te geven. Eerst hebben we geprobeerd per paar een algemeen beeld te krijgen van het proces. Vervolgens hebben we gekeken of de opvallendheden uit de kwantitatieve data verder uitgelegd konden werden aan de hand van het algemene beeld van het betreffende paar.

4 Resultaten

4.1 Kwantitatieve data analyse

Hieronder zijn van elk paar de abstractielagen weergegeven in een grafiek. Zo ontstaat een overzicht van het proces van abstractie (zie figuur 1 t/m 5). De grafieken geven een indicatie van hoe de kinderen wisselden tussen de verschillende lagen en hoeveel tijd zij besteedden aan een laag, voordat zij wisselden naar een andere laag. Er zijn ook momenten waarop de kinderen gedrag vertoonden dat niet gerelateerd was aan de programmeeropdracht. Dat wordt in de grafieken aangegeven met de onderste horizontale lijn, genaamd 0. Daaronder is in tabel 2 en 3 voor elk duo aanvullende informatie te vinden, namelijk de exacte frequentie waarmee de abstractielagen voorkomen, wat niet direct af te lezen is uit de grafieken. Het aantal keren dat de verschillende abstractielagen zijn geobserveerd bij de verschillende paren wordt absoluut en ook relatief gegeven om het goed te kunnen vergelijken. Daarbij is tevens het gemiddelde gegeven.

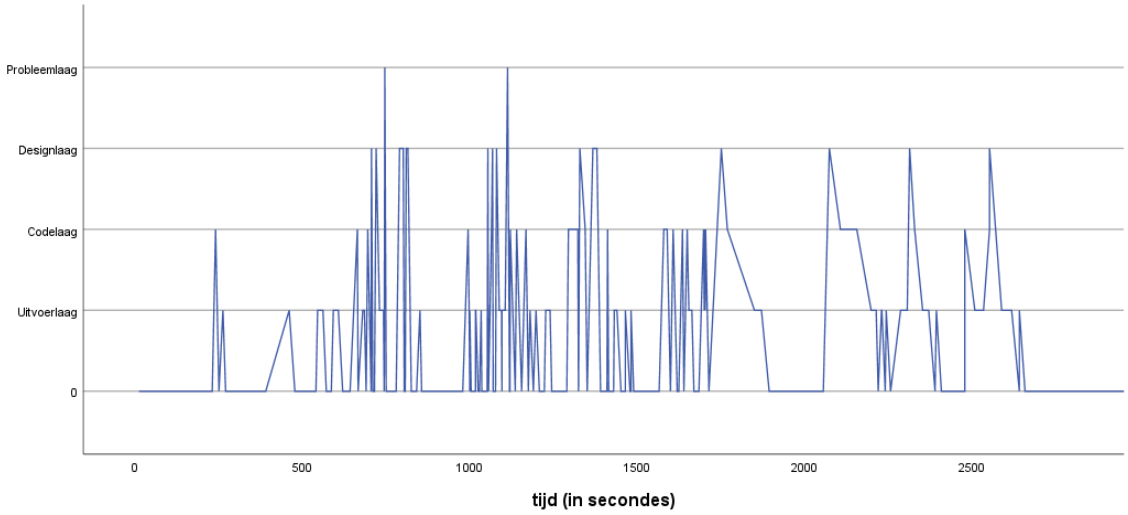
Figuur 1: Paar 1



Bij het eerste paar is te zien dat de kinderen voornamelijk wisselden tussen de codelaag, uitvoerlaag en overig gedrag. De designlaag komt nauwelijks voor en de probleemlaag zelfs helemaal niet. Het paar liet vanaf ongeveer 400 seconden de verschillende astractielagen zien en leek daarna vrij constant in het wisselen tussen de verschillende lagen. Van begin tot eind bleef het paar vooral veel wisselen tussen de codelaag en uitvoerlaag.

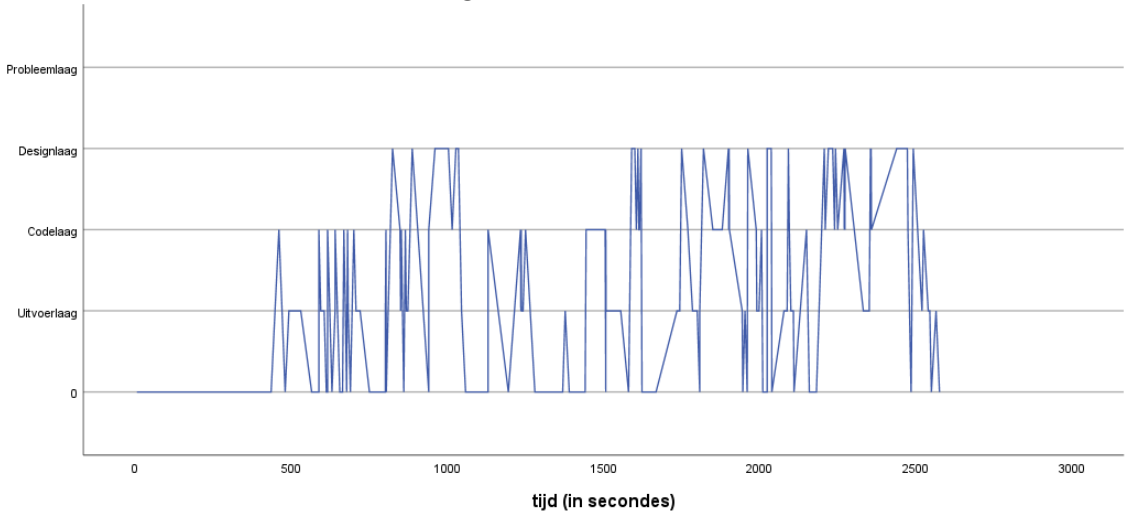
Het tweede paar wisselde nog sneller tussen de abstractielagen dan het eerste paar. Het is het enige paar waarbij de probleemlaag voorkwam, dat was twee maal het geval. Verder kwam de designlaag regelmatig voor. De kinderen wisselden dus voornamelijk tussen de designlaag, codelaag, uitvoerlaag en overig gedrag. Het paar kwam moeizaam op gang, maar rond de 500 secondes begonnen de kinderen steeds sneller te wisselen en lieten ze verschillende abstratielagen zien. Naar het einde toe, ongeveer vanaf 1700 secondes, wisselden ze minder vaak. Soms werd meerdere keren achter elkaar de uitvoerlaag geobserveerd, terwijl tussendoor niet de codelaag werd

Figuur 2: Paar 2



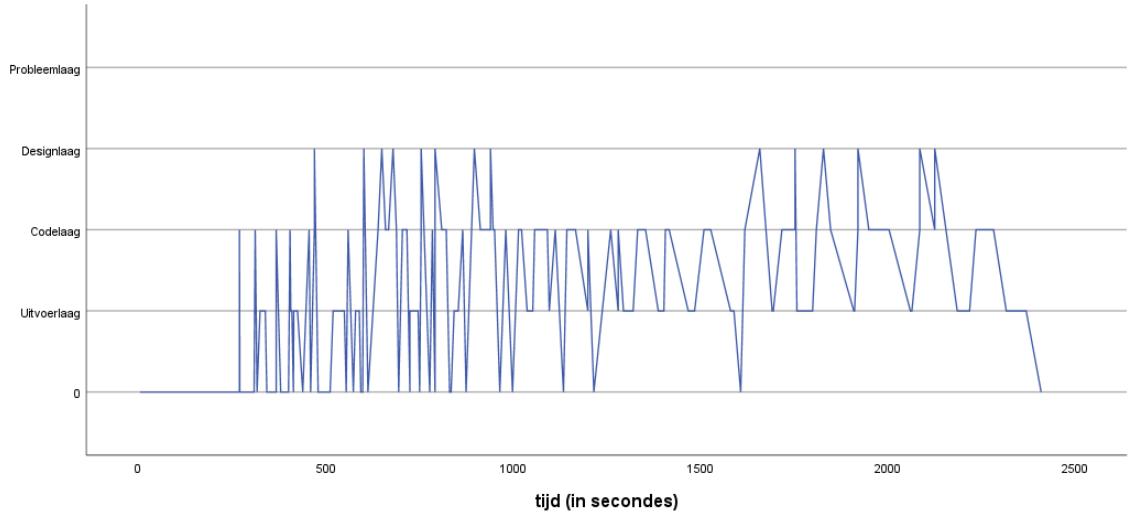
geobserveerd. De kinderen hadden de code dus meerdere keren uitgevoerd, zonder de code aan te passen. Dit gebeurde bij dit paar gedurende de gehele opdracht. Dat is bijvoorbeeld zichtbaar rond de 500 secondes, rond de 1200 secondes, vlak voor de 1500 decondes en enkele keren tegen het einde. De kinderen wisselden op die momenten tussen de uitvoerlaag en overige gedragingen, of bleven lang bij de uitvoerlaag, maar wisselden niet tussen uitvoerlaag en een andere abstractielaag.

Figuur 3: Paar 3



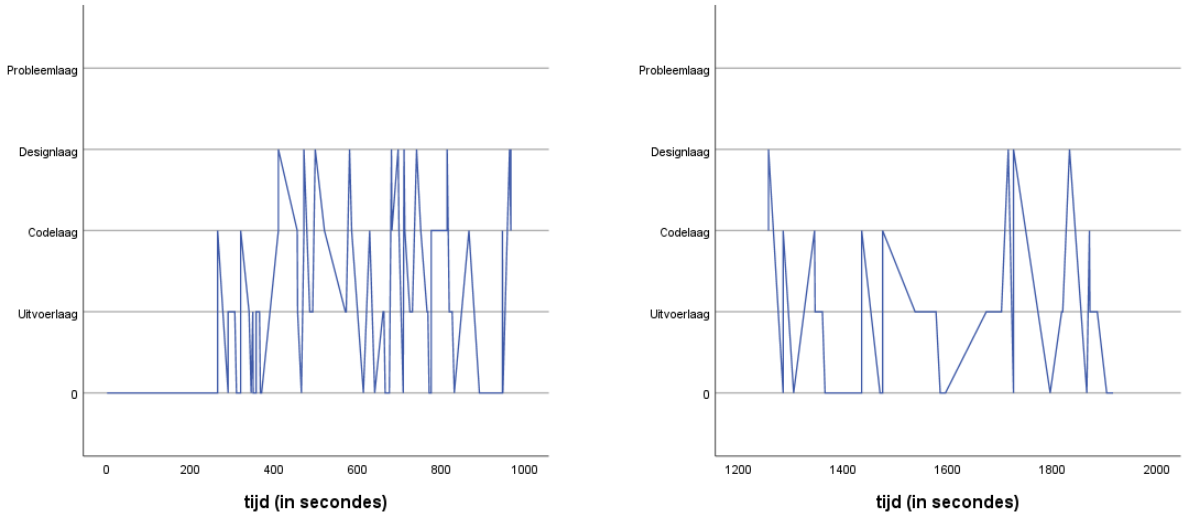
Bij het derde paar begon na ongeveer 500 secondes met het laten zien van de abstractielagen, wat iets later is dan bij de andere paren. Bij hen werd geen enkele keer de probleemlaag geobserveerd, maar wel regelmatig de designlaag. Ook naar het einde toe werd vaak de designlaag geobserveerd. De kinderen wisselden dus vooral tussen de designlaag, codelaag, uitvoerlaag en overig gedrag. Net als bij het eerste paar, maar anders dan het tweede paar, bleef de snelheid van het wisselen tussen de verschillende abstractielagen constant bij het derde paar.

Figuur 4: Paar 4



Het vierde paar liet naar het einde toe steeds minder vaak overige gedragingen zien en bleef steeds meer een abstractielaag laten zien. Daarbij wisselden de kinderen ook steeds minder snel tussen de abstractielagen. De designlaag kwam voornamelijk aan het begin en aan het einde voor, maar in het midden niet. Verder wisselden ze continu veel tussen de codelaag en de uitvoerlaag.

Figuur 5: Paar 5, deel 1 en 2



Bij het laatste paar zijn er twee grafieken, omdat de camera halverwege was uitgevallen en later weer is aangezet. De grafiek van figuur .. geeft de observaties weer die zijn gedaan voordat de camera uitviel en de grafiek in figuur .. geeft de observaties weer van daarna, waar de tijd waarin de observaties zijn gedaan daarom niet bij 0 secondes beginnen, maar rond 1200 secondes. De

probleemlaag kwam niet voor, maar de designlaag wel regelmatig. Voornamelijk in de eerste grafiek van dit paar komt dat naar voren.

In al deze grafieken valt op dat er veel en snel gewisseld werd tussen de verschillende abstractielagen, vooral tussen de codelaag en de uitvoerlaag. Verder kwam de uitvoerlaag soms meerdere keren achter elkaar voor, terwijl er niet tussendoor een observatie van bijvoorbeeld de codelaag werd gedaan. Het wisselen tussen de verschillende abstractielagen lijkt soms ook naar het einde van de opdracht toe iets minder snel te gaan, vergeleken met het begin.

Abstractielagen	**Paar 1**	**Paar 2**	**Paar 3**	**Paar 4**	**Paar 5**	**Totaal**	**Gemiddeld**
Probleemlaag	0	2	0	0	0	2	0,4
Designlaag	4	23	38	15	18	98	19,6
Codelaag	66	38	64	84	24	276	55,2
Uitvoerlaag	92	79	89	89	58	407	81,4
Totaal	162	142	191	188	100	783	156,6

Tabel 2: Absoluut aantal keren dat de verschillende lagen zijn geobserveerd.

Percentages	**Paar 1**	**Paar 2**	**Paar 3**	**Paar 4**	**Paar 5**	**Totaal**
Probleemlaag	0%	1%	0%	0%	0%	0%
Designlaag	2%	16%	20%	8%	18%	13%
Codelaag	41%	27%	34%	45%	24%	35%
Uitvoerlaag	57%	56%	47%	47%	58%	52%

Tabel 3: Relatief aantal keren dat de verschillende lagen zijn geobserveerd.

Bij het eerste paar valt op dat de designlaag veel minder voorkwam dan de codelaag en uitvoerlaag. De designlaag kwam ook veel minder voor bij dit paar dan bij de andere paren. De uitvoerlaag kwam bij de alle paren vaker voor dan de codelaag, maar het verschil tussen de codelaag en uitvoerlaag is bij het tweede en vijfde paar grootst. Daarnaast was het tweede paar het enige paar dat de probleemlaag liet zien. Bij het derde paar valt juist op dat de designlaag het vaakst voorkwam vergeleken bij de andere paren, maar het totale aantal observaties was ook het grootst, dus het is belangrijk om naar de percentages te kijken om een goede vergelijking te maken. De designlaag kwam inderdaad in een hoger percentage voor dan bij de andere paren, maar het verschil blijkt toch niet erg groot te zijn, met 20% bij het derde paar vergeleken bij 18% bij het vijfde paar. Bij het vierde paar is het verschil tussen de codelaag en de uitvoerlaag kleiner dan bij de andere paren. Ten slotte kwam de designlaag en de uitvoerlaag bij het vijfde paar vaker voor dan gemiddeld en de codelaag minder vaak.

Uit de grafieken van figuur 1 t/m 5 en tabellen 2 en 3 blijkt dat de probleemlaag nauwelijks voorkwam, soms kwam het zelfs helemaal niet voor. De designlaag werd ook niet erg vaak geobserveerd. Hierin zijn geen extreme afwijkingen te zien bij de verschillende paren. Een van de grotere afwijkingen is bij de designlaag, waar het eerste paar er minder van heeft dan de overige paren. Ook valt op dat

de uitvoerlaag het vaakst voorkomt, nog vaker dan de codelaag, wat vooral duidelijk te zien is in de tabellen.

4.2 Kwalitatieve data analyse

Na de kwantitatieve data analyse is er kwalitatief per paar gekeken wat er precies gebeurde wanneer de kinderen een bepaalde abstractielaag lieten zien. Door te kijken naar andere gedragingen die tegelijk voorkomen en transcripties te lezen is er per paar een algemeen beeld gevormd. Aan de hand daarvan kunnen de opvallendheden verder onderbouwd worden en voorzien van nadere interpretatie.

Het eerste paar bestaat uit twee slechtziende jongens van 11 en 12 jaar. De jongens praatten veel en lieten veel verschillende gedragingen zien. Daardoor kon er een duidelijk beeld gevormd worden van hoe deze jongens te werk gingen. Continu liet het ene kind een positieve ervaring blijken door te lachen, opgewonden te bewegen en zich positief uit te spreken, terwijl het andere kind continu afleiding liet zien of zich negatief uitsprak. Verder valt op dat ze allebei vooral visueel de code lazen en geen gebruik maakten van hulpmiddelen, zoals de screenreader.
Ten aanzien van de vier abstractielagen laten de observaties bij dit paar zich als volgt samenvatten. Ten eerste kwam de probleemlaag niet voor. Ten tweede kwam de designlaag slechts zelden voor. Dat viel al op bij de grafiek en tabel bij de kwantitatieve data analyse. Een van de kinderen was dus erg enthousiast, terwijl het andere kind regelmatig aangaf niet te begrijpen wat hij moest doen. Het meeste werk werd gedaan door de enthousiaste jongen, maar zowel de testleider als deze enthousiaste jongen probeerden de andere jongen aan te moedigen om ook stukjes code te typen, terwijl hij vooral steeds met lego aan het spelen was. Bij elk moment waarop de designlaag geobserveerd werd, gebeurde dat doordat de testleider en de enthousiaste jongen de andere jongen probeerden te betrekken bij de opdracht. Ten derde kwam de codelaag iets vaker dan gemiddeld voor. Meestal typte de enthousiaste jongen, maar het andere kind, werd ook af en toe aangemoedigd om code te typen. Wanneer hij dit daadwerkelijk deed, dicteerde de enthousiaste jongen wat hij moest typen. Hierbij kwam naar voren dat het kind dat moeite had met programmeren, ook last had van het extra obstakel dat de code niet in het Nederlands was. Hij schreef bijvoorbeeld “play” eerst met een “e” in plaats van met een “a”. Ook typte hij in de eerste instantie “sliep” in plaats van “sleep”. Zelfs wanneer gedicteerd werd wat hij moest intypen, ging het door de Engelstalige code nog steeds moeizaam. Ten vierde kwam de uitvoerlaag ook iets vaker dan gemiddeld voor. Wanneer het was gelukt om de jongen die meestal met lego aan het spelen was een stukje code te laten typen, valt op dat de enthousiaste jongen het vervolgens weer overnam om de code uit te voeren of aan te passen. Het was dus eigenlijk altijd de enthousiaste jongen die de code uitvoerde. Daarbij reageerde hij heel positief op de output, door bijvoorbeeld te dansen, in zijn handen te klappen of zich positief uit te spreken. Het andere kind reageerde soms ook positief op de output.

Het tweede paar bestond, net als het eerste paar, uit twee slechtziende jongens van 11 en 12 jaar. De beide jongen zeiden weinig tijdens de opdracht, ze waren duidelijk minder spraakzaam dan de jongens van paar 1. In plaats van alleen de transcripties, hebben voornamelijk coderingen bij andere gedragscategorieën geholpen een beeld te vormen van wat er bij dit paar gebeurde.
Ten aanzien van de vier abstractielagen laten de observaties bij dit paar zich als volgt samenvatten. Ten eerste viel bij dit paar op dat twee keer de probleemlaag voorkwam, terwijl de probleemlaag bij de andere paren geen enkele keer is geobserveerd. In beide gevallen werd het doel van de opdracht

besproken, na het uitvoeren van de code. Ten tweede bleek de designlaag vaak in combinatie met de codelaag voor te komen, waarbij de kinderen ook de aanwezigheid van samenwerking lieten zien. De kinderen waren dan aan het overleggen en vroegen elkaar advies bij het schrijven van de code. Ten derde werd de codelaag in het begin vooral geobserveerd nadat de testleider uitleg had gegeven of vragen van de kinderen had beantwoord. In de loop van de opdracht gingen de kinderen steeds meer onderling overleggen, in plaats van steeds de testleider om hulp te vragen. Daardoor kwamen de designlaag en de codelaag dus geregeld samen voor, met natuurlijk de gedragscategorie "samenwerking aanwezigheid". Soms waren de kinderen alleen code aan het schrijven, zonder dat ze praatten of tussendoor code uitvoerden. Tegen het einde van de opdracht leerden de kinderen instrumenten toe te voegen. De kinderen ervoeren dit, ondanks de instructie van de testleider, als moeilijk en verwarrend. Voor het toevoegen van instrumenten zijn de woorden "use" en "synth" nodig. De testleider moest dit spellen en de kinderen worstelden ermee. Verder werd er gebruik gemaakt van Word, waardoor de kinderen soms hulp van de testleider nodig hadden voor het kopiëren en plakken in Sonic Pi. Af en toe was er een kind afgeleid, verveeld of was de samenwerking afwezig doordat een van de kinderen niet bij de opdracht werd betrokken, maar over het algemeen werkten ze goed samen. Ten vierde kwam de uitvoerlaag vaak meerdere keren achter elkaar voor, zonder dat tussendoor de codelaag voorkwam. Daar waren verschillende redenen voor. De kinderen werkten met een koptelefoon, waardoor ze om de beurt moesten luisteren naar de output. Daarnaast luisterden de kinderen soms meerdere keren naar dezelfde output, simpelweg omdat ze het leuk vonden. Ze lieten dan een positieve ervaring zien, onder andere door te lachen, zich positief uit te spreken en opgewonden te bewegen.

Het derde paar bestaat uit twee slechtziende meisjes van 11 jaar oud. De testleider begeleidde de meisjes stap voor stap tijdens de opdracht. Meestal was dus de testleider aan het woord, de meisjes zelf zeiden niet erg veel, maar toch geven de verschillende gedragingen wel een beeld van wat er gebeurt.
Ten aanzien van de vier abstractielagen laten de observaties bij dit paar zich als volgt samenvatten. Ten eerste kwam de probleemlaag niet voor, zoals bij de meeste paren. Ten tweede kwam de designlaag wel regelmatig voor, zelfs vaker dan gemiddeld. Op die momenten gaf de testleider meestal uitleg of stelde een volgende stap voor. Daarnaast zien we dat de designlaag soms samenging met de codelaag. Verder lieten de meisjes regelmatig een negatieve ervaring zien, door hun schouders op te halen, ervaren moeilijkheid te laten zien of te fronsen. De testleider besprak ook vaak met de meisjes wat de code nou eigenlijk deed, waardoor de designlaag soms samenging met de uitvoerlaag, want ze voorspellen wat de uitvoer zal zijn, zonder de code al uit te voeren. Bijna altijd luisteren de kinderen naar instructies van de testleider. Ten derde ging de codelaag ook vaak samen met het luisteren naar instructie. De meisjes lieten zien dat ze worstelen met het materiaal. De testleider nam het zelfs af en toe over, ging op de juiste plek in de code staan of gaf letterlijk aan wat de kinderen moesten typen. Dan spelde de testleider bijvoorbeeld het woord "synth" dat nodig was om het geluid van een instrument te gebruiken. Toch (glim)lachen de meisjes zo nu en dan. Een enkele keer werd gebruik gemaakt van de zoomfunctie. Ten vierde kwam de uitvoerlaag in het begin vooral voor in combinatie met een negatieve ervaring. Dan bleek bij het uitvoeren van de code dat het geluid eerst niet te horen was, of het was heel zacht. De kinderen worstelden met het materiaal, stelden vragen en luisterden naar instructie. Later, wanneer het geluid het beter deed, begonnen ze ook te lachen en zich positief uit te spreken. Regelmatig keken de meisjes naar de testleider voor bevestiging, wanneer ze de code gingen runnen, bijvoorbeeld om te controleren of het klopt

dat ze Alt-R moesten gebruiken. Ook hier werd soms de zoomfunctie gebruikt. Het valt op dat bij de uitvoerlaag het meest voorkomt dat de meisjes daarbij lachen en zich positief uitspreken, vergeleken met de andere astractielagen. Ook voorspelden ze regelmatig wat er zou gebeuren zonder de code al uit te voeren en relateerden ze de uitkomst aan de voorspelde uitkomst. Dat kwam allebei voornamelijk doordat de testleider dan daarnaar vroeg. Dat voorspellen en relateren bleken ze moeilijk te vinden. Ze lazen de code dan (visueel) en worstelden ermee, waardoor de testleider het zelf moest uitleggen.

Het vierde paar bestaat uit een slechtziend meisje van 12 jaar, dat door de leerkracht is aangeduid als een "deels braille leerling" en een meisje van 11 jaar dat blind is. Deze kinderen hebben dus minder zicht dan de eerste drie paren. Net als de kinderen van het eerste paar, praatten de kinderen van dit paar veel en lieten veel gedragingen zien.
Ten aanzien van de vier abstractielagen laten de observaties bij dit paar zich als volgt samenvatten. Ten eerste kwam de probleemlaag niet voor. Ten tweede kwam bij dit paar de designlaag niet zo vaak voor als bij de meeste andere paren. Wanneer de designlaag voorkwam, ging dat vaak samen met het luisteren naar instructies van de testleider. Ook werkten ze goed samen. De meisjes lieten dat zien door onder andere elkaar te corrigeren en advies te vragen. Ten derde kwam de codelaag in verhouding tot de andere paren het vaakst voor. Bij het schrijven van de code gebruikten de meisjes een screenreader en Microsoft Word. Daarbij kregen de meisjes regelmatig uitleg van de testleider en reageerden de meisjes positief wanneer de screenreader goed werkte. Het commando "sleep" werd eerst getypt als "sliep". Ook bij de codelaag werkten de meisjes goed samen. Soms begeleidde het slechtziende meisje het blinde meisje door de vingers op de juiste plaatsen op het toetsenbord te leggen. Af en toe lieten de meisjes zien dat ze het moeilijk of verwarrend vonden en spraken zich negatief uit. Dat kwam naar voren wanneer zij instrumenten gingen toevoegen aan de code. Daarbij bleek "use" een lastig woord te zijn om te typen. Los van het typen van code zelf, was het besturen van de computer regelmatig een hele opgave voor de meisjes, op gegeven moment bracht de testleider bijvoorbeeld de cursor naar de goede regel, terwijl de meisjes zich negatief uitspraken door te schelden op de computer. Wanneer het wel goed ging lieten de meisjes een positieve ervaring zien door onder andere opgewonden te bewegen, in hun handen te klappen, zich positief uit te spreken, elkaar opgewonden aan te stoten of te lachen. Ten vierde kwam de uitvoerlaag in verhouding tot de andere paren niet zo vaak voor, net als bij het derde paar. Bij het vierde paar was het verschil tussen de codelaag en de uitvoerlaag duidelijk het kleinst. De uitvoerlaag kwam net als de andere abstractielagen regelmatig voor in combinatie met het luisteren naar instructie van de testleider. Voor het runnen van de code werd gebruik gemaakt van een screenreader en kreeg het blinde meisje vaak tactiel begeleiding van het slechtziende meisje. Ze lieten ook goede samenwerking zien, onder andere door elkaar te corrigeren, te overleggen, elkaar uitleg te geven en te begeleiden. Regelmatig reageerden ze positief op de output, onder andere door te lachen, zich positief uit te spreken, in hun handen te klappen, opgewonden te bewegen of elkaar opgewonden aan te stoten. Wanneer er iets niet klopte aan de output of de meisjes iets niet begrepen, overlegden ze daarover.

Het vijfde paar bestaat uit twee meisjes van 10 jaar oud, waarvan de ene slechtziend is en de ander blind. Het valt op dat deze meisjes meer gebruik maakten van hulpmiddelen dan de eerste drie paren, zoals de screenreader en de mogelijkheid om de code tactiel te lezen. Vooral het blinde meisje maakte vaak van deze mogelijkheid gebruik. Het slechtziende meisje maakte meestal alleen gebruik

van de screenreader. De meisjes keken en dachten met elkaar mee, waardoor de transcripties, naast de verschillende gecodeerde gedragingen, een redelijk volledig beeld gaven van wat er gebeurde.
Ten aanzien van de vier abstractielagen laten de observaties bij dit paar zich als volgt samenvatten. Ten eerste kwam de probleemlaag niet voor, zoals bij de meeste paren. Ten tweede kwam de designlaag iets vaker dan gemiddeld voor. Vaak ging het samen met het luisteren naar instructie en het overleggen met elkaar. Het valt op dat er niet erg vaak een positieve of negatieve ervaring naar voren kwam. In het begin lachten ze een enkele keer. Tegen het einde spraken de meisjes zich negatief uit, lieten afleiding zien of vroegen hoe lang het nog zou duren. Ten derde kwam de codelaag minder vaak dan gemiddeld voor. Hierbij werd steeds gebruik gemaakt van de screenreader, kregen de meisjes instructie van de testleider en overlegden de meisjes met elkaar. Naast het schrijven van code in het algemeen, bleek het navigeren door de code specifiek een hele opgave. Op gegeven moment wilde een van de meisjes graag doortypen, ook al adviseerde de testleider om de code uit te gaan voeren. Helemaal tegen het einde maakte testleider zelf een aanpassing in de code. Ten vierde kwam de uitvoerlaag iets vaker dan gemiddeld voor. Bij het runnen van de code werd vaak de screenreader gebruikt en zochten de meisjes ook bevestiging bij de testleider of bij elkaar, dat ze bijvoorbeeld inderdaad Alt-R moesten gebruiken. Aan het begin van de opdracht lieten ze ook bij het uitvoeren van de code geen positieve of negatieve reactie zien, maar later gingen ze glimlachen, meebewegen en meezingen. De testleider nam het runnen van de code een enkele keer over, tegen het einde van de opdracht.

5 Conclusie en discussie

Het doel van het onderzoek was om meer inzicht te krijgen in hoe kinderen met een visuele beperking leren programmeren. Daarvoor is Sonic Pi een interessant programma om te gebruiken, vanwege de auditieve output. Een manier om te beoordelen hoe kinderen met een visuele beperking werken en leren met het programma, is door te kijken naar de Computational Practices die de kinderen laten zien. Daaruit volgt de volgende onderzoeksvraag: Welke Computational Practices laten kinderen met een visuele beperking zien wanneer ze leren werken met Sonic Pi? Voor het vormen van een volledig beeld, is gekozen voor een mixed-method, waarbij er kwantitatief en kwalitatief onderzoek gecombineerd werd. Om de Computational Practices gestructureerd te kunnen observeren is gebruik gemaakt van het model van de vier abstractielagen [ea19].

5.1 Computational Practices in frequentie van abstractielagen

Alle kindparen lieten de verschillende abstractielagen zien. De frequentie waarmee de verschillende lagen voorkwamen in vergelijking met andere lagen kwam bij alle paren overeen, er waren geen grote uitzonderingen. Sommige lagen kwamen in het algemeen heel weinig voor.
De probleemlaag kwam zeer weinig voor, vaak zelfs geen enkele keer. De designlaag kwam ook veel minder voor dan de codelaag en de uitvoerlaag, maar vaker dan de probleemlaag. Er leken verschillende verklaringen te zijn voor het voorkomen van de designlaag bij de verschillende paren kinderen.
Bij het eerste paar leek de designlaag voor te komen, omdat de testleider en het kind dat de opdracht goed begreep het andere kind, dat er veel moeite mee had, bij de opdracht probeerden te betrekken. Het tweede paar werkte heel bewust samen, waardoor stappen vaker uitgesproken

werden. Het derde paar had juist veel moeite met de opdracht en daarom besprak de testleider veel met hen, waardoor de designlaag regelmatig voorkwam. Bij het vierde en vijfde paar ging het ook vaak samen met luisteren naar instructie van de testleider of samenwerking met elkaar.
De uitvoerlaag kwam het vaakst voor, soms zelfs meerdere keren achter elkaar, zonder een andere abstractielaag zoals de codelaag ertussen. De uitvoerlaag kwam bij elk paar het vaakst voor en bij elk paar ging het vaak gepaard met een positieve reactie. De kinderen wisselden snel tussen de verschillende abstractielagen.

Dat de probleemlaag bijna niet voorkwam zou kunnen liggen aan het materiaal Sonic Pi en aan de opdracht. De opdracht was namelijk heel vrij, waardoor er geen duidelijk eindpunt of doel te bespreken viel, waardoor de probleemlaag niet snel van toepassing was. De designlaag kwam om verschillende redenen voor, dus er lijken grote verschillen te zijn binnen de doelgroep in hoe de kinderen te werk gaan. Het "low-floor and high-ceiling"principe van Sonic Pi maakt dat Sonic Pi een goed bruikbaar programma lijkt voor deze doelgroep. Verder zou de samenwerkingsopzet positief kunnen zijn, waarbij kinderen die het snel oppakken de kinderen kunnen helpen die moeite hebben met het programmeren. De hoeveelheid begeleiding was flexibel, dat past ook goed bij een doelgroep die zo divers is. De uitvoerlaag kwam waarschijnlijk vaak voor doordat de kinderen de auditieve output positief ervaarden. Dit bevestigt de verwachting dat het maken van muziek de leerlingen aanspreekt.

Interessant is de vergelijking met het onderzoek van Faber [ea19], waarin opvallend genoeg ook de probleemlaag weinig voorkwam en er veel gewisseld werd tussen de verschillende abstractielagen. De leerlingen gingen vaak terug naar de code of design laag, nadat ze het resultaat van hun code bij de uitvoerlaag zagen. Wanneer er een onverwacht of ongewenst resultaat was, gingen leerlingen terug naar de codelaag om de code aan te passen en de code opnieuw te runnen, wat een proces van debugging aangeeft. In andere gevallen gingen leerlingen terug naar de designlaag, voor een nieuw design (redesign) van de oplossing. Het is moeilijk om vast te stellen of een kind zich bezighield met debugging of dat hij simpelweg aan het programmeren was met behulp van een trial en error tactiek. Door verbale uitingen van het kind was in meeste gevallen overtuigend vast te stellen dat trial en error een zelden gebruikte tactiek was. Deze overeenkomsten zijn zo opvallend, omdat er een ander materiaal gebruikt werd en de doelgroep was anders. De participanten hadden geen visuele beperking en de kinderen waren 5-6 jaar oud.
Deze resultaten komen dus overeen, ondanks het verschil in doelgroep en materiaal. Het is dus mogelijk dat het proces van abstractie bij kinderen onafhankelijk is van het materiaal dat zij gebruiken en onafhankelijk van hun zicht. Misschien heeft het materiaal Sonic Pi en de vrije vorm van de opdracht wel een versterkend effect gehad, maar komt de probleemlaag ook het minst voor bij andere materialen en wordt er ook bij andere materialen veel gewisseld tussen de verschillende abstractielagen. Of de leeftijd ook geen verschil maakt is lastig te zeggen, met slechts twee leeftijdsgroepen om te vergelijken. Misschien had het wel te maken met hun leeftijd en kwam het doordat beide groepen kinderen nog erg jong waren. De ene groep had een leeftijd van 5-6 jaar en de andere groep van 10-12 jaar. Misschien laten oudere kinderen of kinderen met wat meer ervaring vaker de probleemlaag zien. Dat er een proces van debugging en redesign plaatsvond is positief, het is een belangrijk onderdeel van Computational Thinking.

Er zou onderzoek gedaan kunnen worden naar andere materialen in het algemeen, maar ook specifiek

wanneer de probleemlaag en designlaag vaker voorkomen en of er bij de andere materialen ook zo snel wordt gewisseld tussen de verschillende abstractielagen. Bij de keuze voor verschillende materialen zou er ook kunnen worden gekeken naar verschillende leeftijdsgroepen. Daarnaast zou het interessant zijn om in verder onderzoek ook kinderen zonder visuele beperking mee te nemen, zodat er tussen kinderen met en zonder visuele beperking vergeleken kan worden. Er zou ook geëxperimenteerd kunnen worden met de opzet, door bijvoorbeeld de kinderen niet in paren te laten werken, maar alleen of in grotere groepen.

5.2 Algemeen gebruik Sonic Pi: invloed van Engelse taal

Bij vier van de vijf paren kwam duidelijk naar voren dat de Engelse code voor een extra barrière zorgde.

In eerder onderzoek [Aar16, Pet19] kwam dit niet naar voren, omdat de participanten in die onderzoeken Engelstalig waren. De kinderen zaten in het Britse systeem in jaar 9 of in het Nieuw-Zeelandse systeem in jaar 8. Dat betekent dat de kinderen een leeftijd hadden van ongeveer 13 en 12 jaar respectievelijk. De kinderen waren dus net iets ouder of even oud als de kinderen in dit onderzoek, maar de kinderen hadden het grote voordeel dat hun moedertaal Engels was. Veel kinderen bleken in deze onderzoeken te waarderen dat zij te maken kregen met tekstuele code. Mogelijk zou het voor kinderen met een visuele beperking juist een voordeel kunnen zijn als een programmeertaal op blokken is gebaseerd, in plaats van op tekst. Als de programmeertaal Engels is, helpt het mogelijk als de kinderen niet zelf de woorden goed hoeven te spellen, maar alleen hoeven te herkennen in de blokjes. Qua taal zou het dus voordelen kunnen hebben, maar qua toegankelijkheid geeft het problemen. De toegankelijkheid van programmeertalen gebaseerd op blokken is een van de meest onderzochte problemen bij het leren programmeren voor kinderen met een visuele beperking [ea18b, ea18a].

Er zou een vertaling gemaakt kunnen worden voor niet-Engelstalige kinderen. Het is niet gewenst dat kinderen die al een extra barrière ervaren bij het leren programmeren, door hun visuele beperking, ook nog een barrière ervaren doordat de programmeertaal in het Engels is. Het is onnodig. Verder zou het interessant kunnen zijn om te onderzoeken of de taal waarin een programmeertaal is geschreven een grotere barrière vormt bij kinderen met een visuele beperking dan bij kinderen zonder visuele beperking. Mogelijk is het een grotere barrière voor kinderen met een visuele beperking, omdat coderen vereist om de Engelse woorden te schrijven, in plaats van bijvoorbeeld alleen uit te spreken. Aan de andere kant zou voor elk kind (binnen deze leeftijdsgroep) kunnen gelden dat de Engelse taal een barrière vormt. In Nederland hebben kinderen op deze leeftijd nog maar kort Engelse les. Er zou ook onderzocht kunnen worden of kinderen met een visuele beperking een voorkeur hebben voor een tekstuele programmeertaal of een op blokken gebaseerde programmeertaal. Daarbij zou het belangrijk zijn om te kijken naar mogelijkheden om de toegankelijkheid van op blokken gebaseerde programmeertalen te verbeteren.

5.3 Ondersteuning van kinderen met een visuele beperking

Er waren een aantal opvallendheden aan hoe de kinderen werkten. Ten eerste maakten de paren met alleen slechtziende kinderen niet of nauwelijks gebruik van hulpmiddelen. Ten tweede gebruikten

alleen de paren met blinde kinderen de screenreader om de code auditief te kunnen lezen. Ten derde las alleen het blinde kind van het vijfde paar de code ook tactiel. Ten vierde kregen alle paren veel begeleiding van testleider. Ten vijfde verschilde het erg per paar of de kinderen veel praatten.

De kinderen bleken dus ondanks de mogelijkheid hulpmiddelen te kunnen gebruiken, toch vaak visueel de code te lezen. Mogelijk was Sonic Pi niet toegankelijk genoeg voor hulpmiddelen zoals de screenreader en was het te gecompliceerd om een aparte tekstverwerker te gebruiken naast Sonic Pi. Dat wil echter niet zeggen dat de kinderen die nu geen screenreader hebben gebruikt het wel zouden hebben gedaan als de toegankelijkheid beter zou zijn. Er zijn grote verschillen binnen de doelgroep, zoals eerder is genoemd. Er zijn grote verschillen in hun visus en dat geldt ook voor de voorkeuren van de kinderen qua hulpmiddelen. De screenreader lijkt in dit geval wel het vaakst te zijn gebruikt, voor zover er hulpmiddelen werden gebruikt. Volgens een literatuuroverzicht [ea18a] is het belangrijk dat een programmeeromgeving toegankelijk is voor hulpmiddelen zoals een screenreader. Verder is het bij deze doelgroep belangrijk om een passend thema te kiezen voor programmeeractiviteiten, om het toegankelijk en innemend te maken voor hen. De kinderen lieten een positieve ervaring blijken bij het luisteren naar de output, dus Sonic Pi lijkt wat dit betreft een zeer geschikt programma. Het combineren van programmeren met muziek maken lijkt de kinderen aan te spreken.

Er zou onderzoek gedaan kunnen worden naar verschillende manieren waarop een screenreader code zou kunnen voorlezen. Misschien zijn er ook andere hulpmiddelen mogelijk binnen het visueel lezen. Verder zou onderzoek naar andere materialen kunnen bevestigen dat de auditieve output bij Sonic Pi kinderen met een visuele beperking meer aanspreekt dan bijvoorbeeld een visuele output.
De kinderen in dit onderzoek hadden erg weinig ervaring met programmeren. Het zou ook interessant kunnen zijn om te kijken naar kinderen met al wat meer ervaring of kinderen gedurende een langere tijd te observeren wanneer zij programmeeronderwijs krijgen. De kinderen kregen drie keer een les van een uur, verspreid over enkele weken. Mogelijk zouden de ontwikkelingen gevolgd kunnen worden voor een langere periode. Mogelijk hebben de kinderen nadat ze wat meer ervaring hebben opgedaan minder begeleiding nodig en wordt het onderzoek minder beïnvloed door bijvoorbeeld verschillende begeleiders. Verder hebben kinderen met meer ervaring mogelijk ook een andere werkwijze en daardoor ook andere obstakels bij het leren programmeren.

5.4 Beperkingen

Er kunnen een aantal beperkingen aan dit onderzoek genoemd worden.
Het is opvallend dat bij het derde paar de gedraging *code runnen* soms ontbreekt. Dan werd alleen *output volgen* geobserveerd. Het lijkt duidelijk wat er gebeurde: het runnen van de code was als handeling niet zichtbaar. Het ene kind typte, vervolgens luisteren de kinderen samen naar de output. Runnen gebeurde snel en niet zichtbaar tussendoor. Het beïnvloedt/vertekent de data of het verdere overzicht niet erg. Wanneer dit gebeurde bij andere paren werd *code runnen* en *output volgen* vaak ook als één observatie van de uitvoerlaag gezien.
Verschillende mensen hebben de video's verwerkt naar Excelbestanden met transcripties en codes voor gedragingen. De manier van het coderen van gedragingen verschilt mogelijk licht daardoor. Om dat te beperken waren er protocollen en werden de bestanden individueel en in vergelijking tot elkaar doorgenomen, waarbij opvallendheden werden gecheckt in de video's en eventueel aangepast,

door steeds dezelfde onderzoeker.
Dit onderzoek zou herhaald kunnen worden met een grotere groep kinderen, om de bevindingen te bevestigen of mogelijk nog andere inzichten op te doen. Door de grote verschillen in het gedrag binnen deze doelgroep zouden meer kinderen voor meer inzichten en nieuwe oplossingen kunnen zorgen om het programmeeronderwijs voor kinderen met een visuele beperking uiteindelijk te kunnen verbeteren. Uit de literatuur bleek dat Sonic Pi ook geschikt is voor kinderen met meer programmeerervaring, dus dit materiaal zou daarvoor gebruikt kunnen worden. Daarbij zou het interessant zijn om ook kinderen zonder visuele beperking deel te laten nemen aan het onderzoek, zodat ook met deze groep vergeleken kan worden.
De bedoeling was dat bij het onderzoek thinking-out-loud gestimuleerd zou worden. Dat is mogelijk niet altijd goed gelukt, bijvoorbeeld het tweede paar praatte weinig. Het is niet duidelijk of dat komt doordat de kinderen simpelweg anders te werk gingen of doordat er een andere testleider bij was of doordat er een andere aanpak was van de testleider. Verder zou bij deze kleine groep kinderen de resultaten licht kunnen vertekenen, doordat sommige paren intensiever begeleiding nodig hadden dan andere paren, maar er zijn sowieso grote verschillen binnen de doelgroep.

5.5 Conclusie

De belangrijkste inzichten van dit onderzoek zijn als volgt. De kinderen wisselden snel tussen de verschillende abstractielagen en daarbij kwam vooral de probleemlaag erg weinig voor. Dat lijkt onafhankelijk te zijn van het materiaal en van hun visus. Mogelijk heeft het wel te maken met hun leeftijd. Dat de programmeertaal van Sonic Pi in het Engels is blijkt een onnodige extra barrière te vormen voor de meeste kinderen. Los daarvan lijkt Sonic Pi geschikt te zijn voor doelgroep. Het voldoet namelijk aan het "low-floor and high-ceiling" principe, wat wenselijk is bij een diverse doelgroep als deze. Verder lieten de kinderen regelmatig positieve ervaringen blijken als reactie op de auditieve output.
Sonic Pi lijkt een geschikt programma om kinderen te leren programmeren, ook wanneer zij een visuele beperking hebben. Zo kunnen jonge mensen ondanks hun beperking vaardigheden leren, net als hun leeftijdsgenoten zonder visuele beperking, met hetzelfde programma, in plaats van met een speciaal voor hen ontwikkeld programma. Er valt echter nog genoeg te onderzoeken om programmeeronderwijs te verbeteren.

Referenties

[Aar16] S. Aaron, A. Blackwell & P. Burnard. The development of sonic pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music, Technology and Education*, 9(1):75–94, 2016.

[BMR12] K. Brennan & M. Resnick. New frameworks for studying and assessing the development of computational thinking. *American Educational Research Association meeting, Vancouver, BC, Canada*, 2012.

[ea05] Als et al. Exploring verbalization and collaboration of constructive interaction with children. *INTERACT*, pages 443–456, 2005.

[ea17] Faber et al. Teaching computational thinking to primary school students via unplugged programming lessons. *Journal of the European Teacher Education Network*, 12:13–24, 2017.

[ea18a] Bennett et al. Making programming accessile to learners with visual impairments: A literature review. *International Journal of Computer Science Education in Schools*, 2(2), 2018.

[ea18b] Morrison et al. Torino: A tangible programming language inclusive of children with visual disabilities. *Human-Computer Interaction*, 2018.

[ea19] Faber et al. Observing abstraction in young children solving algorithmic tasks. *ISSEP 2019 Conference Proceedings*, 2019.

[Her] Felienne Hermans. https://www.felienne.com/archives/6004?

[Ken19] Stichting Kennisnet. Programmeren in het po. Technical report, Kennisnet, https://maken.wikiwijs.nl/74282, januari 2019.

[Lab] Programming Education Research Lab. https://perlliacs.wordpress.com/.

[lia] https://liacs.leidenuniv.nl.

[oz] https://www.universiteitleiden.nl/nieuws/2018/11/blinde-kinderen-leren-programmeren.

[Pet19] Christopher Grant Petrie. *Interdisciplinary computational thinking with music and programming: a case study on algorithmic music composition with Sonic Pi*. PhD thesis, University of Canterbury, 2019.

[Vis] Visio. https://www.visio.org/nl-nl/slechtziend-of-blind. geraadpleegd op 19-08-2020.