4
Juni
2018

Misconcepten over Programmeren
- op de Basisschool -

Emily C. A. Bakker

Eerste beoordelaar: Elian de Kleine | e.dekleine@saxion.nl|

Tweede beoordelaar: Jan Willem de Graaf | j.w.degraaf@saxion.nl|

Het Onderzoeken van Misconcepten over Programmeren bij Kinderen van 8-13 Jaar.

Afstudeeronderzoek.

Emily C. A. Bakker

4 juni 2018

Studentnummer: 403888
Toegepaste Psychologie
Academie Mens & Arbeid
Saxion Hogeschool te Deventer

Opdrachtgever:
F. J. J. Hermans
Assistant Professor at
Technische Universiteit Delft

“Juf, ik wordt later ook computervrouw, net als u”

“Dus u bent van de TU Delft? Dan bent u vast heel slim. Dan zie ik u later nog wel in het nieuws met al uw uitvindingen”

“Zeg juf, ik heb dat programmeren van u eens geprobeerd, en het was best oké”

“Juf, wat wilt u later eigenlijk worden?”
‘Hm, nou , dat weet ik eigenlijk nog niet’
“Nog steeds niet?! Wordt dat niet een keertje tijd?!”

“Juf, waarom bent u eigenlijk gaan programmeren?”
‘Nou... Uhm… Gewoon… Het is wel grappig’

Voorwoord

Eindelijk dan staan de laatste woorden op papier en ben ik bijna klaar met de afronding van de allerlaatste beproeving van mijn opleiding Toegepaste Psychologie, zo ongeveer zeven uur voor de deadline weliswaar. Als ik al vier jaar lang tot op het allerlaatste moment moet doorwerken aan opdrachten, waarom zou ik daar dan voor mijn allerlaatste opdracht een uitzondering voor maken? Er is slechts één verschil, ik geloof dat ik voor het eerst (relatief) op tijd begonnen ben. Wellicht een teken van volwassenheid en verstandigheid, maar dat moet zich nog bewijzen.

Van mijn opdrachtgever kreeg ik de kans om, binnen het onderwerp, mijn eigen onderzoek uit te voeren. Dat heeft me veel goeds gedaan, ik zat in ieder geval boordevol motivatie! De vele sparpartijen over ideeën en theorieën hebben daar ook veel bij geholpen, bedankt Marileen en Felienne.

Naast veel kennis, ben ik ook een titel rijker, namelijk ‘Juf’. Als ‘juf’ kreeg ik heel wat taaie momenten voor mijn kiezen, maar ook minstens net zo veel leuke dingen (zoals mijn naam mogen schrijven op de verjaardagskaarten van leerlingen die je hoogstens één keer hebt gezien). En natuurlijk de wijze, nuchtere en onbezorgde uitspraken van een aantal eigenwijze kinderen, die ik wel moest noteren.

De afgelopen vijf maanden heb ik overigens meer dan ooit gezien van Nederland, als kleine bijkomstigheid. Voor bijna dagelijkse reisjes tussen Hengelo en Delft, Rotterdam, Amsterdam, Hoevelaken en Deventer heb ik in totaal zo ongeveer 14.030 kilometer afgelegd. Ter vergelijking, gemiddeld wordt er per auto 13.200 kilometer per jaar afgelegd. Maar dankzij mijn studentenreisproduct, hebben deze kilometers mij gelukkig enkel tijd gekost.

Van clichés en standaard houd ik niet zo, maar tot slot wil ik toch graag een aantal mensen te bedanken. Beste Elian, bedankt voor alle leuke en leerzame momenten, zowel als begeleider als collega! Bedankt voor alle kritische oprechte feedback en adviezen. En natuurlijk alle anderen bedankt voor de enorme hoeveelheid tips en feedback, waardoor ik met trots mijn scriptie mocht inleveren.

Emily Bakker
4 juni 2018

Samenvatting

Digitalisering en automatisering hebben grote veranderingen teweeggebracht in ons dagelijkse leven en de arbeidsmarkt en spelen een cruciale rol in ons dagelijkse leven. Zo ook in het onderwijs. De vraag naar les in programmeren en andere digitale geletterdheid stijgt, ook in het primaire onderwijs. Er zijn echter nog geen richtlijnen voor programmeeronderwijs en wetenschappelijk onderzoek naar programmeeronderwijs voor kinderen staat nog in zijn 'kinderschoenen'.

Programmeren voor het primaire onderwijs en het onderzoek ernaar worden onder andere vanuit het bedrijfsleven aangemoedigd. Enkele bedrijven en organisaties zijn er actief mee, zo ook de TU Delft. Vanuit de TU Delft is lesmateriaal ontwikkeld en is er een onderzoekslijn opgezet naar programmeermisconcepten bij kinderen. Het huidige onderzoek is daar onderdeel van en had als doelen het opleveren van wetenschappelijk onderbouwde kennis, het ontwikkelen van een meetinstrument voor misconcepten en relevante mogelijkheden voor vervolgonderzoek ontdekken.

Er is explorerend onderzoek gedaan naar programmeermisconcepten bij kinderen uit groepen 5 tot en met 8. De onderzoeksgroep, met 369 kinderen, is opgedeeld in een experimentele en controlegroep. Beide volgenden vier programmeerlessen in programmeertaal Scratch, de experimentele groep kreeg daarbij extra uitleg gericht op het voorkomen/verhelpen van misconcepten omtrent variabelen en waarden. Aan het einde van de vier lessen maakten de kinderen een vragenlijst, waarmee misconcepten gemeten werden. De gemeten misconcepten zijn gebaseerd op beschreven misconcepten uit de literatuur en op observaties van de programmeerlessen.

Het experiment, dat betrekking had op de rol van didactiek in het ontwikkelen van een misconcept, had geen effect op het aantal misconcepten dat ontwikkeld werd. Daarnaast is er gekeken naar de rol van enkele persoonlijke factoren in het ontwikkelproces van een misconcept. De leeftijd en attitude van het kind hadden geen invloed op het aantal ontwikkelde misconcepten. De eerdere programmeerervaring van kinderen wel, kinderen met meer programmeerervaring toonden minder misconcepten.

Het onderzoek leverde geen harde bewijzen op, maar wel voldoende resultaten voor vervolgonderzoek en mogelijkheden voor de praktijk. Er is een model ontwikkeld waarin de verschillende aspecten, en hun samenhang, van het ontwikkelproces van een (mis)concept weergegeven zijn. Deze kan als uitgangspunt voor vervolgonderzoek fungeren. Er kan onderzoek gedaan worden naar uit welke onderdelen ieder aspect bestaat en hoe deze invloed hebben op het ontwikkelproces. Daarnaast is er op basis van Piagets theorieën en Neo-Piagetian theorieën een voorstel geschreven voor een leerlijn in programmeeronderwijs. Tot slot is er een meetinstrument voor programmeermisconcepten opgeleverd die bij vervolgonderzoek ingezet kan worden.

Inhoudsopgave

Voorwoord III

Samenvatting IV

Verklarende woordenlijst VII

1 De inleiding 1
1.1 Aanleiding 1
1.2 Onderzoeksvraag 2
1.3 Doelstelling 3

2. Theoretisch kader 4
2.1 Misconcepten 4
2.1.1. De Rol van Misconcepten 4
2.1.2 Ontwikkeling 5
2.1.3 Voorkomen en Verhelpen 6
2.1.4. Misconcepten Meten 8
2.2 Programmeeronderwijs 9
2.2.1 Constructivisme en Objectivisme 9
2.2.2 Programmeer(mis)concepten 10
Drempelconcepten 10
2.3 Leren 11
2.3.1 Fasen 11
2.3.2 Fasen en Programmeren 11
2.3.3 Attitude en Leren 12
2.4 Conceptueel Model 12

3. Onderzoeksdesign 13
3.1 Onderzoeksmethode 13
3.2 Onderzoeksdoelgroep 13
3.3 Onderzoeksinstrument 14
De Vragenlijst 14
3.4 Procedure 15
3.5 Analyses 16

4. Onderzoeksresultaten 17
4.1 Uitvoering Onderzoek 17
4.1.1 Verloop 17
Lessen 17
Vragenlijstafname 18
4.1.2 Respons 18
4.2 Resultaten 19
4.2.1 Homogeniteit van de Vragenlijstschalen 19
4.2.2 Misconcepten en het Experiment 19
Deelvraag 1 20
Deelvraag 2 20
4.2.3 Misconcepten en Leeftijd 20
4.2.4 Misconcepten en Attitude 21
4.2.5 Misconcepten en Ervaring 21

5. Conclusie, Discussie en Aanbevelingen 23
5.1 Conclusies 23
5.1.1 Misconcepten en Didactiek 23

5.1.2 Misconcepten en Cognitieve Fasen 23
5.1.3 Misconcepten en Attitude 24
5.1.4 Misconcepten en Ervaring 24
5.2 Discussie 25
5.2.1 Betrouwbaarheid en Validiteit 25
5.2.2 Bruikbaarheid 26
5.3 Aanbevelingen 26
5.3.1 Vervolgonderzoek 27
5.3.2 Programmeeronderwijs 29
5.3.3 De Toegepast Psycholoog 29
Literatuur ..... A
Bijlage 1 Eigen werkverklaring ..... E
Bijlage 2 Lijst misconcepten Sorva (2012) ..... F
Bijlage 3 Toestemmingformulier ..... K
Bijlage 4 Les 1 L
Bijlage 5 Experiment Les 1 ..... Z
Bijlage 6 Les 2 ..... BB
Bijlage 7 Experiment Les 2 ..... MM
Bijlage 8 Les 3 ..... OO
Bijlage 9 Experiment Les 3 ..... BBB
Bijlage 10 Les 4 CCC
Bijlage 11 Experiment 4 ..... NNN
Bijlage 12 Procedure Vragenlijstafname ..... OOO
Bijlage 13 Vragenlijst ..... QQQ
Bijlage 14 Planning Lessen ..... HHHH
Bijlage 15 Plan van Aanpak IIII

Verklarende woordenlijst

In deze sectie is een overzicht van enkele woorden met toelichting. In de tekst zijn deze gemarkeerd met een asterisk.

Automatisering	Het gebruik van informatietechnologie zoals computers en machines (via www.enncyclo.nl).
Connectis	Een ICT-organisatie uit Rotterdam die actief is in ontwikkelingen rondom veilige online identificatie (via www.connectis.nl). Zij financieren het project van de TU Delft waarbij studenten programmeerlessen geven op Rotterdamse basisscholen.
Constructivisme	Methode voor onderwijs waarbij de nadruk ligt op de theorie dat leerlingen hun eigen werkelijkheid construeren (via www.encyclo.nl).
Digitale geletterdheid	Alle vaardigheden die nodig zijn om zich in de digitale samenleving staande te houden en zichzelf te ontwikkelen (via www.mediawijsheid.nl).
Digitalisering	Het complete traject voor digitale ontsluiting, inclusief het toepassen van ICT (via www.encyclo.nl).
Lijsten	Een functie van programmeertalen om data op te slaan (via www.codingkids.nl).
Objectivisme	De theorie dat waarheden onafhankelijk van het menselijk verstand bestaan (via www.encyclo.nl).
Scratch	Een programmeertaal ontwikkeld voor kinderen. Deze werkt door middel van ‘blokken’ met functies, in tegenstelling tot tekst. Met scratch kunnen onder andere spellen en animaties geprogrammeerd worden (via scratch.mit.edu).
SIDN	Stichting Internet Domeinregistratie Nederland. Domeinnaamregistratie in Nederland (via www.sidn.nl). In samenwerking met Connectis bij de financiering van het project.
Spirograaf	Een instrument om geometrische patronen mee te tekenen (via www.encyclo.nl).
Werkbalkfavorieten	Een balk met knoppen in de internetbrowser, waarin specifieke webadressen zijn opgeslagen.
Wisbordje	Klein white-bord voor leerlingen. Wanneer een klassikale vraag gesteld wordt, moet iedere leerlijn een antwoord opschrijven, in plaats van dat een leerling aangewezen wordt om zijn/haar antwoord te noemen.

1 De inleiding

De aanleiding van het onderzoek wordt gevormd door de huidige (probleem)situatie te omschrijven aan de hand van praktijksituaties en literatuur. Op basis daarvan worden de onderzoeksvragen opgesteld. Tot slot wordt de doelstelling van dit onderzoek voor de opdrachtgever beschreven.

1.1 Aanleiding

De vraag naar programmeren in het onderwijs stijgt aanzienlijk, echter blijkt de implementatie van programmeeronderwijs moeizaam te verlopen (Kennisnet, 2016; FutureNL, z.j.). We weten dan ook nog weinig over de beste aanpak van programmeeronderwijs (Jeuring, Corbalan, Van Es & Leeuwestein, 2016).

Door digitalisering* en automatisering* spelen computers een steeds belangrijkere rol in onze maatschappij, wat een grote verandering teweegbrengt in de arbeidsmarkt en ons dagelijks leven (Wing, 2008). Onder andere door deze ontwikkelingen op de arbeidsmarkt stijgt de vraag naar programmeren en digitale geletterdheid* in het onderwijs (Mediawijsheid, 2017; Kennisnet, 2016). Vaardigheden in programmeren en digitale geletterdheid kennen overigens niet alleen voordelen op de arbeidsmarkt, maar zijn ook belangrijk om goed te kunnen functioneren in de vernieuwende maatschappij (Kennisnet, 2017; SLO, februari 2016; Mediawijsheid, 2017; De Ingenieur, 2014; FutureNL, z.j.).

Programmeren maakt ook zijn intrede in het primaire onderwijs, vaak als onderdeel van digitale geletterdheid. De keuze voor het aanbieden van programmeeronderwijs ligt nu nog bij de school zelf. Momenteel is daarvan sprake bij ongeveer een derde van alle basisscholen in Nederland (Kennisnet, 2016). Programmeren kan dan ook een mooie aanvulling vormen op het primaire onderwijs, doordat het bijdraagt aan de ontwikkeling van de leerling. Programmeren ondersteunt namelijk, onder andere, de ontwikkeling van het probleemoplossend vermogen en doet een beroep op taal, doordat leerlingen ideeën moeten kunnen verwoorden aan de computer (Saeli, Perrenet, Jochems & Zwaneveld, 2011; Szlávi & Zsakó, 2006; Hromkovic, 2006; Papert, 1980). Door programmeren worden tevens de reken- en computervaardigheden gestimuleerd (Resnick et al., 2009). Bovendien wordt er een beroep gedaan op de sociale vaardigheden. Door het effect van hun handelen te evalueren, leren leerlingen namelijk reflecteren op eigen handelswijze (Clements & Gullo, 1989). De vaardigheden die leerlingen leren, zijn relevant en toe te passen op meerdere vakgebieden. Tot slot stelt Papert (1980) dat leren programmeren net is als het leren van een nieuwe taal en dat juist kinderen hier goed in zijn.

De Technische Universiteit Delft levert een bijdrage aan programmeren in het primaire onderwijs. Met de visie zo veel mogelijk kinderen te leren programmeren, ontwikkelt Felienne Hermans, assistent professor, lesmateriaal in programmeertaal Scratch*. Daarnaast is in samenwerking met Connectis* en SIDN* een project opgestart waarbij basisscholen in Rotterdam zich gratis aan kunnen melden voor een set van vier programmeerlessen. Dankzij deze lessen maken leerlingen kennis met programmeren, ontwikkelen zij de beginselen van de vaardigheid, worden zij gemotiveerd deze verder te ontwikkelen en leren scholen laagdrempelig kennis maken met programmeeronderwijs (Connectis, z.j.). Dusver hebben ongeveer 40 scholen van dit aanbod gebruik gemaakt en hebben

ongeveer 900 kinderen de lessen gevolgd (Connectis, 2017b). De TU Delft verzorgt het lesmateriaal en werft studenten die de lessen geven. Connectis en SIDN verzorgen uitbetaling van de studenten en stellen laptops beschikbaar waarop de lessen uitgevoerd kunnen worden. Dat bedrijven investeren in programmeeronderwijs is overigens niet uitzonderlijk, veel bedrijven zijn hiertoe bereid omdat ze zo de interesse in programmeren te vergroten. Hiermee stimuleren ze leerlingen te kiezen voor een programmeeropleiding, waardoor een groter aantal leerlingen later voor een programmeeropleiding zal kiezen (Visser, 2017; NRC, 2017; NOS, 2014).

Daarnaast wordt op de TU Delft onderzoek uitgevoerd om wetenschappelijk onderbouwd inzicht te verkrijgen in hoe programmeren geleerd wordt, om de kwaliteit van programmeeronderwijs te kunnen verbeteren. Twee van deze onderzoeken liggen ten grondslag aan dit onderzoek. Een van de onderzoeken is het eerste onderzoek naar misconcepten over programmeren bij kinderen (Swidan & Hermans, nog niet gepubliceerd). De onderzochte misconcepten zijn overgenomen uit een verzameling van onderzochte misconcepten bij studenten en volwassenen opgesteld door Sorva (2012). Uit het onderzoek bleek dat veel van deze misconcepten ook kinderen veel voorkomen. De twee misconcepten ‘Input heeft effect op de uitvoering’ en ‘Een variabele kan meerdere waarden hebben’ kwamen het meeste voor. Uit het andere onderzoek is naar voren gekomen dat het gebruik van metaforen invloed heeft op het al dan niet ontwikkelen van een misconcept. Zo zou de metafoor 'Een variabele is een soort doosje waar je een getal in kunt stoppen' het misconcept 'Een variabele heeft meerdere waarden' versterken. In het huidige onderzoek zal een meer systematische aanpak gebruikt worden om misconcepten over programmeren bij kinderen te onderzoeken. De onderzoeksgroep zal opgedeeld worden in een experimentele en een controlegroep. Beide groepen krijgen vier programmeerlessen, maar de experimentele groep krijgt daarbij aanvullende uitleg gericht op het voorkomen/verhelpen van het misconcept ‘Een variabele heeft meerdere waarden’.

1.2 Onderzoeksvraag

De onderzoeksvraag is als volgt:

“Welke misconcepten over programmeren komen voor bij kinderen uit groepen 5-8?”

Omdat dit onderzoek meer explorerend dan beschrijvend is, is de vraagstelling heel open. De focus ligt minder op het vinden van harde bewijzen of het toetsen van theorieën. Het doel is het verkennen van de wetenschap met betrekking tot kinderen, programmeren en misconcepten. De focus ligt op het vinden van aanleidingen voor vervolgonderzoek en theorie toepassen in het programmeeronderwijs. Deze vraagstelling wordt ondersteund door de volgende deelvragen:

1. *“Wat zijn de verschillen in het aantal en soorten misconcepten tussen de experimentele en controlegroep?”*
2. *“Wat zijn de verschillen in het aantal en soorten misconcepten met betrekking tot het concept ‘variabelen en waarden’ tussen de experimentele en controlegroep?”*
3. *“Wat zijn de verschillen in misconcepten tussen het jonge en het oude kind?”*

4. *"Wat zijn de verschillen in misconcepten met betrekking tot het concept 'variabelen en waarden' tussen het jonge en het oude kind?"*
5. *"Is er een samenhang tussen attitude met betrekking tot programmeren en het aantal misconcepten?"*
6. *"Is er een samenhang tussen eerdere programmeerervaring en het aantal misconcepten?"*.

Er worden van vijf verschillende concepten mogelijke misconcepten onderzocht. Per concept zijn er meerdere misconcepten opgesteld. In deelvragen 2 en 4 wordt het concept 'variabelen en waarden' uitgelicht, hierop is namelijk de extra uitleg gericht omdat dit een van de meest voorkomende misconcepten is. Onderscheid tussen het jonge en het oude kind in deelvragen 3 en 4 wordt gemaakt op basis van de leerfasen uit de klassieke theorie van Piaget. Deelvraag 6 hangt samen met de neo-Piagetian theorieën [in dit onderzoek vrij vertaald naar 'nieuwe theorieën van Piaget'].

1.3 Doelstelling

De Technische Universiteit Delft is als onderwijsinstelling erg actief in het veld van wetenschappelijk onderzoek op uiteenlopende vakgebieden binnen het technische kader. Met betrekking tot programmeren in het primair onderwijs, zijn zij actief in het ontwikkelen van lessen in programmeertaal Scratch, voor zowel kinderen als docenten. Daarnaast zijn zij actief met het uitvoeren van wetenschappelijk onderzoek naar leren programmeren, om de kwaliteit van programmeeronderwijs te kunnen verbeteren.

Dit onderzoek heeft als doel het opleveren van wetenschappelijk onderbouwde kennis over misconcepten bij programmeren, kennis over didactiek in het programmeeronderwijs en een methode voor het meten misconcepten. De TU Delft kan de wetenschappelijke onderbouwde kennis gebruiken voor het ontwerpen van nieuwe onderzoeken. Kennis over didactiek kan gebruikt worden om de kwaliteit van lesmateriaal voor programmeeronderwijs te verbeteren. Tot slot kan de methode voor het meten van misconcepten gebruikt worden voor meetinstrumenten bij nieuwe onderzoeken en om leerkrachten inzicht te geven in de programmeervaardigheden van hun leerlingen.

2. Theoretisch kader

Het theoretische kader bevat wetenschappelijk onderbouwde informatie die ondersteunt bij het verdere ontwikkelingen van het onderzoek en uiteindelijke resultaten meer context kan bieden. Het hoofdstuk is in vijf paragrafen opgedeeld. Deze paragrafen komen overeen met de verschillende onderwerpen van de onderzoeksvragen.

2.1 Misconcepten

Een misconcept is een aanhoudende incorrecte verklaring voor een concept en komt in het algemeen veel voor (Sorva, 2012, Ma, 2007; Leonard, Kalinowski & Andrews, 2014). Het gevaar van een misconcept is dat het fouten in de uitvoering en nieuwe misconcepten kan veroorzaken (Sorva, 2012; Ma, 2007; Holland, Griffiths & Woodman, 1997). Een misconcept over programmeren kan resulteren in foute code, moeite met het oplossen van programmeerfouten en moeilijkheden met het leren van nieuwe programmeerconcepten. Een fout door misconcepten wordt niet toevallig gemaakt, een misconcept wordt namelijk consistent toegepast en dus worden deze fouten consistent gemaakt. Doordat men zich niet bewust is van het misconcept, zijn deze fouten moeilijk te herkennen.

Misconcepten worden ontwikkeld tijdens het opdoen van kennis (Sorva, 2012, Ma, 2007; Leonard, Kalinowski & Andrews, 2014). Een persoon kan echter ook al (foute) opvattingen hebben over een concept *voordat* hiervan kennis opgedaan wordt, dit wordt ook wel een preconcept genoemd (Leonard, Kalinowski & Andrews, 2014). Preconcepten ontstaan doordat voorkennis wordt gebruikt om situaties te verklaren. Een preconcept kan een misconcept worden, maar daar is niet altijd sprake van. Doorgaans worden beide begrippen samengenomen onder de noemer misconcept, waarvan ook in dit onderzoek sprake is.

Concepten kunnen grootschalig in twee categorieën ingedeeld worden, natuurlijke en artificiële concepten. Natuurlijke concepten zijn gebaseerd op eigen ervaringen (Zimbardo, Johnson & McCann, 2013). Artificiële concepten zijn gebaseerd op een combinatie van regels, zoals bijvoorbeeld een definitie van een woord. De misconcepten in dit onderzoek zijn gebaseerd op artificiële concepten, omdat er bij de programmeerlessen sprake is van aangeleerde kennis.

2.1.1. De Rol van Misconcepten

Naast dat misconcepten fouten in uitvoering veroorzaken, kunnen zij het leerproces van nieuwe concepten en lesstof ernstig verstoren (Sorva, 2012, Ma, 2007; Leonard, Kalinowski & Andrews, 2014). Dit komt doordat misconcepten deel uitmaken van schema's. Een schema is een cluster van verwante concepten die een algemeen conceptueel kader vormen (Zimbardo, Johnson & McCann, 2013; Sorva, 2012; Anderson, 2010). Schema's worden gemaakt en aangepast op basis van (leer)ervaringen. Door nieuwe informatie aan een schema te verbinden, krijgt het context. Wanneer een misconcept deel uitmaakt van een schema, heeft dit invloed op de wijze waarop nieuwe informatie waargenomen wordt. De nieuwe informatie wordt aan foute voorkennis gekoppeld, waardoor er een foute context bij de nieuwe informatie wordt gevormd. De ontwikkeling van valide schema's, een schema zonder misconcepten, is cruciaal om efficiënt te leren programmeren (Ma, 2007). Wanneer een leerling bijvoorbeeld geen misconcepten ontwikkelt zullen er waarschijnlijk minder fouten gemaakt worden tijdens het programmeren, waardoor er minder tijd nodig is om deze

fouten op te lossen. Daarnaast zouden deze fouten sneller geïdentificeerd zijn en is de succeservaring groter, wat een positief effect heeft op de motivatie van de leerling (Gerrichhauzen, 2010).

In de literatuur worden misconcepten veelal in samenhang met mentale modellen beschreven. Mentale modellen zijn, net als schema's, cognitieve structuren die concepten aan elkaar linken. De literatuur in dit onderzoek definieert mentale modellen tevens op deze wijze. Om verwarring te voorkomen, wordt in dit onderzoek mentale modellen gelijkgesteld aan schema's.

2.1.2 Ontwikkeling

Een misconcept kan op tal van manieren ontwikkeld worden en vindt altijd plaats tijdens het aanleren van kennis (Ma, 2007; Holland, Griffiths & Woodman, 1997). Er wordt dan ergens in dit proces een fout gemaakt. De fout kan zich aan twee zijden voordoen; bij het verstrekken van de kennis of bij het verwerken van de kennis.

Bij het verstrekken van de kennis kan onderscheid tussen twee aspecten worden gemaakt; de correctheid van de kennis en de didactiek. Wanneer kennis inhoudelijk niet correct en het concept wel ontwikkeld wordt, is er sprake van een misconcept. Anderzijds kan de oorzaak bij de didactiek liggen waarmee de kennis overgebracht wordt. Wanneer nieuwe kennis zelfstandig tot zich genomen wordt, is er meer ruimte voor interpretatie (Hollingsworth & Ybarra, 2017). Deze interpretatie vergroot de kans dat kennis verkeerd wordt geïnterpreteerd en verwerkt, wat misconcepten veroorzaakt. Daarnaast moet de vorm van de aangeboden informatie aansluiten op de belevingswereld van de leerling, anders is deze moeilijk te koppelen aan bestaande schema's. De kans dat er onterecht verbanden tussen schema's worden gelegd of dat de nieuwe informatie vervormd wordt, is dan aanzienlijk groter (Türkmen & Usta, 2007; Schoenmaker, Stomp & Valstar, 2002; Brown & Clement, 1989; Stavy, 1991). Doordat programmeerconcepten veelal abstract zijn en kinderen moeite hebben met het begrip van abstracte concepten, zorgt dit bij programmeren voor veel problemen (Schoenmaker, Stomp & Valstar, 2002).

Bij het verwerken van de informatie kan wederom onderscheid tussen twee aspecten worden gemaakt; het waarnemen van de informatie en het verwerken van de informatie. Het eerste aspect is eenvoudig; wanneer informatie door de leerling niet volledig of onjuist wordt waargenomen, bijvoorbeeld door leesfouten of onoplettendheid, kan er geen valide schema ontwikkeld worden. Anderzijds ligt de oorzaak in het verwerken van de informatie in de hersenen, bij de ontwikkeling van de cognitieve structuur van schema's. De ontwikkeling van schema's is in drie stadia de omschrijven; het pre-theoretische stadium, het ervaringsgerichte stadium en het deskundige stadium (Sorva, 2012).

In het pre-theoretische stadium vinden de eerste ontwikkelingen van het schema plaats op basis van specifieke waarnemingen. Voor programmeren betreft dit bijvoorbeeld het waarnemen van de output na het uitproberen van een commando. Deze ontwikkelingen komen tot stand door het koppelen van de waarnemingen aan kennis/ervaringen uit soortgelijke schema's. Er worden dus overeenkomsten met reeds aanwezige kennis gezocht om verbanden te kunnen leggen. Wanneer er op dat moment nog maar weinig soortgelijke schema's zijn, is de potentie voor het gebruik van het nieuwe schema laag. Het aanmaken van een nieuw schema vergt namelijk een actieve leerhouding en motivatie (Sorva, 2012; Anderson, 2010).

In het ervaringsgerichte stadium wordt er meer ervaring opgedaan met de nieuwe kennis, bijvoorbeeld door het maken van oefenopgaven, waarin de stof op verschillende wijzen wordt toegepast. De verbanden tussen schema's worden hierdoor gelegd en versterkt. De nieuwe kennis kan hierdoor al naar soortgelijke schema's overgedragen worden. Dit stadium duurt vaak het langst om af te ronden, omdat het oefenen met de stof en het eigen maken van de stof een langdurig proces is (Hollingsworth & Ybarra, 2017).

In het deskundige stadium kan nieuwe informatie eenvoudig aan het schema gekoppeld worden. Nieuwe concepten kunnen dan gemakkelijk bij een schema ingedeeld worden. Er kunnen in dit stadium ook verbanden worden gelegd met heel verschillende schema's.

De kans op het ontwikkelen van misconcepten is het grootst in het eerste stadium. Voorkennis speelt daar namelijk een grote rol bij het opslaan van de informatie. Wanneer deze informatie niet klopt, wordt de nieuwe informatie in een foute context geplaatst en ontwikkelt het zich tot een misconcept. Dit kan tevens gebeuren door het onterecht koppelen van de nieuwe informatie aan de bestaande informatie, dit gebeurt voornamelijk wanneer de informatie onvolledig is waardoor er ruimte is voor eigen interpretatie (Hollingsworth & Ybarra, 2017). Hiervan is bijvoorbeeld sprake wanneer twee concepten aan elkaar gekoppeld worden, die doorgaans niks met elkaar te maken hebben. Daarnaast is, zoals beschreven, de potentie voor de ontwikkeling van het eerste stadium laag wanneer weinig soortgelijke schema's aanwezig zijn. Om de ontwikkeling van het schema door te zetten, is een actieve houding en motivatie tot leren nodig. Wanneer deze ontbreekt, is het voor een leerling makkelijker om de nieuwe informatie aan te passen om deze op bestaande schema's aan te laten sluiten, wat misconcepten tot gevolg heeft.

2.1.3 Voorkomen en Verhelpen

De processen voor het voorkomen en verhelpen van misconcepten komen veel met elkaar overeen. Bij het voorkomen van een misconcept moet echter een belangrijk onderscheid gemaakt worden of de kans op het ontwikkelen van het misconcept sterk aanwezig is of niet. In dit onderzoek wordt de kans op het ontwikkelen van een misconcept sterk aanwezig geacht wanneer de leerling over foute voorkennis beschikt, foute kennis aangeleerd krijgt of niet actief aan het leerproces deelneemt.

Wanneer de kans op het ontwikkelen van een misconcept niet sterk aanwezig is, is het proces van het voorkomen van een misconcept relatief simpel. Het concept moet door de docent gepresenteerd worden op een wijze die aansluit bij de belevingswereld en het cognitieve niveau van de leerling (Schoenmaker, Stomp & Valstar, 2002; Brown & Clement, 1989). Dit maakt het voor de leerling makkelijker om de nieuwe informatie eventueel aan voorkennis te koppelen, wat het leerproces versterkt. Daarna is het belangrijk dat wordt gecontroleerd of de leerling het goed begrepen heeft (Hollingsworth & Ybarra, 2017; Türkmen & Usta, 2007). Wanneer blijkt dat dit niet het geval is, moet de leerling daarvan bewust worden gemaakt en moet de uitleg herhaald worden. Wanneer de leerling het wel goed heeft, helpt herhaling bij het opslaan van de kennis (Hollingsworth & Ybarra, 2017; Türkmen & Usta, 2007; Schoenmaker, Stomp & Valstar, 2002; Brown & Clement, 1989).

Anderzijds is het proces complex en moet met zorg uitgevoerd worden, misconcepten zijn namelijk erg hardnekkig (Ma, 2007; Sorva, 2012; Schoenmaker, Stomp & Valstar, 2002; Türkmen & Usta,

2007). Wanneer er al sprake is van een misconcept of de kans op het ontwikkelen ervan sterk aanwezig is, blijkt het simpelweg aanbieden van de correcte informatie niet genoeg. Wanneer de kans op het ontwikkelen van het misconcept sterk aanwezig is of wanneer er sprake is van een misconcept bij de leerling, moet er een cognitief conflict teweeggebracht worden. Een cognitief conflict daagt de leerling namelijk uit om zijn/haar kennis te herzien (Ma, 2007; Sorva, 2012; Leonard, Kalinowski & Andrews, 2014). Voordat een misconcept opgelost kan worden, moet de leerling beseffen dat zijn/haar kennis onjuist is (Ma, 2007; Sorva, 2012; Leonard, Kalinowski & Andrews, 2014; Brown & Clement, 1989; Türkmen & Usta, 2007; Schoenmaker, Stomp & Valstar, 2002). Een aanpak daarvoor die in de literatuur veel beschreven wordt is het model van cognitieve verandering van Hewson en Hewson (1984) (Ma, 2007).

Een voorwaarde voor succes van het model van cognitieve verandering is dat de leerling een cognitief conflict ervaart en ontevreden is over zijn/haar huidige kennis. Doordat we constant opzoek zijn naar een balans tussen onze kennis en onze omgeving, kan dat al worden veroorzaakt door het besef dat de huidige kennis incorrect is (Piaget, 1977). Gelijker tijd moet de correcte informatie over het concept aanwezig zijn. Om aangenomen te worden moet het concept aan drie voorwaarden voldoen. Allereerst moet het concept begrijpelijk zijn. Dit houdt in dat de leerling het concept begrijpt en logisch vindt, maar deze nog niet wezenlijk voor waar aanneemt. De tweede voorwaarde is dat de nieuwe informatie aanneembaar is. De leerling moet het begrip als correct beschouwen, daarvoor moet het aansluiten bij zijn/haar belevingswereld. Tot slot moet de nieuwe informatie een voordeel geven aan de leerling. Er moet een goede reden zijn voor de leerling om de nieuwe informatie aan te nemen, bijvoorbeeld om een probleem op te kunnen lossen.

Daarnaast spelen de motivatie, epistemologische overtuigingen en leerstrategieën van de leerling een belangrijke rol bij de conceptuele verandering (Leonard, Kalinowski & Andrews, 2014; Ma, 2007). Wanneer de leerling niet of weinig gemotiveerd is tot leren, is het minder waarschijnlijk dat het cognitieve conflict plaats zal vinden of opgelost zal worden. Een goede motivatie impliceert overigens niet dat de leerling beter leert of minder misconcepten heeft, maar zorgt vaak wel voor een actieve leerhouding. Leerlingen die enkel gefixeerd zijn op het leren van feiten, leren overigens vaak niet de onderliggende werking van concepten.

De epistemologische overtuigingen van de leerling bepalen wat de leerling van de informatie vindt. De complexiteit van de informatie en de tijd die nodig is om de nieuwe informatie leren, hebben hier bijvoorbeeld een sterke invloed op. Twee andere bekende belangrijke epistemologische overtuigingen zijn zekerheid over de correctheid van de kennis en de mate waarin de leerling gelooft dat zijn/haar kennisniveau verder kan ontwikkelen. Een leerling die bijvoorbeeld geloofd dat zijn/haar intelligentieniveau niet verder ontwikkelt, gaan minder vaak de uitdaging aan om een cognitief conflict te overwinnen. Zij hebben veelal de overtuiging dat wanneer zij iets niet direct begrijpen, zij het wel nooit zullen begrijpen.

Tot slot bepalen de leerstrategieën van de leerling in hoeverre de conceptuele verandering kan plaatsvinden. Het bewust toepassen van cognitieve strategieën en het monitoren van begrip zijn hierbij belangrijke onderdelen. Cognitieve strategieën die een positieve invloed hebben op het leren

van informatie zijn herhaling van de stof, de nieuwe kennis uitwerken door oefeningen te maken en voorbeelden te gebruiken, de nieuwe informatie te organiseren door connecties met bestaande kennis te leggen en de informatie kritisch beoordelen. De mate van zelfregulatie is belangrijk bij het toepassen van deze strategieën door bijvoorbeeld te plannen en cognitieve strategieën telkens aan te passen.

2.1.4. Misconcepten Meten

Voor het meten van misconcepten zijn nog geen concrete algemene richtlijnen opgesteld. Wel zijn er onderzoeken naar misconcepten uitgevoerd waarvan aan de methode een voorbeeld genomen kan worden. Op basis van deze onderzoeken zijn aandachtspunten voor het meten van misconcepten beschreven.

Een methode die veel gebruikt wordt bij meetinstrumenten voor misconcepten is het geven van meerkeuze antwoorden (Ma, 2007). De onderzochte misconcepten worden dan eerst beschreven, hiervoor kan gebruik gemaakt worden van de literatuur of observaties tijdens de les. Vervolgens worden de misconcepten in het meetinstrument verwerkt door ze te vertalen naar een antwoord en ze weer te geven als een van de antwoordopties. Echter, het ontdekken van *alle* schema's en misconcepten door middel van observaties is onmogelijk (Ma, 2007). De antwoordopties zouden dan bestaan uit één correct antwoord, minstens één antwoord dat uit een beschreven misconcept zal voortvloeien en eventueel willekeurige foute antwoorden. Dat een leerling dan eenmaal een misconcept als antwoord zal kiezen, toont echter nog niet sterk aan dat hij/zij dat misconcept daadwerkelijk bezit. Deze fout zou ook afhankelijk kunnen zijn van fitheid, motivatie, leesfouten of gokken (Tew, 2010). Door meerdere vragen over hetzelfde concept te stellen met telkens hetzelfde soort antwoordopties, kan gemeten worden of consistent dezelfde fout gemaakt wordt. Bij meerkeuze antwoorden is de kans op gokken echter wel groter dan bij open antwoorden.

Wat gebruikelijk is bij meerkeuze testen is dat antwoorden van participanten in groepen worden geplaatst (Ma, 2007). Dehnadi en Bornat (2006) onderscheidden bijvoorbeeld drie groepen. Allereerste de consistente groep met participanten die consistent het hetzelfde misconcept toepasten. Daarnaast ook een inconsistent groep met participanten die inconsistent antwoord gaven. Over hen werd geconcludeerd dat zij vaak per opdracht een mening zochten achter het 'gedrag' van de computer (Ma, 2007; Dehnadi & Bornat, 2006) of gokten. Tot slot de blanco groep met participanten die bij de meeste opdrachten geen antwoord gaven. Een andere gebruikelijke groep is de correcte groep, zij kiezen consistent het correcte antwoord (Ma, 2007).

Anderzijds worden misconcepten vaak getoetst door middel van vragen met open antwoorden (Ma, 2007). Het voordeel hiervan is dat het makkelijker is om achterliggende gedachten van antwoorden te achterhalen, wat mogelijkheid geeft tot het ontdekken van nieuwe misconcepten. Het nadeel van open antwoorden is dat deze niet kwantitatief verwerkt kunnen worden, waardoor conclusies niet statistisch onderbouwd kunnen worden. Daarnaast maakt het geven van open antwoorden vaak 'toets moe' en kan het een negatief effect hebben op motivatie, waardoor men vaker onvolledige antwoorden geeft (Ma, 2007; Tew, 2010). Tot slot doen open vragen beroep op taalvaardigheden. Mensen met lage

taalvaardigheden kunnen hierdoor gedemotiveerd raken en het is de vraag in hoeverre deze interfereren met het meten van de taalvaardigheden.

2.2 Programmeeronderwijs

Er zijn nog geen richtlijnen voor efficiënt programmeeronderwijs en de meningen over de beste aanpak verschillen sterk, waardoor er wereldwijd zorgen ontstaan over de staat van het programmeeronderwijs (Ma, 2007; Sorva, 2012). Studenten presteren vaak slecht op toetsen en wanneer zij wel goed op toetsen presteren, lijken zij vaak alsnog niet over voldoende programmeervaardigheden te beschikken. Voor dit onderzoek wordt een kort overzicht gegeven van onderwijsstrategieën voor programmeeronderwijs, diens toepassingen en effecten en misconcepten bij programmeren.

2.2.1 Constructivisme en Objectivisme

Volgens het objectivisme leren we door in ons brein een spiegeling te maken van objecten uit de wereld om ons heen. Introductie programmeerlessen richten zich vaak op object-georiënteerd programmeren op een manier die aansluit bij de onderwijsstijl objectivisme (Ma, 2007). De interpretatie van waarnemingen is objectief en onafhankelijk van het menselijke brein, de waargenomen wereld is dus voor iedereen gelijk. Door waarnemingen te koppelen aan de werking van de wereld, doe je kennis op (Vrasidas, 2000; Bednar et al., 1991; Ma, 2007). Het resultaat van leren is het correcte begrip van het object. Het doel van onderwijs is bij objectivisme om de statische objectieve kennis over te brengen in het brein van de leerling, de leerling koppelt dit aan de werking van de wereld (Vrasidas, 2000; Bednar et al., 1991; Ma, 2007).

Echter zijn veel onderzoekers ervan overtuigd dat programmeeronderwijs zich zou moeten richten op het constructivisme*, de tegenhanger van het objectivisme* (Ma, 2007). Ervaringen die we opdoen, zijn volgens het constructivisme afhankelijk van onze eerdere ervaringen en bestaande cognitieve structuren zoals schema's. Door nieuwe informatie te verbinden aan eerdere ervaringen, verloopt het opslaan van kennis overigens efficiënter dan wanneer het in geïsoleerde context benaderd wordt (Schoenmaker, Stomp & Valstar, 2002; Türkmen & Usta, 2007; Brown & Clement, 1989; Stavy, 1991). Op deze wijze zijn de realiteit en kennis een product van onze gedachten, en voor iedereen anders (Ma, 2007; Sorva, 2012; Proulx, 2006). Het leerproces wordt vanuit het constructivisme beschouwd als een actief en recursief proces.

Het ervaringsgerichte aspect van het constructivisme is een belangrijk onderdeel voor programmeeronderwijs (Ma, 2007; Sorva, 2012). Programmeren is namelijk meer een toepassing in de praktijk, dan een theoretisch vakgebied. Het gaat meer om het begrijpen en kunnen toepassen van de kennis. Het simpelweg leren van feiten, zoals het objectivisme impliceert, is niet voldoende om deugdelijke programmeervaardigheden te ontwikkelen (Ma, 2007; Sorva, 2012). Wat betreft misconcepten heeft constructivisme als sterk punt het gebruik van cognitieve structuren zoals schema's, waardoor abstracte materie beter begrepen kan worden. Het oplossen van een misconcept behoeft bovendien een cognitief conflict, waarbij het inzetten van kennis en schema's een vereiste is (Ma, 2007; Sorva, 2012; Leonard, Kalinowski & Andrews, 2014). Constructivisme is dus nodig voor het verhelpen van een misconcept.

2.2.2 Programmeer(mis)concepten

De wereld van het programmeren is veelal abstract (Sorva, 2012; Ma, 2007). Hoe programmeeropdrachten door de computer verwerkt worden is namelijk niet waarneembaar en soms is de output van een programma niet fysiek. Daarnaast heeft iemand die begint met programmeren vaak niet de (juiste) basiskennis waar de eerste programmeerervaringen aan gerelateerd kunnen worden, net als wanneer er verbanden worden gelegd tussen bestaande schema's (Sorva, 2012; Anderson, 2010).

Doordat leerlingen vaak niet de juiste schema's beschikbaar hebben, worden nieuwe informatie en ervaringen vaak onterecht aan andere schema's gekoppeld. Hierdoor wordt de kan op het ontwikkelen van een misconcept aanzienlijk vergroot (Sorva, 2012). Door programmeerconcepten concreet te presenteren, worden leerlingen gesteund bij het ontwikkelen van een basis voor een valide schema. Programmeerconcepten kunnen concreter gemaakt worden door deze te vertalen naar de belevingswereld en het ontwikkelingsniveau van de leerling. Wanneer leerlingen de beginselen van een schema hebben ontwikkeld, kunnen zij meer abstracte concepten verwerken. In tegenstelling lijkt het wel effectief dat leerlingen juist direct aan reële programmeervraagstukken worden blootgesteld in plaats van aan versimpelde voorbeelden, deze conclusie is echter wel getrokken uit een onderzoek bij studenten van 18 jaar en ouder (Sorva, 2012).

Ben-Ari (2001) benoemd dat naast het programmeren de computer vaak zelf ook een abstract concept is, hoewel dit relatief eenvoudig concreet gemaakt kan worden. Wanneer een leerling geen reëel beeld heeft van wat zich in de computer afspeelt bij de uitvoer van een programma, kan het in sommige gevallen programmeerconcepten niet voldoende relateren aan de computer. Hierdoor zal de leerling het concept niet begrijpen of een misconcept ontwikkelen. Wanneer een leerling begrijpt hoe een computer werkt, heeft hij/zij een concreet valide schema waar nieuwe ervaringen aan gekoppeld kunnen worden (Sorva, 2012; Ben-Ari, 2001).

In de loop der tijd is er al enig onderzoek gedaan naar welke misconcepten voorkomen bij programmeren. Sorva (2012) heeft een overzicht gemaakt van programmeermisconcepten die in de literatuur beschreven worden (bijlage 2). Wat betreft misconcepten en programmeren zijn er echter ook nog grote gaten in de literatuur. Zo zijn onderzoeken vaak uitgevoerd in het (hoger)beroepsonderwijs (Ma, 2007; Sorva, 2012). Over misconcepten en programmeren bij kinderen en de rol van misconcepten ontbreekt nog veel kennis.

Drempelconcepten

Een ander fenomeen van concepten is drempelconcepten. Een drempel concept speelt een cruciale rol in het leerproces van een vak, het niet begrijpen ervan veroorzaakt namelijk een grote barrière (Sorva, 2012; Zander et al., 2007). Het begrip van deze concepten is noodzakelijk voor het beheersen van de vaardigheid (Kallia & Sentance, 2017). Wanneer een drempelconcept wel begrepen wordt, kan dit een positieve invloed hebben op het begrip van andere concepten. Andere kenmerken van drempelconcepten zijn dat ze niet teruggedraaid worden, het kan een complex concept zijn voor de leerling doordat het vaak niet in het huidige schema past en het integreert andere concepten in het begrip doordat er duidelijke verbanden zijn. Drempelconcepten zouden goed als leerdoelen fungeren in programmeeronderwijs (Kallia & Sentance, 2017; Sorva, 2012).

2.3 Leren

Om het onderscheid in het jonge en het oude kind in deelvragen 3 en 4 te onderbouwen, wordt programmeren beknopt gekoppeld aan de leer- en denkfasen van verschillende leeftijden.

2.3.1 Fasen

Piaget onderscheidt vier fasen in de ontwikkeling/het leren (Zimbardo, Johnson & McCann, 2013; Morra, Gobbo, Marini & Sheese, 2008; Lister, 2011; Piaget, 1952). De mate waarin een taak uitgevoerd of stof begrepen kan worden, hangt samen met de fase van de ontwikkeling van toepasselijke schema's. Het ontwikkelen van schema's zou gebeuren door middel van een wisselwerking tussen het individu en zijn omgeving, wat overeenkomsten laat zien met het constructivisme. Deze wisselwerking noemt Piaget adaptie, waarvoor twee processen verantwoordelijk zijn: assimilatie en accommodatie. Van assimilatie is sprake wanneer nieuwe informatie in schema's wordt opgenomen en van accommodatie is sprake wanneer schema's worden aangepast om nieuwe informatie op te kunnen nemen.

Er wordt onderscheid gemaakt in de klassieke theorieën en de nieuwe theorieën van Piaget. De klassieke theorie van Piaget beschrijft dat de ontwikkelingen naar een nieuwe fase zijn gebaseerd op de rijping van het brein (Lister, 2011). De nieuwe theorieën van Piaget omschrijven dezelfde fasen als de klassieke, echter zouden de ontwikkelingen zijn gebaseerd op ervaring in het domein. Het doorlopen van de fasen gebeurt dus per domein. Dat zou betekenen dat iemand bij het leren van rekenen in een andere fase kan zitten dan bij het leren van taal. De nieuwe theorieën zijn niet beschreven door Piaget zelf, maar door verscheidene wetenschappers op basis van nieuwe ontdekkingen over de ontwikkeling (Lister, 2011). De nieuwe theorieën zijn dus meer toegespitst op de hedendaagse kennis en realiteit (Morra, Gobbo, Marini & Sheese, 2008).

2.3.2 Fasen en Programmeren

De eerste fase voor ontwikkeling die Piaget omschrijft is de sensomotorische fase. In deze fase is er nog geen sprake van denken maar van reflexmatige en aangeboren gedragingen. Deze fase is toepasselijk voor kinderen tot ongeveer 2 jaar (Zimbardo, Johnson & McCann, 2013). Vanuit de nieuwe theorie betekent dit, toegespitst op programmeren, dat iemand in deze fase moeite heeft met het doorlopen van code en is er sprake van een instabiel schema (Teague, 2014; Lister, 2011). Daarop volgt de pre-operationele fase. In deze fase, die tot ongeveer het zesde levensjaar duurt, kunnen mentale representaties worden gemaakt op basis van directe waarnemingen (Zimbardo, Johnson & McCann, 2013; Lister, 2011). Voor het programmeren betekent dit dat iemand code beter kan herleiden, maar is zich nog weinig bewust van het doel van de code en de relatie tussen verschillende stukken code (Teague, 2014; Lister, 2011). Programmeurs in deze fase bepalen de uitkomst van de code op inductieve gevolgtrekkingen, gebaseerd op de input van de specifieke code. De volgende fase is de concreet operationele, deze zou tot ongeveer het elfde levensjaar duren. In deze fase leert iemand logische handelingen uitvoeren en wordt het figuratieve denken verder ontwikkelt; het maken van mentale representaties (Zimbardo, Johnson & McCann, 2013). Het logische redeneren blijft overigens wel gebonden aan concrete situaties, het generaliseren van kennis is nog moeilijk. Voor het programmeren heeft deze fase als gevolg dat iemand kan redeneren over code en het geheel van meerdere regels code kan inzien. Daarnaast kan een programmeur in deze fase bewust kennis

(concepten) toepassen bij het lezen en schrijven van concrete stukken code en de omkeerbaarheid van code beredeneren. Tot slot de formeel operationele fase. De ontwikkeling van het logisch redeneren is voltooid en er kan abstract gedacht worden, dit resulteert in het generaliseren van kennis waardoor er inzicht is in gevolgen van handelingen (Zimbardo, Johnson & McCann, 2013; Lister, 2011). Voor het programmeren resulteert dit in het logisch, consistent en systematisch beoordelen en schrijven van code. Er kunnen gemakkelijk abstracties gemaakt worden om complexe code te kunnen begrijpen en schrijven (Teague, 2014; Lister, 2011).

De theorieën van Piaget liggen ten grondslag aan deelvragen 3, 4 en 6. Ergens in de ontwikkeling van het kind vindt er een schakeling plaats, waardoor het abstracte denken mogelijk wordt. Er wordt verwacht dat dit voor het elfde levensjaar niet plaats zal vinden. Bij andere onderzoeken naar programmeren is het niet regulier om rekening te houden met de cognitieve fase van leerlingen. Deze onderzoeken zijn namelijk gericht op volwassen, waarbij de klassieke theorie van Piaget niet relevant is. Omdat dit onderzoek betrekking heeft tot kinderen, wordt deze theorie wel in acht genomen.

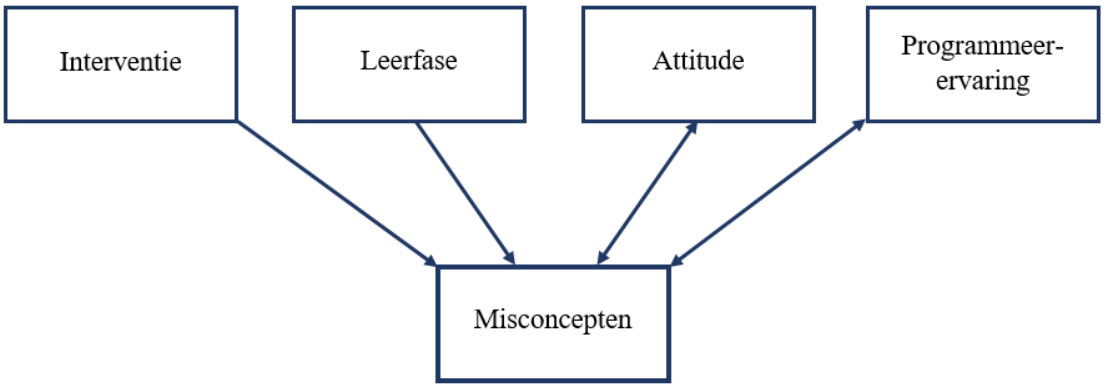
2.3.3 Attitude en Leren

Leerlingen met een positieve attitude over de te leren materie en over het leren zelf, hebben vaak een actievere leerhouding (Anghelache, 2013; Harlen & Crick, 2003; Rost, 2006). Dat houdt in dat zij leerdoelen opstellen en meer gemotiveerd zijn om te leren (Anghelache, 2013; Harlen & Crick, 2003; Rehman & Haider, 2013). Die positieve attitude hoeft overigens niet over de materie zelf te gaan, maar kan bijvoorbeeld gaan over de reden dat de materie geleerd wordt (Rost, 2006). Bijvoorbeeld dat je moet leren programmeren, omdat je graag zelf een computerspel wil maken.

De docenten en ouders van een kind spelen een belangrijke rol bij de ontwikkeling van de attitude van een kind (Castejón, Gilar & Pérez, 2006). Leerlingen met een positieve attitude over de materie, zijn vaak gemotiveerder tot het leren van de materie. Gemotiveerde leerlingen tonen dus vaak een actieve leerhouding, met betere leerprestaties tot gevolg (Castejón, Gilar & Pérez, 2006; Schunk & Zimmerman, 2008; Harlen & Crick, 2003; Anghelache, 2013).

2.4 Conceptueel Model

Het onderstaande model geeft een weergave van de gemeten variabelen en hun invloed op/samenhang met misconcepten.



Figuur 1. Conceptueel model

3. Onderzoeksdesign

In dit hoofdstuk wordt beschreven hoe het onderzoek is opgezet. De methode, de doelgroep, de instrumenten en de procedure worden in detail beschreven. Tot slot wordt beschreven op welke wijze de data wordt geanalyseerd.

3.1 Onderzoeksmethode

In dit onderzoek zal een experiment uitgevoerd worden waarna door middel van een vragenlijst data verzameld zal worden. De verworven data zal kwantitatief verwerkt worden. De participanten worden opgedeeld in experimentele en controle groepen. Alle groepen zullen vier programmeerlessen volgen gegeven door een ervaren informatica student van de TU Delft [verder genoemd als 'gastdocent'] en door de onderzoeker zelf. De experimentele groep kreeg daarbij extra uitleg gericht op het voorkomen/verhelpen van misconcepten omtrent 'variabelen en waarden'. Aan het einde van de vierde programmeerles vullen de participanten een vragenlijst in waarmee gemeten zal worden in hoeverre misconcepten van verschillende programmeerconcepten aanwezig zijn.

Er is gekozen voor een experimenteel onderzoek omdat getracht wordt het effect te meten van de didactiek van de extra uitleg op het ontwikkelen van programmeermisconcepten. Er is voor gekozen de verworven data kwantitatief te benaderen, om resultaten statistisch te onderbouwen.

3.2 Onderzoeksdoelgroep

De onderzochte populatie bestaat uit leerlingen uit de groepen 5 tot en met 8 van de basisschool. De getrokken steekproef is te beschrijven als een clustersteekproef. Hierbij worden bestaande groepen onderzocht, waarbij ieder cluster alle elementen van de gehele populatie bevat (Vos, 2009). Alle basisscholen die zich vrijwillig hebben aangemeld voor de programmeerlessen van het project van de TU Delft, omschreven in hoofdstuk 1.1, en de lessen in de periode maart 2018 tot en met mei 2018 volgen, worden benaderd mee om mee te doen aan het onderzoek. Dit resulteerde in deelname van 1 basisschool, de Nicolaasschool te Rotterdam. Van deze basisschool doen 13 klassen uit groepen 5 tot en met 8 mee. Daarnaast nemen ook groepen 6 en combinatie 6-7 van basisschool Het Blokhuus te Hoevelaken deel aan het onderzoek. Een vertegenwoordiger heeft de school via persoonlijke communicatie met de TU Delft, normaliter komen alleen basisscholen uit Rotterdam in aanmerking. In totaal zullen 369 leerlingen de programmeerlessen volgen in de periode van 13 maart 2018 tot en met 22 mei 2018.

Tabel 1

Overzicht van onderzoeksdoelgroep

Groep	Aantal Klassen	Aantal Kinderen	Experimenteel	Controle
5	4	86	43	43
6	4	108	80	28
Combinatie 6/7	1	26	0	26
7	3	76	49	27
8	3	73	24	49
Totaal	15	369	196	173

De indeling van de kinderen en experimentele groepen, vond plaats op basis van hun klas. Het indelen van de klassen in de experimentele en controlegroep is per leerjaar aangepakt. Waar mogelijk werd de helft van het aantal klassen willekeurig in de experimentele groep en de andere helft in de controlegroep ingedeeld. Voor groep 8 is willekeurig bepaald of er een of twee klassen in de experimentele groep ingedeeld en vervolgens willekeurig welke klas.

Alle leerlingen uit de beschreven groepen zullen deelnemen aan de lessen. Er zal echter alleen data verzameld worden van de leerlingen van wie ouders daarvoor expliciet toestemming hebben gegeven door middel van een toestemmingsformulier (bijlage 3).

3.3 Onderzoeksinstrument

Het lesmateriaal voor de programmeerlessen is reeds ontwikkeld door Felienne Hermans, assistent professor aan de TU Delft. Deze lessen worden gebruikt bij projecten van de TU Delft in het onderwijs. In de eerste les wordt een digitaal potlood geprogrammeerd om willekeurige spirograven* te tekenen (bijlage 4). In deze les worden de begrippen ‘x-as en y-as’ en variabelen geïntroduceerd. Tevens leren de leerlingen kennismaken met basisprincipes van Scratch zoals het aan elkaar schuiven van blokken. In de tweede les wordt een creatieve rekenmachine gemaakt (bijlage 6). In deze les gaan werken leerlingen weer met variabelen en leren zij meer basisprincipes van Scratch kennen en toepassen. In de derde les gaan de leerlingen zelfstandig aan de slag met Scratch door een verhaal te programmeren (bijlage 8). Zij leren zij het begrip ‘lijsten’* kennen en werken zij wederom met variabelen. In de vierde les gaan de leerlingen een minigame maken en gebruiken zij blokken waar zij in de vorige lessen ook mee gewerkt hebben (bijlage 10).

De Vragenlijst

De door de onderzoeker ontwikkelde vragenlijst bestaat uit items die demografische kenmerken, misconcepten en attitude over programmeren meten (bijlage 13).

De te meten misconcepten zijn onder andere gekozen op basis van de lijst misconcepten uit de literatuur (bijlage 2). Misconcepten die sterk verwant zijn aan een specifieke programmeertaal of rekenen zijn uitgesloten. Van misconcepten die sterk met elkaar overeenkwamen, is de minst uitgebreide versie uitgesloten. Vervolgens zijn de misconcepten uitgesloten waarmee de leerlingen in de vier lessen geen kennis mee hebben gemaakt. Van de overige lijst misconcepten zijn de bijbehorende en overkoepelende concepten omschreven.

Er worden in totaal 14 misconcepten gemeten, onder te verdelen in 5 concepten. De misconcepten gemarkeerd met een asterisk, zijn gebaseerd op observaties van antwoorden op de wisbordjes* van de leerlingen. Tijdens de lessen werden vragen gesteld aan de leerlingen, die hun antwoorden op de wisbordjes moesten schrijven. Op basis van deze antwoorden en toelichtingen van de leerlingen, zijn nieuwe misconcepten omschreven.

Tabel 2

Overzicht concepten, misconcepten en bijbehorende itemnummers uit de vragenlijst

Concept	Misconcept 1	Misconcept 2	Misconcept 3	Itemnummers
Variabelen en waarden	"Een variabele heeft meerdere waarden"	"Nieuwe waarden van een variabele worden opgeteld"*	"Een variabele heeft een statische waarde"*	14, 19, 24, 29
Variabelen en namen	"De variabele naam bepaald de waarde"	"De waarde van een variabele wordt aangepast naar zijn naam"	"De variabele krijgt alleen de waarde toegepast wanneer deze past bij zijn naam"*	15, 20, 25, 30
Volgorde uitvoering programma	"Fouten bij het volgen van de volgorde"	"Alle statements worden op tegelijkertijd verwerkt"*	"Statements worden ook verwerkt wanneer dit niet is geprogrammeerd"*	16, 21, 26, 31
Taalbegrip van de machine	"De machine kan strings interpreteren"	"De machine vertaald integers naar strings"*	-	17, 22, 27
Logica	"Onlogische opdrachten worden niet uitgevoerd"	"Onlogische opdrachten worden aangepast"*	"Er is geen onderscheid tussen het programma en de werkelijke wereld"*	18, 23, 28, 32

Het construct attitude bestaat uit items 33 tot en met 39.

3.4 Procedure

De eerste twee lessen worden klassikaal gegeven. De gastdocent gebruikt bijlagen 4 en 6 als draaiboek van de les, zodat alle klassen dezelfde opdrachten uitvoeren en dezelfde uitleg krijgen. Een leerling verwerkt de opdrachten in de computer, de andere leerlingen kijken via de beamer mee. Alle leerlingen krijgen een wisbordje waar zij de antwoorden, op de tussendoor gestelde vragen, op kunnen schrijven. Vervolgens houden zij hun wisbordje omhoog zodat de gastdocent kan indiceren of de leerlingen de stof begrijpen. De vragen zijn afhankelijk van het niveau van de leerlingen en het verloop van de les. Een typische vraag is het voorspellen van de output van het programma dat via de beamer wordt weergegeven.

In de eerste programmeerles wordt besproken wat programmeren is en waarvoor het gebruikt kan worden. Het doel is een introductie voor programmeren en leerlingen enthousiasmeren voor de lessen. Vervolgens wordt de eerste programmeerles uitgevoerd. Bij de introductie van variabelen wordt bij de experimentele groepen experiment 1 uitgevoerd (bijlage 5). Aan het begin van de tweede les wordt eerst kort teruggeblikt op wat er in de vorige les gedaan is en welke nieuwe begrippen ze hebben geleerd. Vervolgens wordt de programmeerles uitgevoerd. Bij de herhaling van het begrip variabele wordt bij de experimentele het tweede experiment uitgevoerd (bijlage 7).

In de derde les gaan de leerlingen zelfstandig met Scratch programmeren, daarvoor gaan zij naar het computerlokaal. Op de computers is het pdf van les 3 (bijlage 8) vast voor ze klaargezet. De les wordt

begonnen door een korte terugblik op de vorige les en de uitleg met de werkwijze van deze les. Ieder kind gaat individueel de opdrachten in Scratch verwerken, maar zij mogen het pdf met de opdracht wel gezamenlijk doorlopen. Bij vragen mogen zij met elkaar overleggen of het de gast- of eigen docent vragen. De introductie van de opdrachten in het pdf wordt klassikaal doorlopen, het pdf is door middel van de beamer zichtbaar voor iedereen weergegeven. Bij de experimentele groepen wordt bij de herhaling van het begrip variabele het derde experiment uitgevoerd (bijlage 9). De eerste opdracht die in Scratch uitgevoerd moet worden, wordt klassikaal gedaan. Daarna wordt laten zien hoe het programma werkt wanneer de opdracht klaar is. Het voltooide programma wordt het begin van de les klaargezet.

In de vierde en tevens laatste les gaan de leerlingen op dezelfde wijze als in de derde les aan de slag. De introductie is tevens hetzelfde als in de derde les. Echter is er voor de experimentele groepen geen nieuw experiment meer, maar wordt er een samenvatting gegeven van de experimenten (bijlage 11). Daarnaast wordt aan alle klassen vertelt dat de leerlingen halverwege de les moeten stoppen met de opdrachten voor het maken van de vragenlijst.

Twintig minuten voor het einde van de les moeten alle leerlingen stoppen met waar zij mee bezig zijn en luisteren naar de uitleg voor de afname van de vragenlijst. Ze moeten Scratch en het pdf niet afsluiten. Er wordt uitgelegd dat wanneer de leerlingen hun naam horen, zij naar de website ‘link.shareyourwork.nl’ moeten gaan, deze staat klaar in de werkbalkfavorieten*. Vervolgens worden de namen opgenoemd van de leerlingen uit de klas die wel een toestemmingsformulier hebben ingeleverd. De leerlingen van wie hun naam niet is opgenoemd, krijgen de opdracht verder te werken met de opdracht van Scratch. Zij mogen vanaf nu niet meer praten, niet meer samenwerken en geen vragenstellen. De leerlingen die wel de vragenlijst gaan maken wordt verteld op knop ‘1’ te klikken, waardoor de vragenlijst opent. De introductie van de vragenlijst wordt klassikaal doorgenomen, tot aan de melding dat de leerlingen zelfstandig verder werken. Tijdens de afname van de vragenlijst is het geheel stil in het klaslokaal. Wanneer leerlingen een vraag hebben, mogen zij hun handopsteken en komt de gastdocent helpen. De gastdocenten houden zich hierbij aan de ‘instructies afname vragenlijst’ (bijlage 12).

3.5 Analyses

Om deelvragen 1 tot en met 4 te kunnen beantwoorden, moet achterhaald worden of er een significant verschil is in het aantal misconcepten tussen de omschreven groepen. Omdat de variabelen waarmee de participanten in groepen worden gesplitst nominaal zijn en de testvariabele van ratiomeetniveau is, wordt er gebruik gemaakt van de t-toesta (Baarda, Van Dijkum & De Goede, 2014). Wanneer er sprake is van een significant verschil ($p < 0,05$), zal door middel van frequenties beoordeeld worden welke misconcepten bij beide groepen voorkomen. Bij deelvragen 1 en 3 zullen misconcepten behorende bij het concept ‘variabelen en waarden’ niet berekend worden, omdat deze in deelvragen 2 en 4 uitgelicht worden.

Omdat er bij deelvragen 5 en 6 sprake is van ordinale variabelen, zal Spearman’s rangcorrelatie gebruikt worden om te achterhalen of er samenhang is tussen de beschreven variabelen. De correlatiecoëfficiënt wordt alleen aangenomen bij een p-waarde kleiner dan 0,05.

4. Onderzoeksresultaten

Voordat in gegaan wordt op de onderzoeksresultaten, zal de uitvoering van het onderzoek beschreven worden door middel van het verloop van en de respons op het onderzoek. Vervolgens worden in de resultaten de bevindingen van de statische analyses per deelvraag besproken.

4.1 Uitvoering Onderzoek

Bij het verloop van het onderzoek worden ongeplande en opvallende zaken besproken die een storingsfactor voor de resultaten zouden kunnen vormen. Vervolgens wordt de uiteindelijke respons van het onderzoek besproken.

4.1.1 Verloop

De onregelmatigheden van de lessen en de vragenlijstafname worden apart besproken, beide in chronologische volgorde. Er wordt telkens naar de klassen verwezen, zoals die ook in tabel 1 beschreven staan. De beschrijvingen zijn gebaseerd op de korte verslaglegging van onregelmatigheden per les, door zowel de onderzoeker als de gastdocent. In bijlage 14 is een overzicht opgenomen van de planning van de lessen en de indeling van klassen bij de gastdocent en de onderzoeker.

Lessen

Les 1 van klas 6C verliep chaotisch. De gastdocent gaf de les ‘Spirograaf’ voor het eerst waardoor hij tijdens de les een aantal keren op de hand-out moest kijken om de volgende stappen door te nemen. Terwijl hij dat deed, werd het onrustig in de klas doordat de leerling de tijd kregen om met elkaar te kletsen. In het orde brengen in de klas ging regelmatig veel tijd zetten, waardoor er minder tijd over bleef om de lesstof te behandelen en de laatste twee pagina’s van de les niet behandeld zijn. Daarnaast waren de wisbordjes pas halverwege de les aanwezig, waardoor de leerling de eerste helft minder actief mee konden doen.

Les 1 van klas 5D verliep tevens erg chaotisch. Er waren veel leerlingen die de les verstoorden door de andere leerlingen af te leiden door bijvoorbeeld te kletsen. Het overgrote deel van de leerlingen vond de lesstof erg moeilijk en gaf aan veel niet te begrijpen. Doordat er veel uitleg herhaald moest worden en de verstoringen, is maar ongeveer de halve les behandeld. De uitleg en het oefenen met variabelen is hierdoor overgeslagen. De school beschrijft deze klas als ‘moeizaam’ doordat veel leerlingen leerproblemen ervaren en veel leerlingen afleiders zijn voor andere leerlingen. In de volgende lessen met deze klas kwamen deze factoren in veel mindere mate naar voren.

Les 1 van klas 8C verliep slecht. Deze klas is net als groep 5D ‘moeizaam’. Er waren veel verstoringen en veel leerlingen gaven aan de uitleg niet te begrijpen, waardoor deze soms herhaald werd wat veel tijd kostte. Daar bovenop liep halverwege de les de computer vast. Scratch werd opnieuw opgestart, waardoor de code tot dan toe verloren ging. Toen alles weer werkte, heeft de onderzoeker eerst de voormalige code zelfstandig nagemaakt, zodat dit niet klassikaal opnieuw hoefde. Terwijl de problemen met de computer opgelost werden, werd het erg rumoerig in de klas. De lesstof werd voor ongeveer drie kwart behandeld, het experiment is niet uitgevoerd.

In les 2 van klas 6C deden zich problemen voor het experiment. Veel kinderen begrepen het voorbeeld niet. Het voorbeeld ging over het veranderen van de waarde variabelen, om dit relevant werd het voorbeeld gekoppeld aan het veranderen van leeftijd wanneer je jarig bent. Allereerst ontstond er een discussie doordat de leeftijd van de kinderen verschilde. Daarnaast begrepen de kinderen de uitleg niet. De manier waarop de kinderen reageerden gaf zowel de onderzoeker als gastdocent de indruk dat ze moeite hadden met de twee lijnen van uitleg: de context van leeftijd en de nieuwe informatie over variabelen. Dit is direct aangepast door de alternatieve uitleg te gebruiken, die wel begrepen werd. Vanaf dit moment is alleen de alternatieve uitleg gebruikt, beschreven in bijlage 7.

Les 2 van klas 8C verliep goed, maar de inhoud is wel aangepast op andere lessen. Het stuk uitleg over variabelen dat in les 1 behandeld had moeten worden, is ingehaald. De leerlingen hadden deze kennis nodig, omdat les 2 hierop ingaat. De uitleg is ingehaald bij de introductie van variabelen in les 2.

Les 3 ervoeren veel leerlingen van alle klassen als moeilijk. Ze moesten voor het eerst zelf aan de slag met Scratch. Sommigen hadden moeilijkheden met het vinden van de juiste blokken en gebruiken van de blokken (bijvoorbeeld het aan en in elkaar schuiven). In het uitproberen van Scratch ging bij veel leerlingen veel tijd zitten. Veel leerlingen kregen de opdracht niet af, waardoor uiteindelijk niet alle leerlingen dezelfde stof hebben behandeld.

Les 3 van klas 6 uit Hoevelaken werd klassikaal uitgevoerd, doordat Scratch nog niet op de iPads geïnstalleerd was. Hierdoor hebben de leerlingen minder zelfstandige ervaring met Scratch opgedaan. De opdracht is, in tegenstelling tot bij de meeste leerlingen, overigens wel geheel afgerond.

Tussen de leerlingen van de scholen uit Rotterdam en Hoevelaken was een opmerkelijk verschil in gedrag tijdens de lessen. Leerlingen uit Rotterdam waren over het algemeen snel afgeleid, veel rumoerig en deden vaak ongeïnteresseerd overkomen op de onderzoeker en gastdocent door passieve houdingen. Leerling uit Hoevelaken toonden over het algemeen een actieve leerhouding. De lesstof was in Hoevelaken vaak snel afgerond, waardoor er tijd was om ideeën van leerlingen klassikaal uit te werken.

Vragenlijstafname

Tijdens de afname van de vragenlijst in klas 8C was het regelmatig rumoerig. De onderzoeker vroeg regelmatig de klas in stilte verder te werken. Dit kan een dubbele afleiding hebben gevormd voor de leerlingen de die vragenlijst invulden: de rumoerigheid zelf en de opmerkingen van de onderzoeker.

4.1.2 Respons

Tot de respons behoren alleen leerlingen die de vragenlijst hebben ingevuld. Een overzicht van de respons en enkele kenmerken zijn in tabel 3 weergegeven. De respons is beduidend kleiner dan de onderzoeksdoelgroep (bestaande uit 369 leerlingen). Het toestemmingsformulier werd door veel kinderen niet ingeleverd. Dat ouders geen toestemming wilden geven, onoplettendheid van leerlingen met de formulieren en een gebrek aan motivatie voor de programmeerlessen worden hiervoor als oorzaken geacht.

Tabel 3

Overzicht respons met demografische kenmerken

Groep	Aantal	Experimentele Groep	Controle Groep	Gem. Leeftijd	Jongens	Meisjes
5	46 (24,1%)	22 (20,2%)	24 (29,3%)	9,8 (sd: 0,61)	20 (43,5%)	26 (56,5%)
6	64 (33,5%)	43 (39,4%)	21 (25,6%)	9,8 (sd: 0,69)	26 (40,6%)	38 (59,4%)
7	35(18,3%)	3 (2,8%)	32 (39,0%)	11,4 (sd: 0,94)	17 (48,6%)	18 (51,4%)
8	46(24,1%)	41 (37,6%)	5 (6,1%)	11,7 (sd: 0,66)	23 (50%)	23 (50%)
Totaal	191 (100%)	109 (57,1%)	82 (42,9%)	10,6 (sd: 1,13)	86 (45,5%)	105 (54,5%)

4.2 Resultaten

In deze paragraaf worden de resultaten van de analyses beschreven. De resultaten voor deelvragen 1 en 2 zijn beschreven in sub paragraaf 1. De resultaten voor deelvragen 3 en 4 zijn beschreven in sub paragraaf 2. De resultaten voor deelvragen 5 en 6 worden afzonderlijk van elkaar besproken in sub paragrafen 3 en 4.

4.2.1 Homogeniteit van de Vragenlijstschalen

Om de betrouwbaarheid van het onderzoeksinstrument te meten en daarmee ook een beroep te doen op de betrouwbaarheid, wordt normaliter een homogenitieitsanalyse, zoals een Cronbach's Alpha [CA], uitgevoerd op de items van een construct. In dit onderzoek is daar betreffende de programmeeritems (P1 t/m P19) geen sprake van. De items zijn bewust inconsistent opgesteld, wat een lage CA zou uitlokken. Deze lage CA zou onterechte twijfels voor de betrouwbaarheid van het onderzoek teweegbrengen.

Het construct 'Attitude' is meer gebruikelijk opgesteld, waardoor de waarde van de CA wel een toegevoegde waarde heeft. Allereerst zijn items A2, A4 en 5 gehercodeerd, zodat bij alle items een hogere score een positievere attitude impliceert. De CA van items A1 tot en met A7 heeft een waarde van 0,50. Doordat items A4 is verwijderd uit de analyses, heeft de CA een waarde van 0,60.

4.2.2 Misconcepten en het Experiment

Er zijn twee analyses uitgevoerd. Allereerst een analyse voor het meten van het verschil in het aantal misconcepten tussen de experimentele en controlegroep, exclusief het concept 'variabelen en waarden'. Daarnaast een analyse voor het meten van het verschil in misconcepten met betrekking tot 'variabelen en waarden' tussen de experimentele en controlegroep. Voordat er analyses zijn uitgevoerd, is de geschiktheid van de data geëvalueerd. Om te kunnen beoordelen of er sprake is van een misconcept, moet de respondent alle items van het concept beantwoord hebben. Bij 22 respondenten was hiervan geen sprake, dus zijn hun antwoorden uit de dataset verwijderd. Daarnaast is bij 2 respondenten een deel van de antwoorden verwijderd. Enkel van het concept 'taalbegrip van de machine' hebben zij alle items beantwoord. In de analyse voor dit concept zijn deze respondenten

meegenomen. Bij de analyse van andere concepten en de berekening van het totale aantal misconcepten zijn deze respondenten niet meegenomen.

De analyses zijn uitgevoerd met data van 169 respondenten. In tabel 4 is een overzicht van de verdeling van de respondenten per experimentele en controlegroep en het aantal misconcepten dat berekend is. Met de kolommen 'n = ...' wordt het aantal misconcepten (min.=0, max.=4) aangeduid van de concepten 'Variabelen en Namen', 'Volgorde Uitvoering Programma', 'Taalbegrip van de Machine' en 'Logica'. Het concept 'Variabelen en Waarden' (min.=0, max. =2), concept 1, wordt afzonderlijk beschreven voor deelvraag 2.

Tabel 4

Overzicht van het aantal respondenten per aantal misconcepten bij de experimentele en controle groepen.

Groep	n = 0	n = 1	n = 2	n = 3	n = 4	Totaal	Concept 1 n = 1
Experimenteel	41 (42,3%)	41 (42,3%)	14 (14,4%)	1 (1,0%)	0	99 (100%)	34 (67,0%)
Controle	23 (32,9%)	34 (48,6%)	11 (15,7%)	2 (2,9%)	0	70(100%)	25 (35,7%)

Deelvraag 1

Om te meten of er sprake is van een verschil in het aantal misconcepten bij de experimentele en controlegroep, is er een t-toets uitgevoerd. Het aantal misconcepten is berekend over alle concepten, exclusief het concept 'Variabelen en Waarden'. Gemiddeld genomen is er geen verschil in het aantal misconcepten tussen de experimentele (gem.= 0,73; sd = 0,74) en de controlegroep (gem.= 0,89; sd = 0,77) (t = -1,35; df = 167; p > 0,05). Omdat er geen significant verschil is in het aantal misconcepten tussen beide groepen, wordt er geen onderscheid gemaakt in de soorten misconcepten door middel van een frequentieanalyse.

Deelvraag 2

Er is wederom een meting uitgevoerd naar het verschil tussen het aantal misconcepten bij de experimentele en controlegroep, ditmaal met misconcepten van het concept 'Variabelen en Waarden'. Gemiddeld genomen is er geen verschil in het aantal misconcepten van 'Variabelen en Waarden' tussen de experimentele groep (gem.= 0,22; sd = 0,42) en de controlegroep (gem.= 0,33; sd = 0,47) (t = -1,51; df = 136,81; p > 0,05). Omdat er geen significant verschil is tussen beide groepen, worden de verschillende soorten misconcepten niet geanalyseerd.

4.2.3 Misconcepten en Leeftijd

Voorafgaand aan de analyses voor deelvragen 4 en 5, zijn de respondenten op basis van leeftijd ingedeeld in twee klassen: 'Jong' en 'Oud'. Met het jonge kind wordt verwezen naar respondenten van 10 jaar en jonger, met het oude kind wordt verwezen naar respondenten van 11 jaar en ouder. De splitsing is gebaseerd op de theorie van Piaget, zoals omschreven in het theoretisch kader. Tabel 5 geeft een beknopt overzicht van het aantal misconcepten voor het jonge en het oude kind.

Tabel 5

Overzicht van het aantal respondenten per aantal misconcepten bij het jonge en het oude kind.

Leeftijd	n = 0	n = 1	n = 2	n = 3	n = 4	Totaal	Concept 1 n = 1
Jong	32 (47,5%)	41 (44,9%)	15 (16,9%)	3 (3,4%)	0	89 (100%)	19 (21,9%)
Oud	35 (43,8%)	35 (43,8%)	10 (12,5%)	0	0	80 (100%)	26 (32,6%)

Om deelvragen 4 en 5 te beantwoorden, zijn t-toetsen uitgevoerd. Er is wederom naar het aantal misconcepten gekeken, ditmaal op basis van de leeftijd van het kind. Bij de misconcepten wordt er wederom onderscheid gemaakt in misconcepten behorende bij het concept 'Variabelen en Waarden' en de overige misconcepten.

Gemiddeld genomen is er geen verschil in het aantal misconcepten tussen het jonge kind (gem.= 0,89; sd = 0,80) en het oude kind (gem.= 0,69; sd 0,69) (t = 1,75; df = 166,57; p > 0,05). Voor het aantal misconcepten met betrekking tot het concept 'Variabelen en Waarden' is tevens geen verschil tussen het jonge kind (gem.= 0,21; sd = 0,41) en het oude kind (gem.= 0,33; sd = 0,47) (t = -1,63; df = 157; p > 0,05).

4.2.4 Misconcepten en Attitude

Er is gemeten of er sprake is van een samenhang tussen attitude en het aantal misconcepten dat een leerling heeft, zoals beschreven in deelvraag 5. Om te kunnen toetsen met het construct 'attitude' is een totaalscore (min. = 6; max.= 18) opgesteld door items 33, 34, 35, 37, 38 en 39. In tegenstelling tot deelvragen 1 tot en met 4, houdt 'misconcepten' bij deze meting het totale aan misconcepten in, inclusief van het concept 'Variabelen en Waarden'.

Data van respondenten die op een of meer van deze items niet hebben geantwoord, wordt uitgesloten in de meting. De respons op het construct attitude is per groep aangegeven in tabel 6.

Tabel 6

Respons en kengetallen per groep op het construct attitude.

Kengetal	Groep 5	Groep 6	Groep 7	Groep 8	Totaal
N	20 (14,2%)	55 (39,0%)	26 (18,4%)	40 (28,4%)	141 (100%)
Gemiddelde	15,25	15,67	15,73	14,05	15,16
sd	1,92	2,12	2,12	2,16	2,20

Op basis van Spearman's rangcorrelatie blijkt er geen samenhang te zijn tussen attitude en het aantal misconcepten (r_s = 0,03; p > 0,05). Een betere attitude over programmeren heeft dus geen invloed op het aantal misconcepten dat een leerling ontwikkelt.

4.2.5 Misconcepten en Ervaring

Tot slot is er gemeten of er sprake is van een samenhang tussen eerdere programmeerervaring en het aantal misconcepten, zoals beschreven in deelvraag 6. Eerdere programmeerervaring houdt de

ervaring die leerlingen al hadden voor dit onderzoek. Met het aantal misconcepten wordt het aantal misconcepten op alle concepten bedoeld.

Van de 169 respondenten, hadden er 28 (16,6%) eerdere programmeerervaring. Daarvan hadden 24 respondenten ervaring met Scratch. Er waren 7 respondenten die ervaring hadden opgedaan met Scratch op school, allen leerling van de basisschool Het Blokhuus te Hoevelaken. Daarnaast hadden respondenten ervaring met Lego Mindstorms (n=3), Java (n=2) en Minecraft (n=2). Er waren 3 respondenten die met 2 programmeertalen ervaring hadden.

Er is een Spearman's rangcorrelatie uitgevoerd waaruit blijkt dat er sprake is van een zwakke negatieve samenhang tussen programmeerervaring en het aantal misconcepten ($r_s = -0,21$; $p < 0,01$). Leerlingen met meer programmeerervaring ontwikkelden minder misconcepten dan leerlingen zonder programmeerervaring.

5. Conclusie, Discussie en Aanbevelingen

De beschreven onderzoeksresultaten zijn verwoord naar conclusies, welke geïnterpreteerd zijn en in theoretische en praktische context plaatst zijn. Vervolgens is de kwaliteit van de onderzoeksresultaten bediscussieerd. Het onderzoek wordt afgesloten met aanbevelingen naar de opdrachtgever.

5.1 Conclusies

In de inleiding van dit onderzoek zijn de maatschappelijke context van programmeeronderwijs en de rol de rol van de opdrachtgever hierin besproken. Een combinatie van de staat van het huidige programmeeronderwijs en eerdere onderzoeken van de opdrachtgever gaven aanleiding om te onderzoeken welke misconcepten voorkomen bij kinderen uit groepen 5 tot en met 8 uit het primair onderwijs. Het doel was het opdoen van wetenschappelijk onderbouwde kennis en het exploreren van mogelijkheden voor relevant vervolgonderzoek. De resultaten zouden voor de opdrachtgeven ten grondslag kunnen liggen aan vervolgonderzoek en gebruikt kunnen worden om de kwaliteit van het programmeeronderwijs te verbeteren. Daarnaast is er een meetinstrument voor misconcepten ontwikkeld met een, voor de opdrachtgever, vernieuwde meetwijze die bij vervolgonderzoeken. Tot slot kan er met de ontwikkelde dataset verdere analyses uitgevoerd worden.

5.1.1 Misconcepten en Didactiek

Voor deelvragen 1 en 2 werd het verschil in misconcepten gemeten tussen de experimentele en controlegroep (pagina 2). De experimentele groep heeft gedurende de vier programmeerlessen een nieuwe vorm van uitleg gevolgd gericht op het voorkomen/verhelpen van misconcepten met betrekking tot ‘Variabelen en Waarden’. De eerste deelvraag betrok misconcepten afgeleid van de concepten ‘Variabelen en Namen’, ‘Volgorde Uitvoering Programma’, ‘Taalbegrip van de Machine’ en ‘Logica’. De tweede deelvraag betrok misconcepten afgeleid van het concept ‘Variabelen en Waarden’. Uit beide analyses blijkt er geen verschil te zijn in het aantal misconcepten tussen de experimentele en controlegroep. De voor dit onderzoek ontwikkelde uitleg over variabelen en waarden heeft geen effect op het aantal misconcepten dat leerlingen ontwikkelen.

De uitleg speelde in op het feit dat de wijze waarop kennis aangeboden wordt, onderdeel is van het ontwikkelproces van een misconcept (Ma, 2007; Holland, Griffiths & Woodman, 1997). Er zouden meerdere oorzaken ten grondslag kunnen liggen aan het feit dat de uitleg geen evident effect heeft. Allereerst doordat het effect van een les moeilijk te meten is, aangezien er meerdere factoren van invloed zijn. Tevens zou het kunnen zijn dat de uitleg niet deugdelijk genoeg is om een verschil teweeg te brengen, bijvoorbeeld doordat het gekozen metafoor niet aansluit bij de belevingswereld van een kind. Anderzijds zijn vier lessen wellicht niet genoeg om de kennis afdoende te presenteren. Daarnaast zou het kunnen dat didactiek niet doorslaggevend is in het al dan ontwikkelen van een misconcept. Ingeval het de vraag is bij welk aspect van het ontwikkelproces van een misconcept dan wel efficiënt is om misconcepten proberen te voorkomen of op te lossen. Hier wordt in de derde paragraaf verder op in gegaan.

5.1.2 Misconcepten en Cognitieve Fasen

Voor deelvragen 3 en 4 werd het verschil in misconcepten gemeten tussen het jonge en het oude kind. Onder het jonge kind werd verstaan kinderen tot 11 jaar, onder het oude kind kinderen van 11 jaar en

ouder. Deze scheiding was gebaseerd op de klassieke theorie van Piaget die stelt dat kinderen vanaf het elfde levensjaar in de formeel operationele fase belanden (Zimbardo, Johnson & McCann, 2013). De fase bepaald welke materie het kind kan begrijpen en toepassen (Zimbardo, Johnson & McCann, 2013; Teague, 2014; Lister, 2011). Deelvragen 3 en 4 richten zich op de vraag of die cognitieve fase invloed heeft op de mate waarin een leerling een misconcept ontwikkelt. De derde deelvraag betrok dezelfde misconcepten als de eerste deelvraag en de vierde deelvraag betrok dezelfde misconcepten als de tweede deelvraag. Uit beide analyses blijkt dat er geen verschil is in het aantal misconcepten tussen het jongere en het oudere kind. De leeftijd (waarmee volgens de klassieke theorie van Piaget een samenhang met de cognitieve fase bij wordt geïnterpreteerd) heeft geen invloed op het aantal programmeermisconcepten dat een kind ontwikkelt.

5.1.3 Misconcepten en Attitude

Voor deelvraag 5 werd de samenhang tussen attitude en het aantal misconcepten onderzocht. De attitude van een leerling heeft invloed op diens leerhouding (Anghelache, 2013; Harlen & Crick, 2003; Rost, 2006). De leerhouding heeft op zijn beurt weer invloed op de wijze waarop kennis opgenomen wordt. Het feit dat de wijze waarop kennis opgenomen wordt een misconcept kan veroorzaken, gaf aanleiding tot het onderzoeken van de samenhang tussen attitude en het aantal misconcepten. Als attitude is de attitude ten opzichte van programmeren en de lessen gemeten. Met het aantal misconcepten wordt in deze deelvraag verwezen naar alle onderzochte concepten, in tegenstelling tot de eerste vier deelvragen. Er bleek geen sprake te zijn van een samenhang tussen attitude en het aantal misconcepten. Dat betekent dat een positieve attitude niet een kleiner aantal misconcepten indiceert dan een negatieve attitude. De vraag of en welke persoonlijke factoren invloed hebben op het al dan niet ontwikkelen van een misconcept, blijft met dit resultaat nog onbeantwoord.

5.1.4 Misconcepten en Ervaring

Voor deelvraag 6 werd gemeten of er een samenhang is tussen programmeerervaring en het aantal misconcepten. Bij het vragen naar programmeerervaring werd geduid op ervaring die de leerling voor de start van de eerste programmeerles hadden. Voor het aantal misconcepten werden wederom alle concepten betrokken. Tussen programmeerervaring en het aantal misconcepten blijkt een licht negatieve samenhang. Dit houdt in dat programmeurs met meer ervaring, minder misconcepten hebben. Deze theorie wordt ondersteund door Ma (2007) en Sorva (2012). Voor de praktijk zou dit kunnen inhouden dat eventuele maatregelen die getroffen worden om misconcepten te voorkomen of te verhelpen, bij de ervaren programmeurs overgeslagen kunnen worden. Er zou dan meer tijd gestoken kunnen worden in het programmeren zelf, bijvoorbeeld aan gecompliceerdere opdrachten.

Waarom ervaring negatief samenhangt met het aantal misconcepten, is nog onduidelijk. Daar zouden meerdere oorzaken aan ten grondslag kunnen liggen. Allereerst zou het kunnen zijn dat misconcepten bij ervaren programmeurs sneller verholpen worden. Het feit dat zij ervaren zijn, impliceert hoogstwaarschijnlijk dat zij gemotiveerd zijn. Juist die motivatie speelt een cruciale rol bij conceptuele verandering (Leonard, Kalinowski & Andrews, 2014; Ma, 2007). Daarnaast zou het kunnen zijn dat de kans op een cognitief conflict, wat ten grondslag ligt aan conceptuele verandering (Ma, 2007; Sorva, 2012; Leonard, Kalinowski & Andrews, 2014; Brown & Clement, 1989; Türkmen & Usta, 2007; Schoenmaker, Stomp & Valstar, 2002), meer voor de hand liggend is bij het opdoen

van meer ervaring. Doordat ervaring wordt opgedaan, is de programmeur telkens blootgesteld aan (nieuwe) (correcte) kennis. Een cognitief conflict ontstaat wanneer iemand wordt blootgesteld aan informatie die niet overeenkomt met zij huidige kennis/overtuigingen (Ma, 2007; Sorva, 2012; Leonard, Kalinowski & Andrews, 2014). Naarmate iemand met een misconcept, meer programmeerervaring opdoet, is dat kan groter dat zich een cognitief conflict voordoet. Daarnaast kan het zijn dat kinderen met meer programmeerervaring minder misconcepten ontwikkelen, doordat zij basiskennis hebben waaraan ze de nieuwe kennis kunnen koppelen. Het laatste argument hangt samen met de validiteit van het onderzoek. Meer ervaren programmeurs hebben minder misconcepten die in dit onderzoek gemeten zijn, echter zijn uiteraard niet alle soorten misconcepten in dit onderzoek gemeten. De gemeten misconcepten zijn gebaseerd op het aangeboden lesmateriaal. Het aangeboden lesmateriaal is gericht op beginnende programmeurs. Het zou niet vreemd zijn dat de behandelde materie voor de ervaren programmeurs relatief eenvoudig is en dat zij zelf met complexere materie werken. Het feit dat kinderen met meer programmeerervaring minder van de gemeten misconcepten hebben, betekent niet dat zij daadwerkelijk minder misconcepten. Zolang een kind niet deskundig is in programmeren, heeft het waarschijnlijk misconcepten van andere aard.

5.2 Discussie

Om de reikwijdte van de conclusies en de kwaliteit van de resultaten in kaart te brengen, worden de validiteit, betrouwbaarheid en bruikbaarheid van het onderzoek in kaart gebracht.

5.2.1 Betrouwbaarheid en Validiteit

De grootste bedreiging voor de validiteit van het onderzoek, is de constructvaliditeit van het meetinstrument. Of de wijze van het meten van misconcepten correct is, is niet getoetst. Er zijn wel overtuigingen die de construct validiteit in twijfel doen trekken. Met het instrument kan wel met enige zekerheid worden gezegd dat er sprake is van een misconcept, maar niet over het aantal. Een enkel concept wordt in een vraag vertaald. Naast het correcte antwoord, zijn mogelijke misconcepten naar antwoordopties vertaald en weergeven (zie bijlage 13). Per vraag kan maar een antwoordoptie gekozen worden. Als een kind een misconcept heeft, zal het een consistent het bijbehorende antwoord kiezen. Het kan niet achterhaald worden of een kind meerdere misconcepten van hetzelfde concept heeft, wanneer het maar een ‘misconcept-antwoord’ kan kiezen. Kinderen die meerdere misconcepten zouden hebben, antwoordden waarschijnlijk inconsistent op de vragen van een concept. Echter zullen gokkers dit eveneens doen, waardoor zij moeilijk te onderscheiden zijn. Daarnaast zijn enkele ‘misconcept-antwoorden’ te herleiden uit meerdere misconcepten van verschillende concepten. Dit is enigszins afhankelijk van de vraagstelling. Het zou bij enkele vragen kunnen zijn dat een de respondent een ander misconcept heeft dan met deze vraag bedoeld wordt.

Daarnaast is de externe validiteit van de resultaten over leeftijd en misconcepten betwistbaar. Het argument om jong en oud in de deelvragen te baseren op de overgang van het concreet operationele naar het formeel operationele (vanaf 11 jaar), is niet vanzelfsprekend. De keuze komt voort uit de klassieke theorie van Piaget, zonder het in acht nemen van de nieuwe theorieën. Tevens is de ontwikkeling van de fase geen abrupte overgang in het elfde levensjaar, maar een soepele overgang die rond het elfde levensjaar plaatsvindt. Doordat de cognitieve fase niet direct aan leeftijd te koppelen is, heeft de conclusie een smalle reikwijdte. Dit haalt niet weg dat de (nieuwe) theorieën van

Piaget relevant zijn om te onderzoeken in combinatie met programmeeronderwijs. Dan zou er gekeken worden naar de leerfase is het programmeren, die samenhangt met ervaring.

Tot slot een blik op de statistische validiteit van het onderzoek, deze heeft betrekken op de variabele ‘ervaring’. Het enige gemeten feit is of er sprake is van eerdere programmeerervaring. De mate van programmeerervaring blijft hierbij onbekend. Wellicht zijn er respondenten die eenmalig tien minuten iets met programmeren hebben uitgeprobeerd en hebben zij dit als eerdere programmeerervaring waargenomen. Zij worden in de dataset mogelijk onterecht gelijkgesteld aan respondenten met omvangrijke programmeerervaring. Voor vervolgonderzoek wordt aangeraden meer door te vragen naar de aard van programmeerervaring.

Er zijn geen grote fouten gemaakt die directe gevolgen hebben voor de betrouwbaarheid van het onderzoek, echter was er in dit onderzoek veel ruimte voor onregelmatigheden. Doordat de lessen en experimenten in het eigen klaslokaal van de leerlingen plaatsvonden, was er geen sprake van een gewaarborgde omgeving. De aangeboden stof was afhankelijk van het verloop van de les. Wanneer leerlingen bijvoorbeeld erg rumoerig waren of veel vragen hadden, was er soms niet genoeg tijd om alle stof te behandelen. Er kan dus niet gewaarborgd worden dat iedere leerling precies dezelfde les heeft gevolgd. De betrouwbaarheid van het onderzoek wordt hierdoor echter niet in de ban gedaan. Wanneer deze resultaten van dit onderzoek eventueel gebruikt zouden worden voor het programmeeronderwijs, zou er van dezelfde setting sprake zijn.

5.2.2 Bruikbaarheid

Ondanks dat dit onderzoek niet veel harde bewijzen opleverde, zijn de resultaten heel goed bruikbaar. Enerzijds zijn de resultaten instrumenteel goed inzetbaar. Het meetinstrument vormt een goede opzet voor meetinstrumenten de in vervolgonderzoeken naar misconcepten ingezet zullen. Het vergt nog aanpassing, maar kan in de huidige staat tevens gebruikt worden voor het meten van programmeervaardigheden in Scratch. Daarnaast zijn voor het onderzoek de huidige programmeerlessen aangepast, enkele oude inhoud bleek op basis van ander onderzoek incorrect te zijn. Daarnaast heeft het literatuuronderzoek nieuwe bronnen en namen van relevante wetenschappers opgeleverd, die geraadpleegd zouden kunnen worden bij vervolgonderzoek naar misconcepten. Tot slot is er een database ontwikkeld. Hieruit kunnen door middel van statische analyses nog resultaten geformuleerd worden die in dit onderzoek niet relevant zijn, maar wellicht wel informatie over misconcept kunnen opleveren.

Het onderzoek heeft eveneens een hoge conceptuele bruikbaarheid. De opdrachtgever zal onderzoek naar misconcepten en andere aspecten van programmeeronderwijs doorzetten. De resultaten van het onderzoek leveren een variatie aan nieuwe onderzoeksvragen op. Bij de aanbevelingen wordt hier dieper op ingegaan.

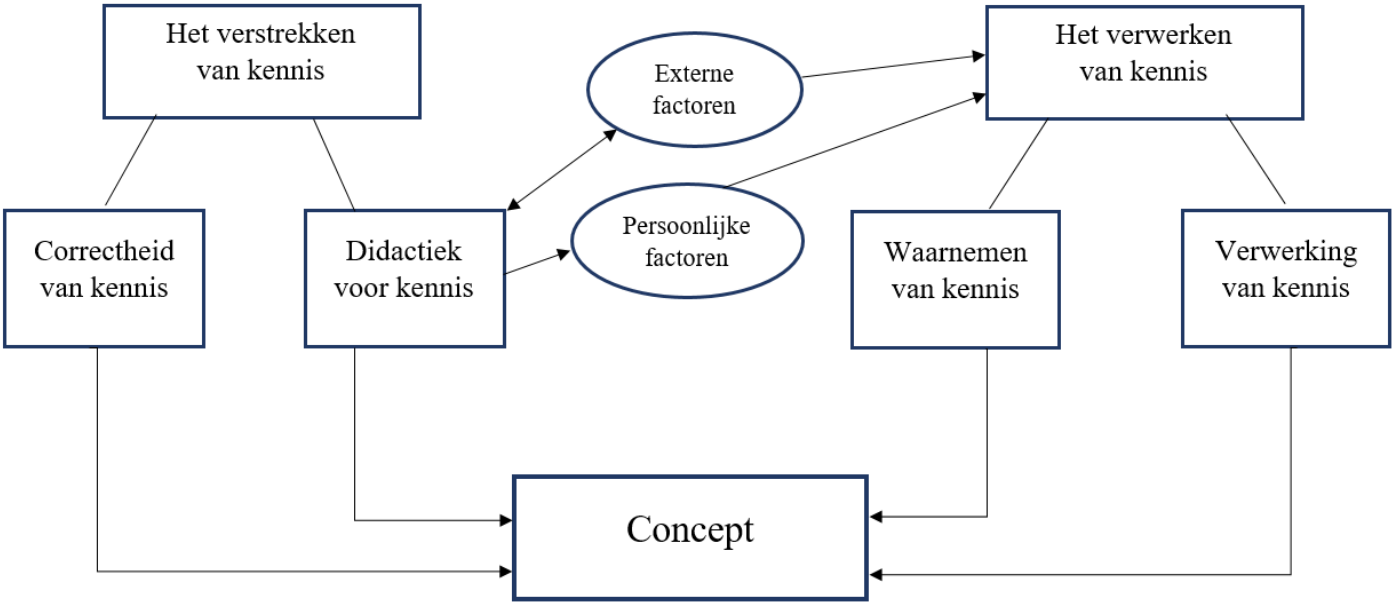
5.3 Aanbevelingen

In de inleiding van het onderzoek staat beschreven dat de opdrachtgever actief is met het ontwikkelen en onderzoeken van wetenschappelijk onderbouwde kennis om daar, onder andere, het programmeeronderwijs mee te kunnen verbeteren. Met die visie voeren zij verscheidene onderzoeken uit, waar dit onderzoek deel van uitmaakt. Door telkens onderzoeksvragen te beproeven die

voortvloeien uit andere onderzoeken, kan hier steeds verder naartoe gewerkt worden. De aanbevelingen zijn gericht op zowel vervolgonderzoek als op concrete aandachtspunten voor de praktijk van programmeeronderwijs.

5.3.1 Vervolgonderzoek

Er is een model ontwikkeld waarin de samenhang van de aspecten bij het ontwikkelen van een concept weergegeven is, weergegeven in figuur 2. Het merendeel van de aanbevelingen voor vervolgonderzoek is hierop gebaseerd. Enkele onderdelen van het model zijn gebaseerd op de beschreven literatuur in het theoretisch kader, andere onderdelen zijn gebaseerd op de resultaten van het onderzoek.



Figuur 2, *Model ontwikkeling concepten*

Alle weergegeven variabelen spelen een rol in het proces van de ontwikkeling van een concept. De onderdelen ‘Het verstrekken van kennis’ en ‘Het verwerken van kennis’ zijn opgedeeld in vier concrete onderdelen, welke direct van invloed zijn op het al dan niet ontwikkelen van een misconcept. Als deze vier onderdelen juist zijn of juist verlopen en de ontwikkeling van het concept wordt afgerond, ontstaat er een valide concept zonder misconcepten. Wanneer bij een of meer van deze onderdelen een fout gemaakt wordt en de ontwikkeling van het concept doorgezet wordt, ontstaat er een misconcept. Daarnaast zijn er factoren die indirect invloed hebben op de ontwikkeling van een concept. Bij persoonlijke factoren kan gedacht worden aan fitheid, intelligentie, motivatie en attitude. Bij externe factoren kan gedacht worden aan de leeromgeving (lawaai), het tijdstip en de docent.

Dit model zou efficiënt gebruikt kunnen worden als uitgangspunt voor vervolgonderzoeken. Doordat de beschreven aspecten allemaal een rol spelen in de ontwikkeling van een concept, zijn ze relevant om nader te onderzoeken. Op het aspect ‘didactiek voor kennis’ is in dit onderzoek al een beroep gedaan, deelvragen 1 en 2 hadden hier betrekking op. De experimenten deden een beroep op de didactiek. De gekozen didactiek had geen invloed op het al dan niet ontwikkelen van een misconcepten. Dit zou kunnen komen doordat de gekozen didactiek niet gepast is, maar ook doordat didactiek wellicht geen doorslaggevende rol speelt. Door de samenhang van ‘didactiek’ voor de

kennis met de ontwikkeling van een concept te onderzoeken, kan hier duidelijkheid over geschept worden. Daarnaast kan onderzocht worden wat de beste methoden en technieken zijn voor didactiek in programmeeronderwijs. Deze twee uitgangspunten voor onderzoek gelden overigens ook voor de andere aspecten.

Daarnaast is er met deelvragen 3 tot en met 6 een beroep gedaan op verschillende onderdelen van het aspect 'persoonlijke factoren'. Naar aanleiding van de resultaten op deelvragen 3 en 4 kan de samenhang tussen cognitieve fasen en het ontwikkelen van misconcepten nader onderzocht worden. Vanuit de literatuur zijn er sterke aanwijzingen dat de cognitieve fase samenhangt met de ontwikkeling van misconcepten. Bijvoorbeeld dat de aangeboden kennis moet aansluiten op de belevingswereld van het kind, welke afhankelijk is van diens cognitieve fase. In het theoretisch kader staat op basis van Piagets nieuwe theorieën beknopt beschreven welke programmeervaardigheden per cognitieve fase uitgevoerd kunnen worden. Die verschillen uiteraard aanzienlijk per fase, wat aanleiding geeft tot het onderzoeken van welke misconcepten kinderen in de verschillende cognitieve fasen hebben. Hierdoor zou programmeeronderwijs beter afgestemd kunnen worden op de leerling. Voor vervolgonderzoek wordt er aangeraden onderzoeksgroepen niet meer te baseren op leeftijd, maar op leerjaar. Van leerlingen wordt geacht dat zij competent genoeg zijn voor het leerjaar waarin zij geplaatst zijn. Doordat er de mogelijkheid is om leerlingen een leerjaar over te laten slaan of opnieuw te laten doen, is leeftijd minder bepalend.

Programmeerervaring blijkt invloed te hebben op het al dan niet ontwikkelen van een misconcept. Onderzoek naar de invloed programmeerervaring op misconcepten zou bijvoorbeeld twee vormen kunnen aannemen. Enerzijds hoe programmeerervaring zo veel mogelijk gestimuleerd kan worden zodat er sprake is van minder misconcepten. Anderzijds op welke manier programmeerervaring invloed heeft op het aantal misconcepten. Zijn programmeurs met meer ervaring bijvoorbeeld meer gemotiveerd tot leren? Of doet programmeerervaring misconcepten vanzelf oplossen, doordat het samenhangt met het constructivisme? Wat betreft persoonlijke en externe factoren in zijn algemeen, kan onderzocht worden uit welke onderdelen deze bestaan en welke invloed deze hebben op de ontwikkeling van een concept.

In het bijzonder zou de belevingswereld van de leerling relevant zijn om te onderzoeken, als onderdeel van persoonlijke factoren. Zoals omschreven, hebben kinderen bij het beginnen van programmeren nog geen of weinig schema's of basiskennis waar zij op terug kunnen vallen. Dat terwijl deze wel belangrijk zijn bij het leerproces. Dus hoe ziet de belevingswereld van programmeren van een kind eruit en hoe kan programmeeronderwijs daarop aansluiten? Programmeerconcepten zijn veelal abstract maar concrete basiskennis kan een goede vormen. Een sterk verwant onderwerp is de computer, welke heel concreet is. In hoeverre kan de belevingswereld van programmeren van een kind gevormd worden door kennis van de computer?

Een belangrijk onderwerp voor vervolgonderzoek is het meten van misconcepten. In de literatuur zijn enkele mogelijkheden beschreven, maar wat in de praktijk het beste werkt moet nog blijken. Dat is uiteraard tevens afhankelijk het onderzoek. De ontwikkelde methode voor dit onderzoek is geschikt voor een globale meting. Er moet rekening worden gehouden met het feit dat de respondent per vraag

maar een van de drie mogelijke misconcepten kan toepassen. De inconsistente groep in dit onderzoek is groot, de samenstelling van deze groep kan interessant zijn om te onderzoeken. Een ander aandachts- en verbeterpunt voor het instrument is dat sommige antwoordopties uit verscheidene misconcepten voort kunnen komen. Een goede methode voor een gedetailleerde meting van misconcepten moet nog onderzocht worden. Een optie is dat er wederom vier vragen over een concept weergegeven worden, met het correcte antwoord, een misconcept en twee willekeurige antwoorden als antwoordopties. Wanneer misconcepten geïsoleerd gemeten worden kunnen er meer misconcepten achterhaald worden. Doordat een respondent met meerdere misconcepten dan waarschijnlijk consistent het misconcept-antwoord kiest, zal de inconsistente groep kleiner worden.

Tot slot zijn voor dit onderzoek nieuwe misconcepten beschreven. Voor dit onderzoek is niet geanalyseerd of er van deze misconcepten daadwerkelijk sprake was bij de doelgroep, maar dit kan met de ontwikkelde dataset alsnog gedaan worden. De tot nu onderzochte misconcepten zijn allemaal opgesteld bij onderzoek naar programmeren bij volwassen. Een misconcepten is echter afhankelijk van de belevingswereld, welke bij kinderen heel anders is. Als er nieuwe misconcepten ontdekt worden die specifiek op kinderen van toepassing zijn, kunnen deze meegenomen worden in vervolgonderzoek. Het aantal nieuw opgestelde misconcepten in dit onderzoek is relatief klein, omdat de behandelde onderwerpen in de lessen als grens dienden bij het opstellen van misconcepten. Bij ander lesmateriaal zijn er natuurlijk andere misconcepten mogelijk met wederom nieuwe mogelijkheden voor vervolgonderzoek.

5.3.2 Programmeeronderwijs

Omdat dit meer een explorerend theoretisch onderzoek is dan een onderzoek gericht op de praktijk, zijn er op basis van dit onderzoek minder aanbevelingen voor de praktijk geformuleerd dan voor vervolgonderzoek. De praktische aanbeveling die wordt gedaan, is om de theorieën van Piaget als uitgangspunt te nemen bij het kiezen van de aard en de complexiteit van het materiaal. Er kan het beste een combinatie gebruikt worden van de klassieke en nieuwe theorieën. De nieuwe theorieën beschrijven dat een kind per onderwerp door ervaring de vier fasen van cognitieve ontwikkeling doorloopt, volgens de klassieke theorieën gebeurt dit op basis van biologische ontwikkelingen van het brein (en samenhangt met leeftijd). Het ontwerp van een leerlijn in programmeeronderwijs zou zich kunnen richten op de nieuwe theorie van Piaget, zodat een leerling de fasen doorloopt bij het leren programmeren. Uiteraard kan een leerling uit groep 4 daarbij niet eerlijk gelijk worden gesteld aan een leerling uit groep 8. De klassieke Theorie van Piaget kan onderliggen aan de keuze van niveaus waarop het onderwijs wordt aangeboden. In bijlage 15 is een concreet plan opgesteld.

5.3.3 De Toegepast Psycholoog

Het terrein van programmeeronderwijs heeft nog vele ontwikkelingen te ondergaan, waarbij de (programmeer-)enthousiaste toegepast psycholoog een rol kan spelen. Bij vervolgonderzoek zou een toegepast psycholoog een waardevolle rol aan kunnen nemen als adviseur met kennis van ontwikkelings- en cognitieve psychologie. Er zou dan samengewerkt worden met programmeurs en onderwijsdeskundigen om relevante en waardevolle onderzoeken naar programmeeronderwijs op te zetten en uit te voeren. Daarnaast kan de toegepast psycholoog zelfstandig onderzoeken opzetten en uitvoeren met betrekking tot de persoonlijke factoren, zoals om weergegeven in het ontwikkelde

model. Er kan achterhaald worden van welke persoonlijke factoren daadwerkelijk sprake is en de invloed die zij uitoefenen.

In de praktijk van het programmeeronderwijs kan de toegepast psycholoog ook zijn intrede doen. Het programmeeronderwijs moet zo goed mogelijk aansluiten bij de belevingswereld van de leerling om efficiënt te zijn. Daarvoor kan kennis van ontwikkelings- en cognitieve psychologie ingezet worden. Verder in de toekomst, op basis van het vervolgonderzoek, kan de toegepast psycholoog een rol spelen bij het beïnvloeden van de persoonlijke factoren uit het model. Denk daarbij aan het verbeteren van attitude en motivatie bij leerling.

Literatuur

Anderson, J. R. (2010). *Cognitive Psychology and Its Implications*. New York: Worth Publishers

Anghelache, V. (2013). Determinant factors of student's attitudes towards learning. *Social and Behavioral Sciences, 93,* 478-482

Baarda, B., Van Dijkum, C. & De Goede, M. (2014). *Basisboek Statistiek me SPSS.* Nederland, Groningen/Houten: Noordhoff Uitgevers

Bednar, A. K., Cunningham, D., Duffy, T. M. & Perry, J. D. (1991). Theory into practice: How do we link? *Instructional Technology: Past, Present and Future.* Englewood: Libraries Unlimited, Inc.

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computer in Mathematics and Science Teaching, 20*(1), 45-73

Brown, D. E. & Clement, J. (1989). Overcoming misconceptions via analogical reasoning: abstract transfer versus explanatory model construction. *Instructional Science, 18*(4), 237-261.

Castejón, J. L., Gilar, R. & Pérez, A. M. (2006). Complex learning: The role of knowledge, intelligence, motivation and learning strategies. *Psicothema, 18*(4), 679-685

Clements, D. H. & Gullo, D. F. (1994). Effects of Computer Programming on Young Children's Cognition. Journal of Educational Psychology. Vol. 76, No. 6, 1051-1058.

Connectis. (2017a). Programmeerles voor alle basisschoolkinderen in Rotterdam. Verkregen van: https://connectis.nl/nl/nieuws/persbericht-programmeerles-voor-alle-basisschoolkinderen-in-rotterdam/

Connectis. (2017b, oktober). Al 900 Rotterdamse basisschoolleerlingen leerden programmeren. Verkregen van: https://connectis.nl/nl/nieuws/al-900-rotterdamse-basisschoolleerlingen leerden-programmeren/

Connectis. (zj). Informatie voor kinderen, ouders, scholen en studenten. Verkregen van: https://connectis.nl/nl/les/

De Ingenieur. (2014). Leer Jeugd Programmeren. Verkregen van: https://www.deingenieur.nl /artikel/leer-jeugd-programmeren

Dehnadi, S. & Bornat, R. (2006). *The Camel has Two Humps*. Middlesex University Working Paper

FutureNL. (zj). Je kunt je afvragen of je kind meer gebaat is bij de Franse taal of bij een programmeertaal?. Verkregen van: http://futurenl.org/waarom-digitale-vaardigheden/

Gerrichhauzen, R. (2010). 'Motivatie om te leren!'. Masterthesis, Fontys Hogescholen

Harlen, W. & Crick, R. D. (2003). Testing and Motivation for Learning. *Assessment in Education, 10*(2), 169-207. DOI: 10.1080/0969594032000121270

Hewson, P. W. & Hewson, M. G. (1984). The role of conceptual conflict in conceptual change and the design of science instruction. *Instructional Science, 13*, 1-13

Holland, S., Griffiths, R. & Woodman, M. (1997). Avoiding Object Misconceptions. *ACM SIGCSE Bulletin, 29*(1), 131-134. DOI: 10.1145/268084.268132

Hollingsworth, J. R. & Ybarra, S. E. (2017). *Expliciete Directe Instructie: Tips en technieken voor een goede les*. Nederland, Huizen: Uitgeverij Pica.

Hromkovic, J. (2006). Contributing to general education by teaching informatics. *Informatics Education, 4226*, 25-37. DOI: 10.1007/11915355_3

Jeuring, J., Corbalan, G., Van Es, N. & Leeuwestein, H. (2016). Leren programmeren in het PO. Universiteit Utrecht

Kallia, M. & Sentance, S. (2017). Computing Teachers' Perspectives on Threshold Concepts: Functions and Procedural Abstraction. *Proceedings of WiPSCE '17, Nijmegen, Netherlands, November 8-10, 2017*, p. 15- 24

Kennisnet. (2017). Werken aan digitale geletterdheid: van visie naar praktijk. Verkregen van: https://www.kennisnet.nl/publicaties/werken-aan-digitale-geletterdheid-van-visie-naar-praktijk/

Kennisnet. (2016). Programmeeronderwijs is toe aan de volgende stap. Verkregen van: https://www.kennisnet.nl/artikel/programmeeronderwijs-is-toe-aan-volgende-stap/

Leonard, M. J., Kalinowski, S. T. & Andrews, T. C. (2014). Misconceptions Yesterday, Today, and Tomorrow. *CBE Life Sciences Education, 13*(2), 179-186

Lister, R. (2011). Concrete and other Neo-Piagetian Forms of Reasoning in the Novice Programmer. *Conferences in Research and Practice in Information Technology, 114*, 9-18

Ma, L. (2007). Investigating and Improving Novice Programmers' Mental Models of Programming Concepts. University of Strathclyde

Mediawijsheid. (2017). De digitale generatie heeft nog veel hulp nodig. Verkregen van: https://www.mediawijzer.net/vanzelf-mediawijs-2017/

Morra, S., Gobbo, C., Marini, Z. & Sheese, R. (2008). *Cognitive Development in Neo-Piagetian Perspectives.* New York: Lawrence Erlbaum Associates. DOI: 10.13140/2.1.4196.6722

NOS. (2014). Tech-bedrijven willen iedereen leren programmeren. Verkregen van: https://nos.nl/artikel/2007893-tech-bedrijven-willen-iedereen-leren-programmeren.html

NRC. (2017). Leren programmeren met een gratis computer. Verkregen van: https://www.nrc.nl/nieuws/2017/10/17/leren-programmeren-met-een-gratis-computer 13516957-a1577472

Papert, S. (1980). *Mindstorms. Children, Computers and Powerful Ideas*. New York: Basic Books, Inc. Publishers.

Piaget, J. (1977). *The development of thought: Equilibration of cognitive structures*. New York: Viking Penguin

Proulx, J. (2006). Constructivism: A re-euilibration and clarification of the concepts and some potential implications for teaching and pedagogy. *Radical Pedagogy, 8*(1)

Rehman, A. & Haider, K. (2013). The Impact of Motivation on Learning of Secondary School Students in Karach: And Analytical Study. *Educational Research International, 2*(2), 139-147

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Kafai, Y. (2009). Scratch: Programming for All. *Coomunications of the ACM, 52*(11), 60-67. DOI: 10.1145/1592761.1592779

Rost, M. (2006). Generating Student Motivation. *WorldView*. New York: Pearson Education Inc.

Saeli, M., Perrenet, J., Jochems, W. M. G. & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education, 10*(1), 73-88.

Schunk, D. H. & Zimmerman, B. J. (2008). *Motivation and Self-Regulated Learning: Theory, Research, and Applications*. New York: Lawrence Erlbaum Associates

Schoenmaker, K., Stomp, L. & Valstar, J. (2002). Van vakdidactiek naar een gedeelde opleidingsdidactiek: Het geheel van de opleidingsdidactiek is meer dan de som van de vakdidactische delen. *VELON Tijdschrift voor Lerarenopleiders, 23*(2), 14-22.

SLO. (Februari 2016). ICT basisvaardigheden. Verkregen van: http://curriculumvandetoekomst.slo.nl/21e-eeuwse-vaardigheden/digitale-geletterdheid/ict-basisvaardigheden

Sorva, J. (2012). *Visual Program Simulation in Introductory Programming Education*. Espoo: School of Science. ISBN: 978-952-60-4626-6

Stavy, R. (1991). Using analogy to overcome misconceptions about conservation matter. *JRST, 28*(4), 305-313

Szlávi, P. & Zsakó, L. (2006). Programming versus Application. *Informatics Education, 4266,* 48-58. DOI: 10.1007/11915355_5

Teague, D. (2014). Neo-Piagetian Theory and the Novice Programmer. University of Sussex. Verkregen van: http://ppig.org/biblio?f[author]=645

Tew, A. E. (2010). *Assessing Fundamental Introductory Computing Concept Knowdledge in a Language Independent Manner.* Georgia Institute of Technology, School of Interactive Computing

Türkmen, H. & Usta, E. (2007). The role of learning cycle approach in overcoming misconceptions in science. *Kastamonu Egitim Dergisi, 15*(2), 491-500.

Visser, J. (2017, oktober). Hoe bedrijven kinderen aan het programmeren krijgen. Verkregen van: https://decorrespondent.nl/7542/hoe-bedrijven-kinderen-aan-het-programmeren krijgen/599234526-4439fa00

Vos, H. J. (2009). *Social Research Methods*. Engeland, Essex: Pearson Custom Publishing

Vrasidas, C. (2000). Constructivism versus objectivism: Implication for interaction, course design and evaluation in distance education. *International Journal of Educational Telecommunications, 6*(4), 339-362

Wing, J. M. (2008). Computational thinking and thinking about computing. *Phil. Trans. R. Soc. A, 366,* 3717-3725. DOI: 10.1098/rsta.2008.0118

Zander, C., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M. & Sanders, K. (2007). *Threshold concepts within the Disciplines,* 105-118. Rotterdam, Nederland: Sense Publishers

Zimbardo, P. G., Johnson, R. L. & McCann, V. (2013). *Pyschologie Een Inleiding*. Amsterdam, Nederland: Pearson Benelux b.v.

Bijlage 1 Eigen werkverklaring

Kom verder SAXION

Eigen werkverklaring

Ondergetekende: (Naam student)

Emily C.A. Bakker

verklaart ondubbelzinnig dat:

1) dit werkstuk eigen werk is en daarom geen inbreuk maakt op het auteursrecht van een ander,

2) alle gebruikte bronnen (waaronder internetpagina's) zijn voorzien van bronvermelding,

3) het verslag voor niet meer dan 5 % aan overgenomen passages uit ' werk van anderen' bevat.

4) dit verslag ook digitaal is ingeleverd via Safe Assign (Blackboard).

Plaats: Hengelo

Datum: 4 juni 2018

Handtekening:

N.B. Schending van bovengenoemde 'Eigen werkverklaring ' wordt als fraude aangemerkt als bedoeld in Art. 19 OER.

saxion.nl

Bijlage 2 Lijst misconcepten Sorva (2012)

1. The computer knows the intention of the program or of a piece of code, and acts accordingly.
2. The computer is able to deduce the intention of the programmer.
3. Values are updated automatically according to a logical context.
4. The system does not allow unreasonable operations.
5. Difficulties understanding the lifetime of values.
6. Difficultieswithtellingapartthestaticanddynamicaspects of programs.
7. The machine understands English.
8. Magical parallelism: several lines of a (simple nonconcurrent) program can be simultanenously active or known.
9. A variable can hold multiple values at a time / 'remembers' old values.
10. Variables always receive a particular default value upon creation.
11. Primitive assignment works in opposite direction.
12. Primitive assignment works both directions (swaps).
13. Limited understanding of expressions which lacks the concept of evaluation.
14. A variable is (merely) a pairing of a name to a changeable value (with a type). It is not stored inside the computer.
15. Primitive assignment stores equations or unresolved expressions.
16. Assignment moves a value from a variable to another.
17. The natural-language semantics of variable names affects which value gets assigned to which variable.
18. The order of declaration of variable names affects which value gets assigned to which variable.
19. There is no size or precision limit to the values we can store in a programming variable.
20. Incrementing a counter variable is an indivisible operation (no separate evaluation of right-hand side).
21. Primitive types (in Java) have no default value.
22. Unassigned variables of primitive type (in Java) have no memory allocated.
23. Difficulties in understanding the sequentiality of statements.
24. Code after statement is not executed if the _____ _____clause is.
25. Statement gets executed as soon as its condition becomes true.
26. A false condition ends program if no branch.
27. Both _____ _____ and branches are executed.
28. The _____ _____ branch is always executed.
29. Adjacent code executes within loop.
30. Control goes back to start when condition is false.
31. Difficulty in understanding automated changes to _____ loop control variables.
32. Loops terminate as soon as condition changes to false.
33. Loop control variables do not have values inside the loop or their values can be arbitrarily changed.
34. Print statements are always executed, irrespective of branching statements.
35. All statements of a program get executed at least once.
36. Subprogram code is executed according to the order in which the subprograms are defined.
37. A return values does not need to be stored (even if one needs it later).

38. A method can be invoked only once.
39. Numbers or numeric constants are the only appropriate actual parameters corresponding to integer formal parameters.
40. Difficulties distinguishing between actual and formal parameters. Confusion over where parameter values come from.
41. Difficulties understanding the invocation of a method from another method.
42. Confusion over where return values go.
43. Parameter passing forms direct name-based procedure-toprocedure links between variables with the same name (in call and signature).
44. Parameter passing forms direct procedure-to-procedure links between variables with different names (in call and signature).
45. Parameter passing requires different variable names in call and signature.
46. Subprograms can (routinely) use the variables of calling subprograms.
47. Cannotuseglobalsinsubprogramswhentheyhavenotbeen passed as parameters.
48. When the value of a global variable is changed in a procedure, the new value will not be available to the main program unless it is explicitly passed to it.
49. A function (always) changes its input variable to become the output.
50. Expressions (not their values) are passed as parameters.
51. Argument expressions cause changes in existing variables.
52. Upon return, the value of a variable changes to correspond to match a previously given parameter.
53. Null model of recursion: recursion is impossible.
54. Active model: only the 'active aspect' of recursion understood, not return values.
55. Step model: single- or two-step recursion, but no more.
56. Return value model: each instantiation first produces a value, before the next one is started; then all values are combined to get the result
57. Passive model: active part not understood, only the combinatorics of the return values.
58. Recursion is merely a programming construct used as a template in certain kinds of programs; runtime behavior is 'magic'.
59. Recursion is perceived as an algebra problem.
60. Looping model of recursion (single-instantiation); recursion is merely a construct for producing repetition, not understood as self-reference.
61. Even primitive values (in Java) are handled through references.
62. The variables for storing and assigning primitive values are fundamentally different from the variables used for storing objects (in Java).
63. Avariableassociatedwithanobjectismerelyanamewhich canbeusedtomanipulateanobjector'thing'inaprogram.
64. A (non-primitive Java) variable does not hold a reference but a set of object properties.
65. Assigning an object means 'sending' an object (or a copy) to a variable.
66. Assigning to an object causes it to become equal to the assigned object.
67. Assigningtoanobjectmeansthat(some)instancevariables get new values from the assigned object.
68. Confusionbetweenaninstancevariablesuchas"name"and a variable referring to the object.
69. Two objects with the same value for a "name" attribute are the same object.
70. A value of an instance variable such as "name" replaces reference as a value of an attribute.
71. An object is represented (inprogramstate)by only the value of a particular identifying instance variable such as "name".
72. A variable that refers to an object uniquely specifies the object for all time.
73. Onceavariablereferencesanobject, itwillalwaysreference that object.
74. Two different variables must refer to two different objects.
75. Two objects of the same class with the same state are the same object.
76. Two objects can have the same identifier if there is any difference in the values of their attributes.
77. Objects know what refers to them. References point in opposite direction (to referring object attribute, not from there).
78. Confusion between a class and its instance.

79. An object is a subset of a class. / A class is a collection of objects.
80. An object is a subtype of a class.
81. A set (such as “team” or “the species of birds”) cannot be a class.
82. Constructors can include only assignment statements to initialize attributes.
83. Instantiation involves only the execution of the constructor method body, not the allocation of memory.
84. Difficulties understanding the empty constructor.
85. Initializing an attribute with a constant as part of its declaration causes confusion in distinguishing between a class and an object.
86. Initializing an attribute with a constant within the constructor declaration causes confusion in distinguishing between a class and an object.
87. Invocation of the constructor method can replace its definition.
88. Difficulties in understanding where objects of a simple class are created before the creation of the object of a composed class.
89. Difficulties understanding objects if their attributes are not explicitly initialized.
90. Objects are created by themselves, without explicit instructions to create them.
91. Declaring a variable also creates an object.
92. You can define a non-constructor (Java) method to create a new object.
93. A textual representation of an object is a reference to the object.
94. There is no need to invoke the construct or method, because its definition is sufficient for object creation.
95. If objects of a simple class already exist, there is no need to create the object of a composed class that builds on them.
96. If the object attributes are initialized in the classdeclaration, there is no need to create objects.
97. Thecreationofanobjectofacomposedclassautomatically creates objects of the simple class that appear as attributes of the composed class.
98. During a method call, an object attribute is duplicated as a local variable. Assignments update both.
99. During a method call, an object attribute is duplicated as a local variable. The local variable is initialized from the object, updated by the method, then returned to object at end.
100. During a method call, an object attributes is duplicated as a local variable, which is initialized to a default value (e.g., zero). Assignments only affect the local variable, not the object.
101. Parameters belong to the called object.
102. Local variables belong to the called object.
103. A main method’s local variables belong to the called object.
104. An object can only hold instance variables of a single type.
105. No object attributes within objects; they only exist locally during method calls.
106. An object cannot be the value of an attribute.
107. Attributes of composed classes include object attributes from the simple classes instead of the objects.
108. Attributes of composed classes include object attributes from the simple classes in addition to the objects.
109. An object is a wrapper for a single variable. The object is equated with the variable.
110. Attributes in a simple class are automatically replicated in the composed class by transferring its meaning.
111. To change the value of an attribute of an object of a simple class that is the value of an attribute in an object of a composed class, you need to construct a new object.
112. Difficulty grasping what the properties are that represent an object’s state.
113. The name of the variable that the object was most recently assigned to is a part of an object’s state.
114. Storing an object in memory means storing a copy of the program code of the object’s class.
115. Storing an object means storing the constructor parameters given upon object creation. These parameter values unambiguously define the object. (Methods do not affect state.).
116. You can define a (Java) method that adds an attribute to the class.

117. 118 ObjState Objects of a simple class, used as values of the attributes of a composed class, have to be identical.
118. In a composed (Java) class, you can develop a method that adds an attribute of a simple class to the composed class.
119. In a composed (Java) class, you can develop a method that removes an attribute of a simple class from the composed class.
120. Objects of the same class cannot have equal attribute values.
121. Attributes of the simple class must be directly accessed from the composed class instead of through an interface.
122. Objects are allocated the same amount of memory regardless of definition and instantiation.
123. Methods Objects 'know' which methods are operating on them (rather than method calls 'knowing' which object they operate on).
124. Cannot have methods with the same name in different classes.
125. The dot operator can only be applied to methods.
126. You can define a method that replaces the object itself.
127. You can define a method that destroys the object itself.
128. You can define a method that divides the object into two different objects.
129. Methods can only do assignment.
130. Methods from the simple class are not used; instead, new equivalent methods are defined and duplicated in the composed class.
131. You can invoke a method on an object only once.
132. Difficulties in understanding that a method can be invoked on any object of the class.
133. Methods can only be invoked on objects of the composed class, not on objects of the simple class defined as values in its attributes.
134. After a composed class is defined, new methods cannot be defined in the simple class.
135. A (Java) method must always be invoked on an explicit object.
136. Static and dynamic existence of methods mixed up.
137. Methods that are declared in a simple class have to be declared again in a composed class for each of the simple objects.
138. There is no need for mutators and accessors for attributes that are of a simple class within a composed class.
139. Object lookup involves searching through each object in memory for a suitable ID.
140. Objects are stored in folders on the hard drive and lookup means searching through them.
141. An object is just a piece of code (and not a dynamic actor at runtime).
142. An object is just a record.
143. An object is a 'way of working' consisting of action: expressions, return expressions etc. It has a control flow. No concept of state.
144. Difficulties in understanding how one class recognizes another.
145. Difficulties in understanding inheriting functionality (methods).
146. Difficulties in understanding how the computer knows what class attributes and methods are.
147. Assignment confused with subclassing.
148. Inheritance hierarchies express the parts of composite classes.
149. Difficulties understanding the effect of input function calls on execution.
150. Confusion between an array and its cell.
151. Difficulties with dealing with 2D array subscripts and dimensions.
152. Difficulties with arrays containing indices as data.
153. Values of conditional expressions get printed out.
154. Numbers are just numbers. (Why have ——————————— and ——————————— separately?)
155. A type is a set of constraints on values.
156. Types can change on the fly (in Java).
157. Confusion between data in memory and data on screen.

158. The computer keeps what has been printed in memory (as part of state?).
159. Confusing textual representations with each other, e.g., the string “456” with the number.
160. Boolean values are just something used in conditionals and not data comparable to numbers or strings.
161. A loop control variable constrains the values that can be input within the loop.

Bijlage 3 Toestemmingformulier

Het toestemmingformulier dat leerlingen door hun ouders moesten laten ondertekenen om mee te mogen doen aan de vragenlijst. Beide scholen plaatsten bij deze brief een formulier met uitleg vanuit het bestuur en goedkeuring voor het onderzoek.

Programmeeronderwijs voor kinderen

Inleiding
We willen uw kind vragen om mee te doen aan een onderzoek naar programmeerles.

De keuze om uw kind te laten deelnemen is aan u. In deze studie willen we meten wat het effect is van misconcepten op programmeervaardigheden.

Gedurende de lessen
Er komt een gastdocent in de klas die programmeerlessen geeft. Aan het eind van de vier lessen krijgen leerlingen programmeeropdrachten die meten of ze de concepten goed begrepen hebben. De opdrachten nemen meestal ongeveer een kwartier tijd in beslag aan het einde van de les.

Voordelen
Door het meedoen aan de studie dragen kinderen bij aan ons begrip over programmeerlessen, maar het is niet gegarandeerd dat individuele kinderen er profijt van hebben.

Vertrouwelijkheid
Als de data van het onderzoek wordt gepubliceerd, worden namen geanonimiseerd. We doen ons uiterste best om alle resultaten inclusief de scores van kinderen vertrouwelijk te behandelen.

Beloning
Kinderen ontvangen geen beloning voor deelname.

Rechten van deelnemers
Deelnemers doen vrijwillig mee en kunnen op ieder moment stoppen met de opdrachten. Kinderen kunnen bij de gastdocent aangeven dat ze hun data niet meer beschikbaar stellen. Hun data wordt dan uit het onderzoek verwijderd.

Contactinformatie
Bij vragen of problemen, kunt u zich altijd wenden tot verantwoordelijke wetenschapper van de TU Delft: Felienne Hermans: f.f.j.hermans@tudelft.nl of 015 278 7750.

Toestemming
Ik verklaar op een voor mij duidelijke wijze te zijn ingelicht over de aard, methode en doel van het onderzoek. Ik weet dat de gegevens en resultaten van het onderzoek alleen anoniem en vertrouwelijk aan derden bekend gemaakt zullen worden. Mijn vragen zijn naar tevredenheid beantwoord.

Als ouder of verzorger van_____ (naam kind) stem ik geheel vrijwillig in met deelname aan dit onderzoek.

Geboortedatum kind: _____ Klas:_____

Datum: _____ Handtekening: _____

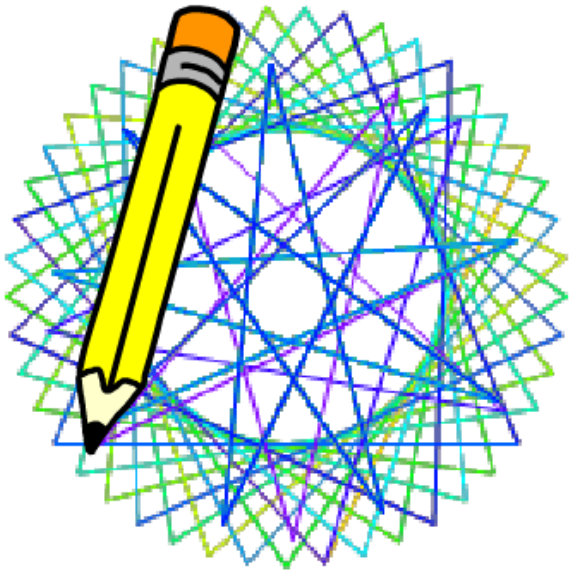
Bijlage 4 Les 1

* De pagina lay-out wijkt af i.v.m. ruimtebesparing.

Scratch les 3: Spirograaf

Een kunstwerk maken

Met programmeren kun je alles maken! Ook een kunstwerk!
In deze les maken we zelf een kunstwerk met Scratch.
En je leert ook over herhalingen en over variabelen. Als je nog niet weer wat dat zijn, geen zorgen! Dat gaan we samen leren.



Zo gaat jouw kunstwerk er straks uitzien. Of misschien wel heel anders.

Ga naar deze website: https://scratch.mit.edu/projects/160386928/#editor

Dit materiaal is gemaakt door Felienne en aangepast door Emily Bakker. Het is Creative Commons by-nc-sa-4.0

Simpel gezegd: Je mag het gebruiken in je lessen, aanpassen, uitprinten, kopieren, wat je maar wilt. Maar: Je moet mijn naam erbij zetten, je mag er geen geld mee verdienen en als je het aanpast, moet je dat ook weer Creative Commons maken. Het linkje bevat alle informatie.

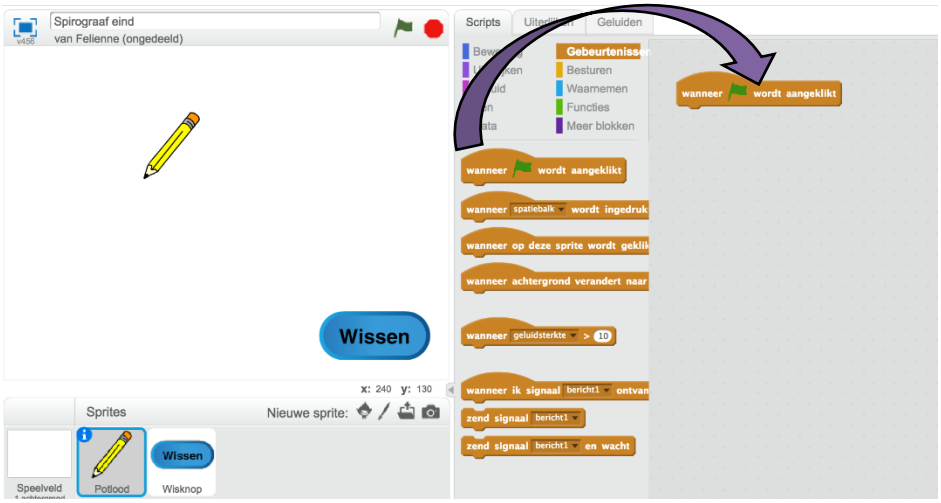
De eerste lijnen

Eerst maar eens zorgen dat we iets op het scherm kunnen tekenen.

Ga nu naar

, kies

en sleep het op het grijze vlak.
We gaan de spirograaf laten beginnen met de groene vlag.



Het blokje

geeft aan *wanneer* iets gebeurt.
Er gebeurt iets, als je op de groene vlag klikt. Nu moet er nog iets gebeuren.

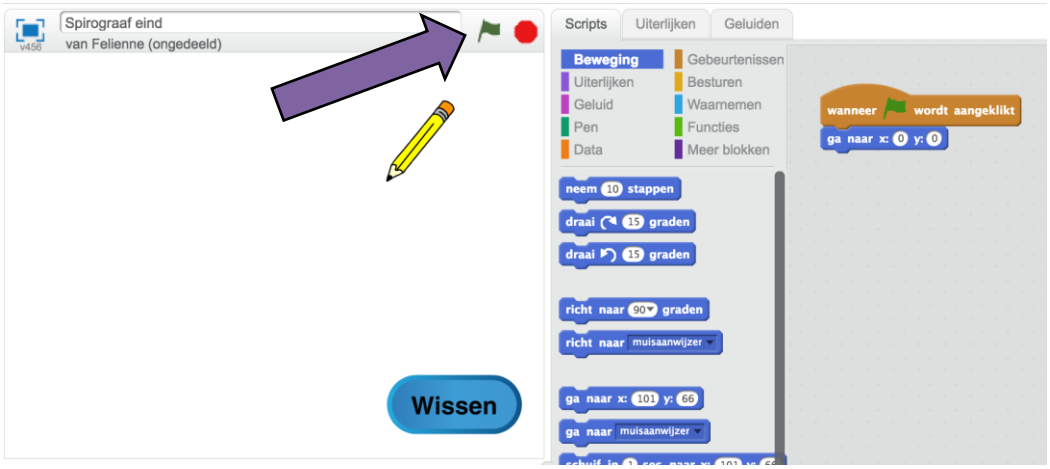
We gaan als eerste de pen verschuiven. Ga naar

en pak

.

Plak dat onder

. Verander de getallen die je hebt in nullen. Dat doe je door er op te klikken met je muis en dan de 0 in te tupen. Je krijgt dan dit:



Probeer je verhaal eens uit door op de groene vlag te klikken. Die vind je bij de hele pijl. Wat gebeurt er nu?

Precies! Het potlood beweegt naar het midden van het scherm. Plak hier ook een

blokje onder zodat het potlood naar rechts gaat bewegen.

Een eerste streep

Nu kan het echte tekenen beginnen. Ga maar eens naar Pen. Pak het blokje pen neer.

Nu is de pen op het scherm neergezet, en kunnen we dus tekenen. Als je nu een neem 10 stappen pakt (Beweging) en onderaan je blokjes legt, dan komt er een klein streepje op het scherm.

Heb jij ook dit resultaat?

Dan zit je op de goede weg.

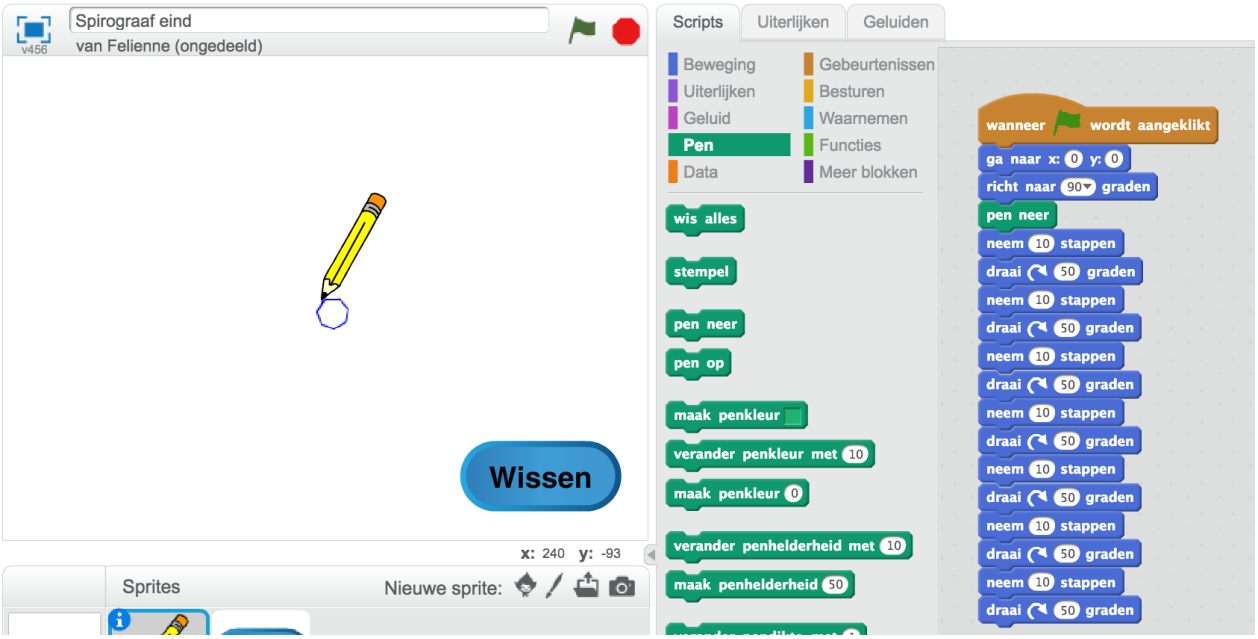
Draaien maar

Voor een mooie spirograaftekening willen we natuurlijk niet alleen een rechte lijn. we willen ook mooie bochtjes. Daarvoor moet de pen gaan draaien. Daar us ook een blokje voor.

Het zit ook bij Beweging. Plak een draai 15 graden blokje onderaan, en maak het getal maar wat groter, dan zie je het goed. Bijvoorbeeld 50.

Kijk, het potloodje is nu een gedraaid. Dat is het begin van een rondje.

Nu zouden we zo'n rondje met heel veel blokjes kunnen maken, zo:



Dat mag je proberen, maar dat is dus heel veel werk! We kunnen het ook slimmer doen, kijk maar eens op de volgende bladzijde.

Herhaalverhaal

In Scratch heb je ook een handig blokje om dingen te herhalen.

Het blokje heet:

en zit bij

Doe de

om de twee blokjes

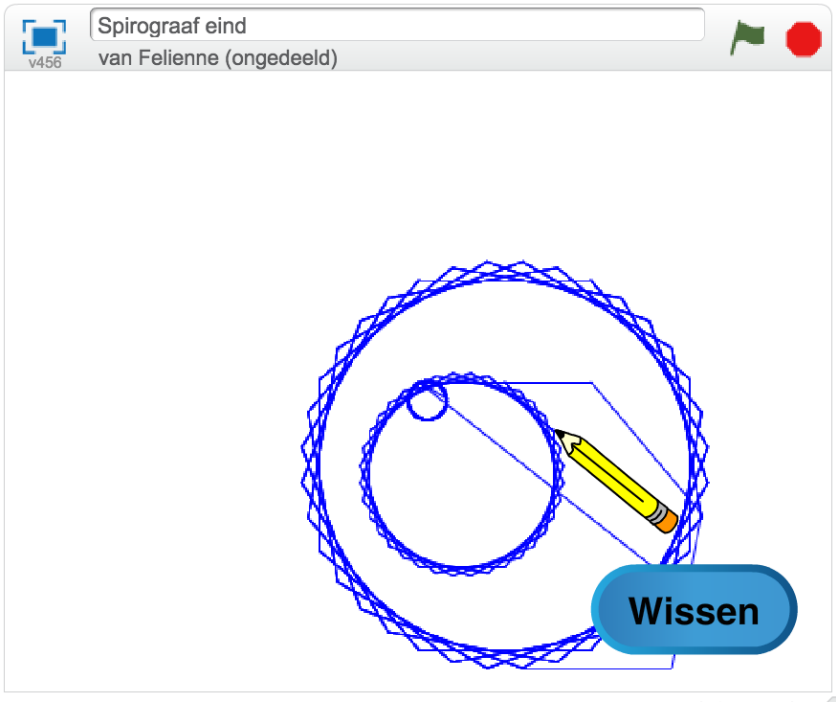
heen.
Als je het een beetje in de buurt houdt, dan klikt het er vanzelf omheen:

Probeer je tekening weer eens uit. Dat is mooi he, een heel rondje met weinig blokjes!

Helaas krijgen we nu steeds hetzelfde rondje. Zullen we eens wat variatie toevoegen?

Probeer bijvoorbeeld eens 50 of 100 stappen te nemen:

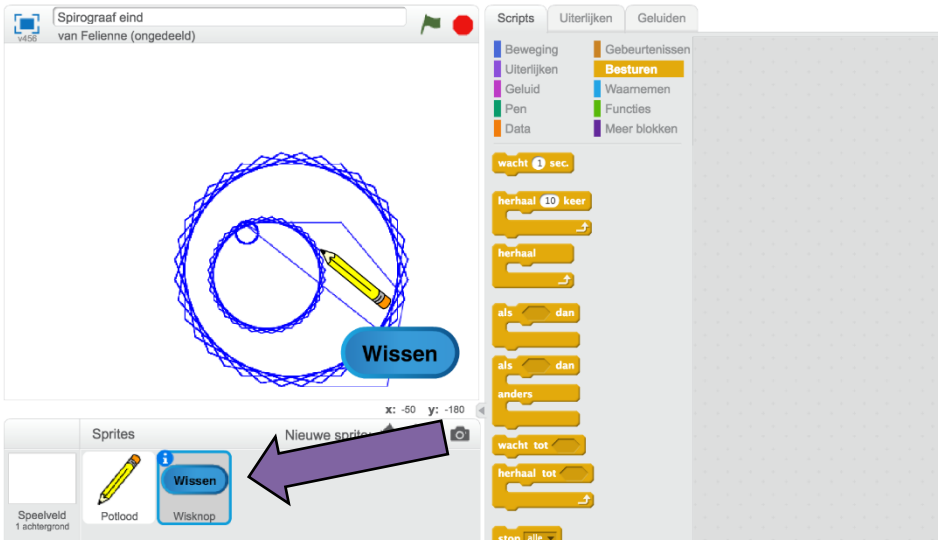
Hier heb ik 50 en 100 geprobeerd:



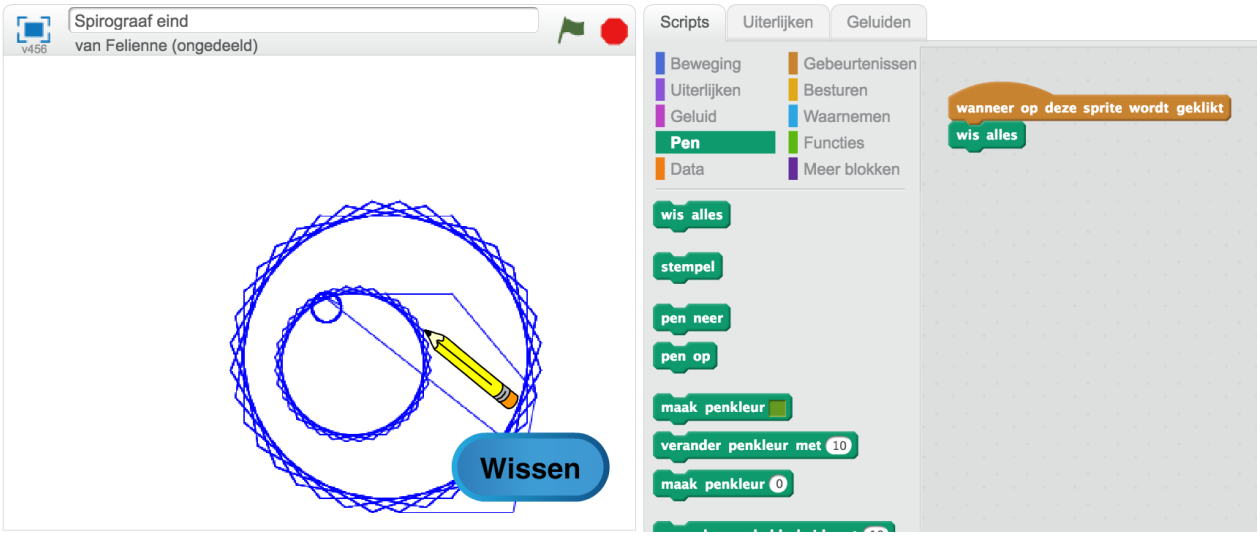
Zo kun je steeds een ander rondje maken. Helaas wordt het nu wel een rommeltje.

Er is ook een wisknop! Zullen we die wisknop eens gaan programmeren?

Klik onderaan op de Wisknop. Die heeft nog geen blokjes:



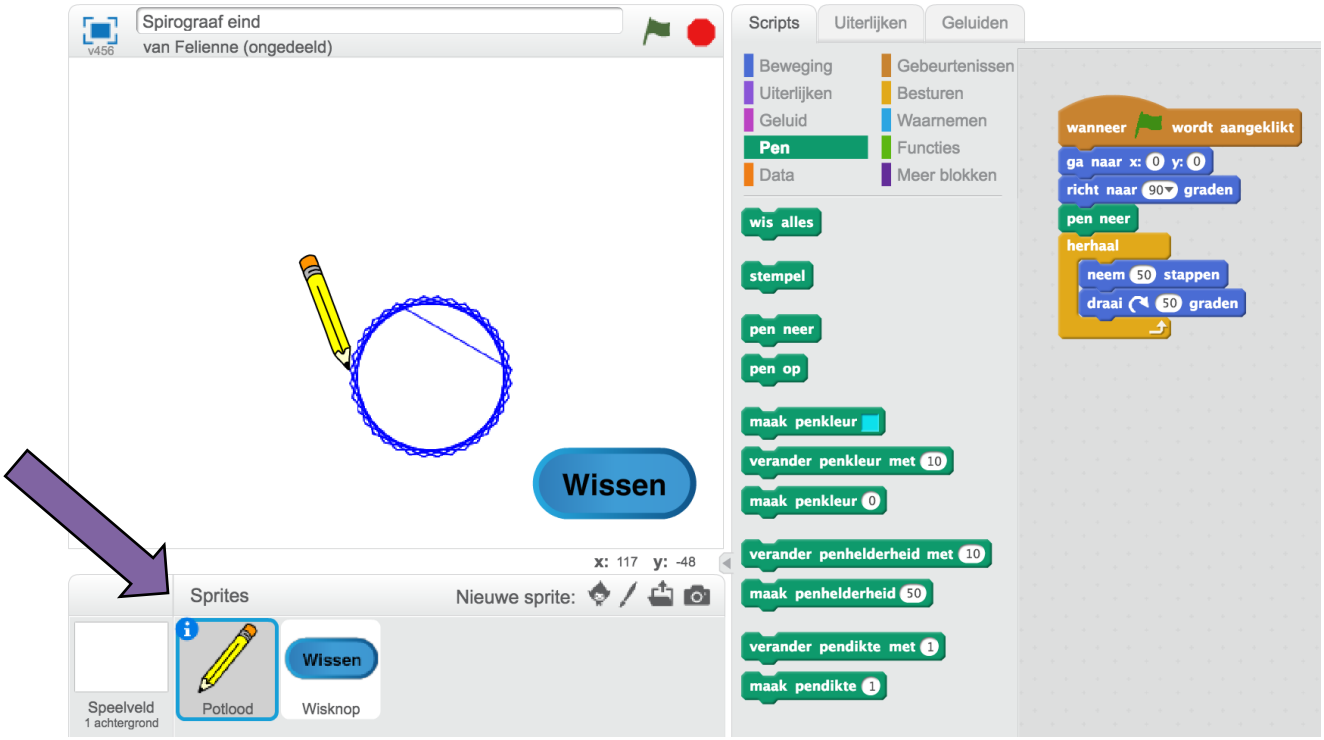
Als we op de knop klikken, dan moet alles gewist worden. Dat doen we zo:



Klik nu maar eens op de knop (bij de gele pijl). Is alles nu netjes opgeruimd? Bij mij wel!

Als het te rommelig wordt, kun je nu altijd op de wisknop drukken.

Ga nu weer naar het potloodje:



Misschien heb je ook soms zo'n streepje in je rondje zoals hierboven. Dat is niet mooi he? We kunnen dat oplossen door de pen omhoog te doen als we terug naar het begin gaan:

Heb je ook al geprobeerd om een ander getal te gebruiken bij

?

Probeer maar eens wat getallen! Wat gebeurt er dan? Teken hier jouw figuren maar eens:

Vergeet niet tussendoor te wissen.

Programma	Tekening

Vul hier eigen getallen in!

Ik hou het even op 200 en 200, maar misschien heb jij wel wat anders.

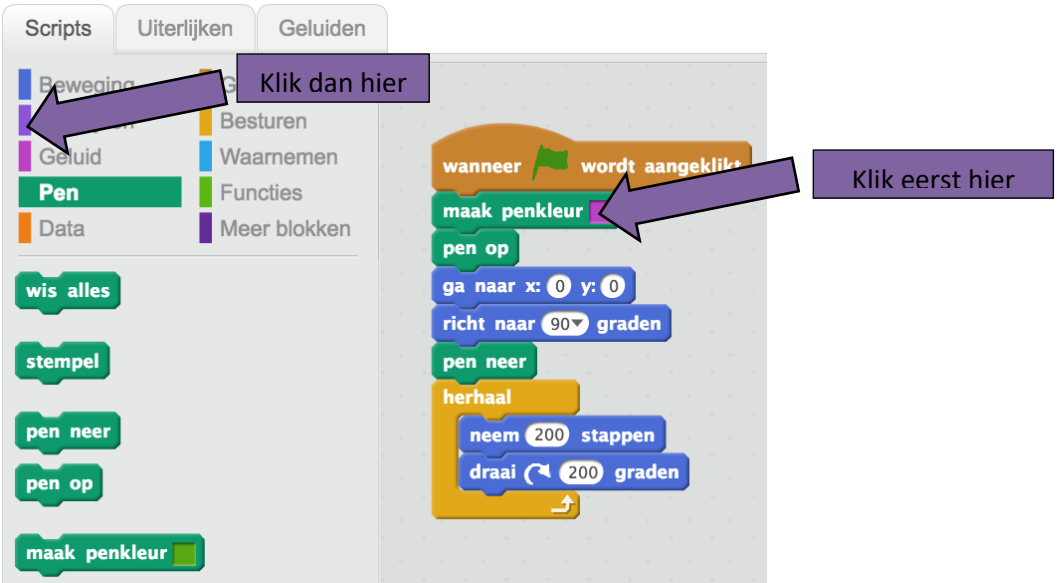
Er zijn nog wat leuke dingen die je kan doen om je tekening mooier te maken.

Kleurtjes

Alle tekeningen zijn nu blauw. Dat is best mooi, maar misschien wil je wel iets anders.

Je kunt de kleur veranderen met het blokje

. Misschien staat er bij jou een andere kleur, dat maakt niets uit! Plak dit blokje bovenaan.
Nu moet je eerst op de kleur klikken in het blok, en dan op de kleur die je wilt. Kijk goed hier:

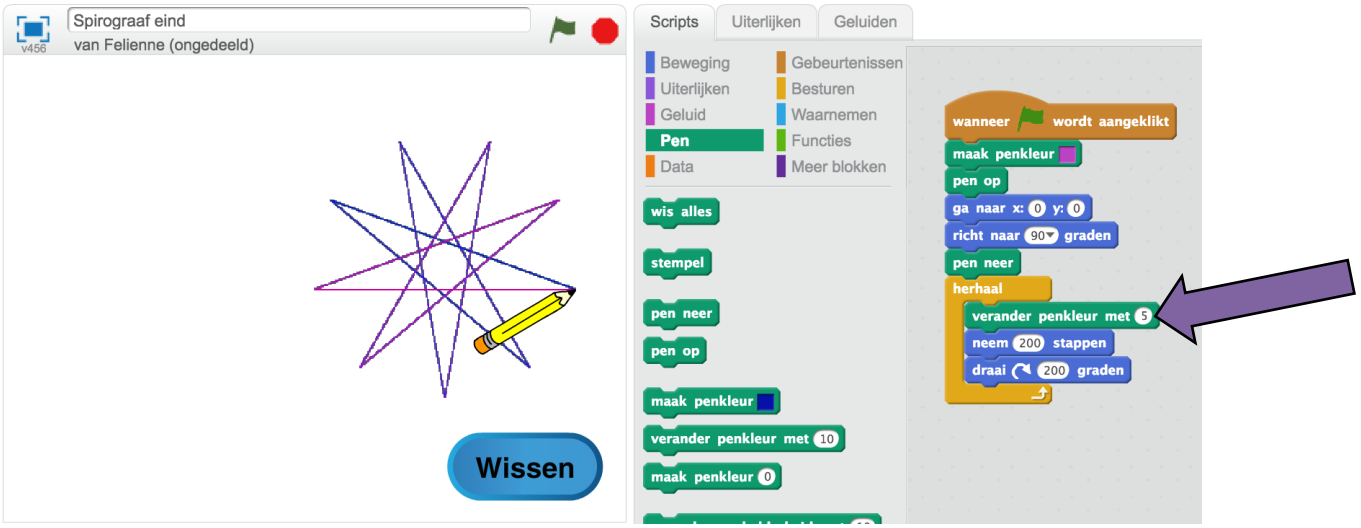


Nu wordt je tekening mooi paars of een andere kleur als je die hebt gekozen.

Heb je moeite met kiezen? Je kunt de kleur ook automatisch aanpassen!

Dat doe je met

. Zet het bij de gele pijl.

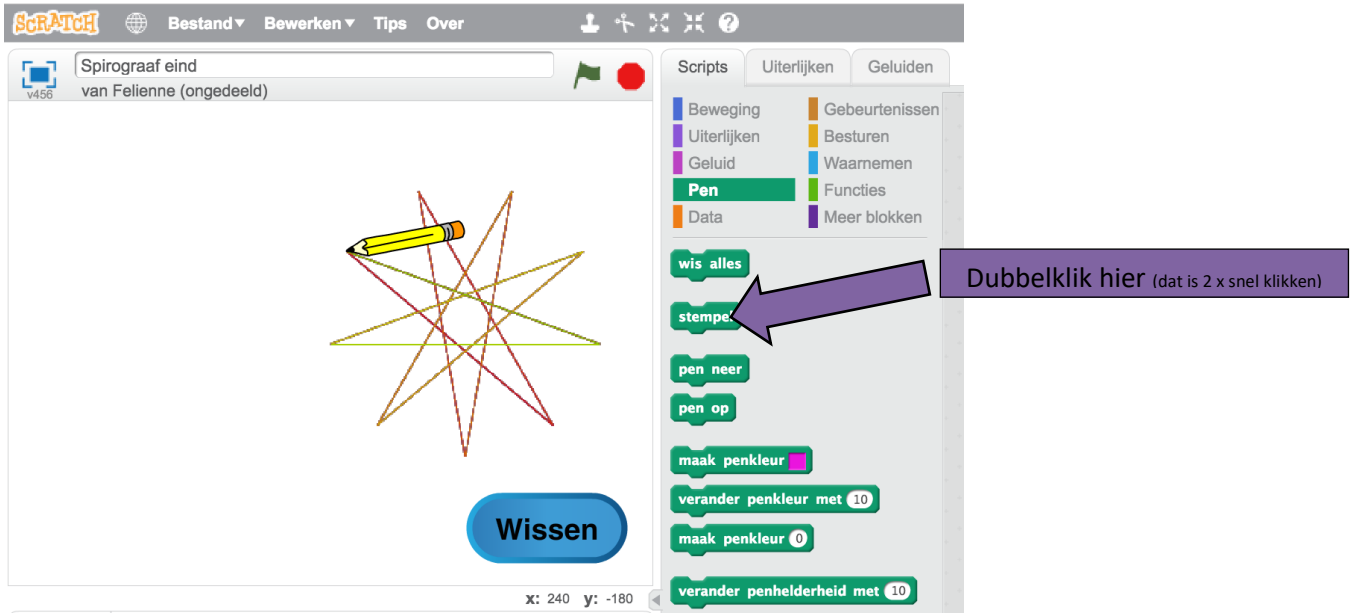


Nu krijg je bij iedere stap een andere kleur. je kunt hier ook weer het getal 5 op iets anders zetten, tot je het zelf mooi vindt.

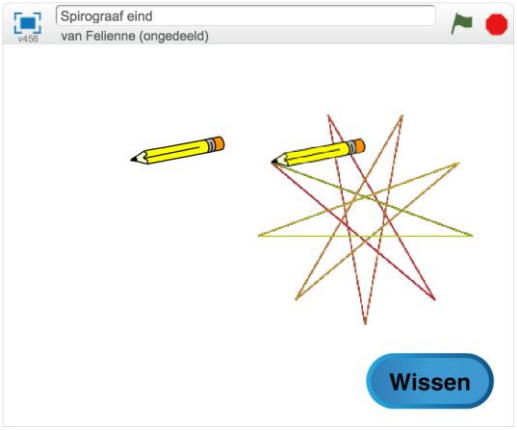
Druk er je stempel op!

We gaan ook nog wat versiering toevoegen rondom je tekening.

Hiervoor gebruiken we het stempelblok. Klik maar eens twee keer op het blok in het midden.
Je hoeft het dus niet op het grote grijze vlak te slepen.



Er gebeurt nu nog niks, maar.... sleep nu het potloodje eens weg met je muis.



Het potlood heeft een afdruk gemaakt!

Met zo'n afdruk kun je leuke dingen doen. Zet bijvoorbeeld eens een

in de

.

Probeer je tekening uit met de groene vlag. Wat gebeurt er nu?

Wil je een ander figuurtje? Dat kan ook!

Bij Uiterlijken vind je verander uiterlijk naar streep. Plak die in de herhaal (let op! zet 'm boven stempel)

Kies met het kleine driehoekje een mooi figuurtje uit. Een voetbal of een eenhoorn? Jij mag kiezen.

Groter en groter!

Tot slot gaan we de tekening nog wat gekker maken. Vind je het ook een beetje saai dat je steeds dezelfde tekening krijgt? Zou het niet leuker zijn als Scratch zelf wat verzint?

Ga maar eens naar Data. Daar staan twee oranje blokjes voor je klaar.
Dat zijn variabelen "va-rie-ja-buh-luh".

Het is een soort knijper, dat een getal vasthoudt. In een variabele kan maar één getal gelijker tijd. Als je er een nieuw getal in stopt, gaat de oude weg.

Sleep ze in de blauwe blokjes: LET OP! Ze staan net andersom.

Nu moeten we de getallen nog instellen. Zet twee

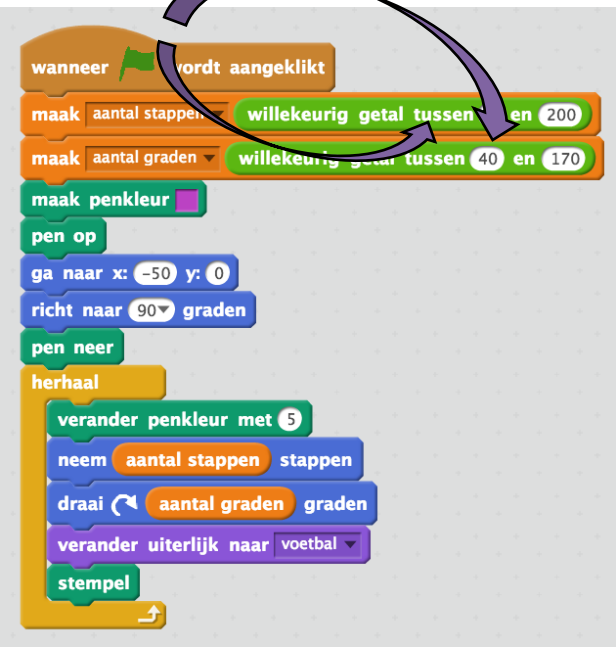
bovenaan.
Zet er eentje op aantal stappen met het kleine driehoekje.

Ga nu naar

en pak daar

. Stop die in

.



Klik nu eens een paar keer op de groene vlag. Wat gebeurt er?

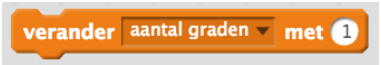
Tip: als je geen zin hebt om steeds op de wisknop te drukken, kun je ook nog een

blok helemaal bovenin zetten

Ben je klaar? Goed gedaan!

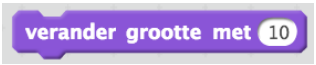
Hier zijn nog wat dingen die je kan proberen.

- Zet deze eens in de herhaal. Wat gebeurt er nu?

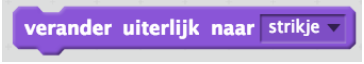
  

  Probeer ook eens andere getallen. Onder nul kan ook! Zet dan een minnetje voor het getal.

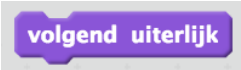
- En zet deze eens in de herhaal:

- Wat je ook nog kan doen is dit blokje weghalen

en deze ervoor op de plaats zetten



Wat gebeurt er nu?

Goed geScratcht!!

Bijlage 5 Experiment Les 1

Benodigdheden: wasknijper met label, aantal vellen papier, stift/pen.

“het zijn een soort doosjes waar je een getal in op kan slaan” vervangen door:
“Het is een soort knijper, dat een getal vasthoudt. In een variabele kan maar één getal gelijker tijd. Als je er een nieuw getal in stopt, gaat de oude weg” (voor alle groepen).

Benodigdheden:

- Wasknijper met sticker met ‘aantal kinderen’
- Wasknijper met sticker met ‘aantal spijkerbroeken’
- Stift & papier.

Extra uitleg:
Controleer aan de hand van de volgende vragen of de leerlingen begrijpen wat een variabele is, voordat je het voorbeeld geeft. Eronder de uitleg nadat ze het antwoord gegeven hebben. Laat ze telkens op hun wisbordje ‘waar’ of ‘niet waar’ schrijven.

1. In een variabele kun je een getal stoppen. (waar)
- In een variabele kun je een getal bewaren.
2. Het getal van een variabele kun je niet veranderen. (niet waar)
- Het getal van een variabele kun je wel veranderen, zo vaak als je wil.
3. In een variabele kun je meerdere getallen bewaren (niet waar!)
- In een variabele past maar één getal gelijker tijd (hoe groot die is maakt niet uit).
4. Als ik een nieuw getal in de variabele stop, ‘vergeet’ hij de oude (waar)
- Er past maar één getal in een variabele, als je een nieuw getal in de variabele stop, gaat de oude weg.

Oefening:
Geef nu het voorbeeld met de wasknijper (met sticker ‘aantal leerlingen’).
Laat de wasknijper aan de leerlingen zien. “Deze wasknijper werkt net als een variabele. Hij heet: ‘aantal kinderen’ en laat zien hoe veel kinderen er in de klas zitten. Laten we eens tellen hoe veel kinderen er vandaag in de klas zitten”. Tel samen met de klas het aantal kinderen en schrijf het getal op een briefje en klem het in de wasknijper. “De variabele geeft nu aan dat er …. kinderen in de klas zitten. Maar stel dat er vandaag 1 kind ziek zou zijn”. Wijs 1 kind aan en vertel “we doen net alsof hij/zij ziek is vandaag”. Zet hem/haar op een plek waar de kinderen hem/haar niet meer kunnen zijn, maar het kind de les nog wel kan horen.
“Onze variabele zegt dat er nu … kinderen in de klas zitten, maar als (*naam)* ziek is, klopt dat getal niet meer dus moeten we het veranderen. Misschien weten jullie het nieuwe getal al, maar laten we toch even tellen”. Tel samen met de kinderen het aantal kinderen in de klas. Schrijf het nieuwe getal op een nieuw papier. Vertel “als ik een nieuw getal in mijn variabele stop, valt de oude eruit en is die weg. Kijk maar”. Houdt de wasknijper omhoog, doe hem open en laat het papier eruit vallen en stop het nieuwe erin. Herhaal “een variabele houdt dus maar één getal vast”.

“Laten we nog een voorbeeld oefenen”. Pak de wasknijper met sticker “aantal spijkerbroeken” en laat deze aan de klas zien. “ik heb nu een andere wasknijper die ook net werk als een variabele. Deze variabele heet “aantal spijkerbroeken”, zo kunnen we bijhouden hoeveel kinderen in de klas een spijkerbroek aanhebben. Ga allemaal maar eens staan, dan kunnen we samen tellen”. Tel samen het

aantal spijkerbroeken in de klas. Schrijf het getal op een briefje en klem het briefje zichtbaar voor iedereen in de wasknijper. Vertel “volgens de variabele, zitten er nu … kinderen met een spijkerbroek in de klas. Maar net als het aantal leerlingen in de klas, kan dit getal veranderen. Er zijn morgen vast kinderen die iets anders aandoen. Stel je voor dat er morgen 2 (voor groep 7/8: 5) kinderen meer een spijkerbroek aanhebben. Welk getal denk jij dat er dan in de variabele moet? Schrijf maar op”. Controleer de antwoorden. Vertel “Het oude getal is…., daar moeten we 2 (of 5) bij optellen. Het nieuwe getal is dus….”. Schrijf het nieuwe getal op, houd de wasknijper omhoog, knijp hem open en stop het nieuwe briefje erin. Vertel ondertussen “Er past maar één getal gelijker tijd in, als ik een nieuw getal in de variabele stop, gaat de oude weg”.

Ga nu verder met de les.

Bijlage 6 Les 2

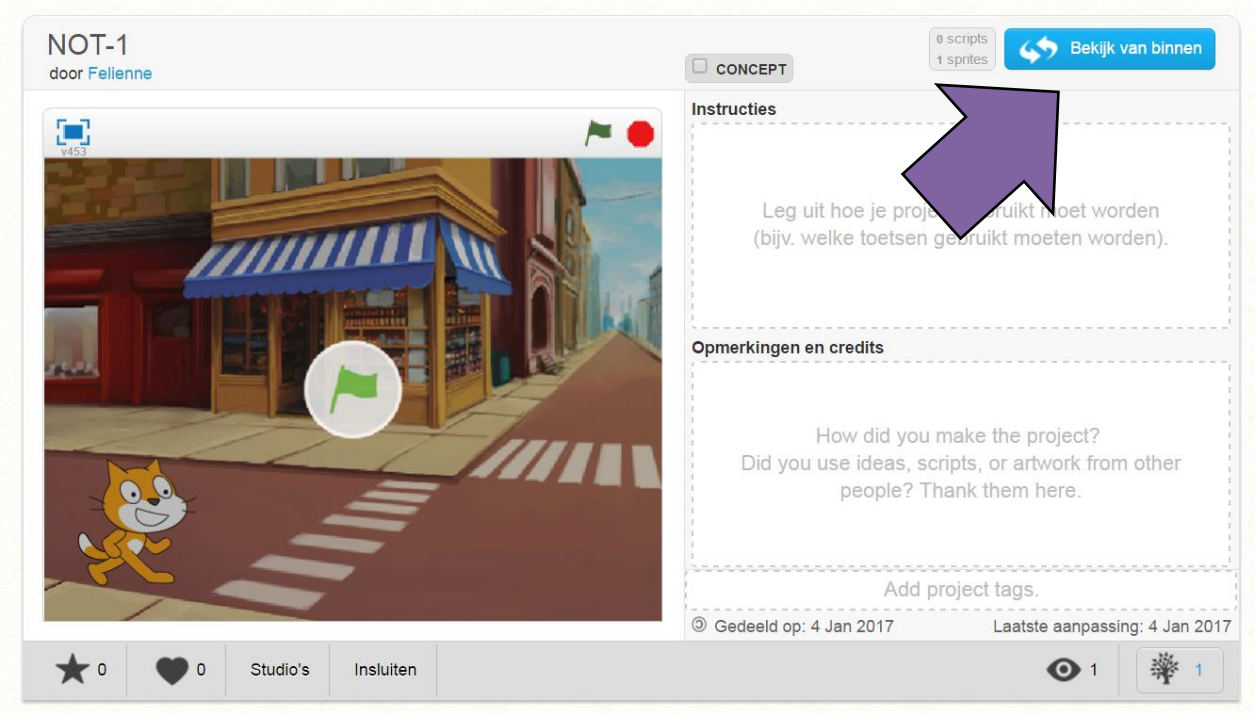
* De pagina lay-out wijkt af van het originele ontwerp i.v.m. ruimtebesparing.

Scratch les 2 | Rekenen

Een rekenmachine met Scratch!

We kunnen met Scratch zelfs rekenen! Laat Karel de Kat je helpen met tafels. Misschien kun jij met Scratch zelfs je broertje of zusje wel helpen rekenen!

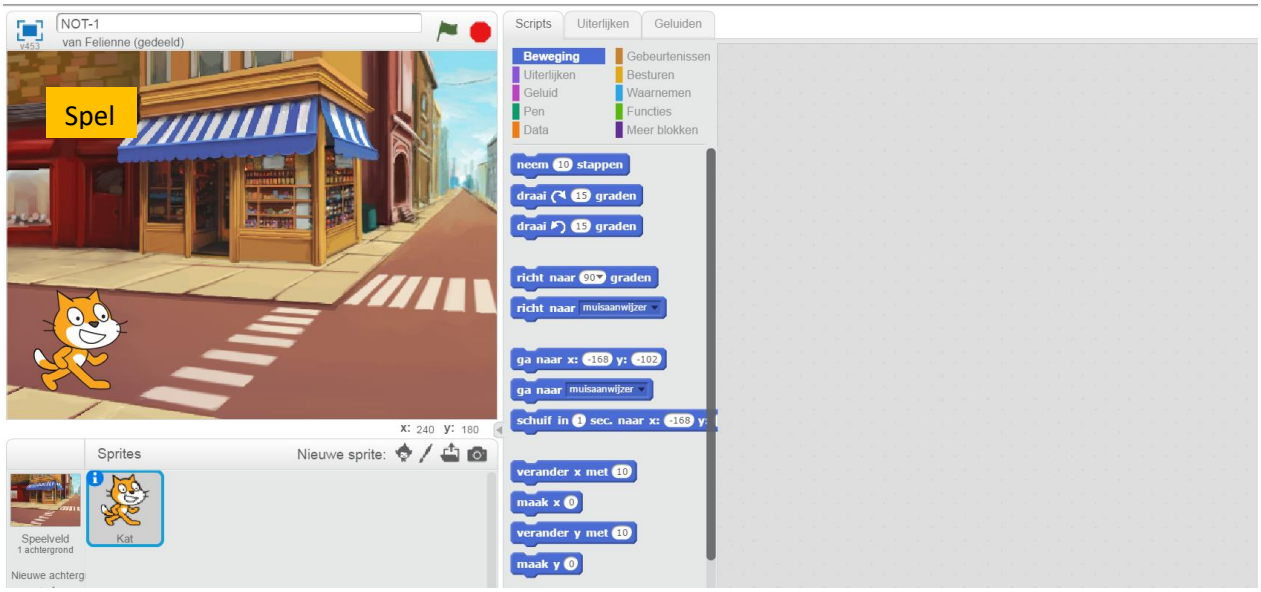
Ga naar: http://bit.ly/NOT-1



Je ziet nu het plaatje zoals hierboven. Klik op Bekijk van binnen (zie bij de pijl).

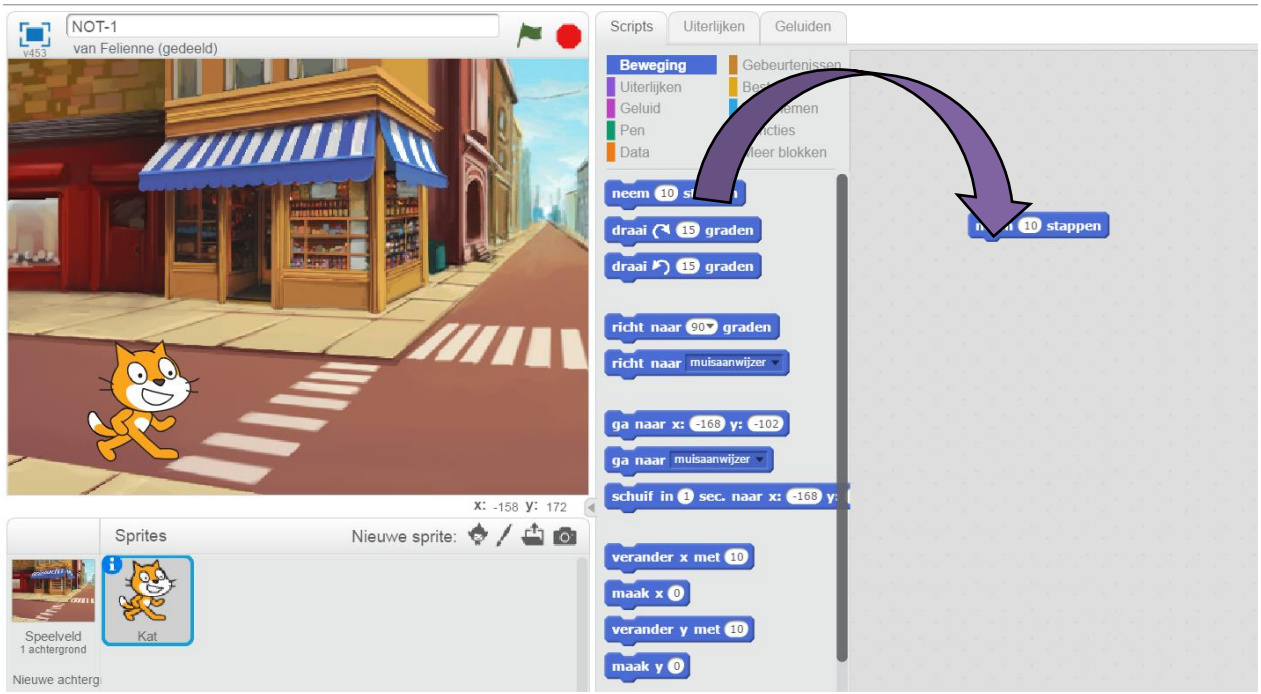
Dit materiaal is gemaakt door Felienne en aangepast door Emily Bakker. Het is Creative Commons by-nc-sa-4.0 Simpel gezegd: Je mag het gebruiken in je lessen, aanpassen, printen, kopiëren, wat je maar wilt. Maar: Je moet mijn naam erbij zetten, je mag er geen geld mee verdienen en als je het aanpast, moet je dat ook weer Creative Commons maken.

Hier gaan we programmeren!



We zetten eerst Karel de Kat aan het lopen. Pak maar eens het blokje

op en sleep het op het programmeervlak. Zo:

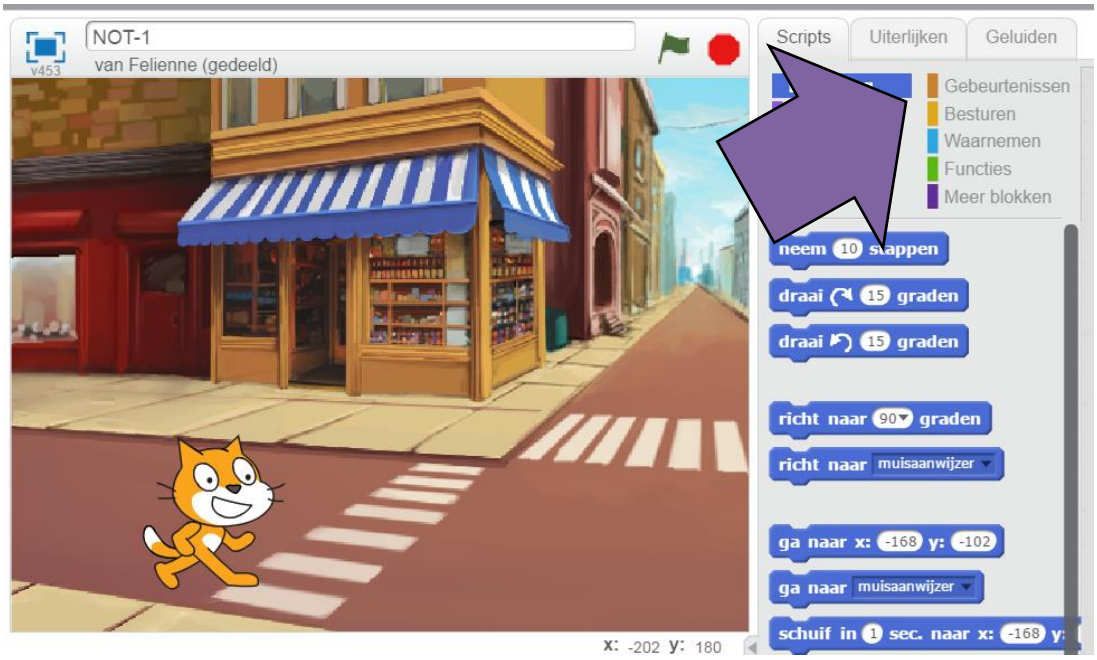


Opdracht. Klik nu dubbel (twee keer snel achter elkaar) op het blokje dat je naar het programmeervlak hebt gesleept.
Wat zie je nu? De kat gaat vooruit!!

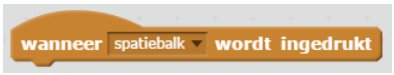
Het is natuurlijk niet zo'n leuk spel als je steeds moet dubbelklikken, laten we er een 'knop' voor maken.

Zoals je ziet, hebben de blokjes de vorm van een soort puzzelstukje. Dat betekent dat je ze aan elkaar kan leggen om grotere programma's te maken.

Dat gaan we nu proberen! Klik op

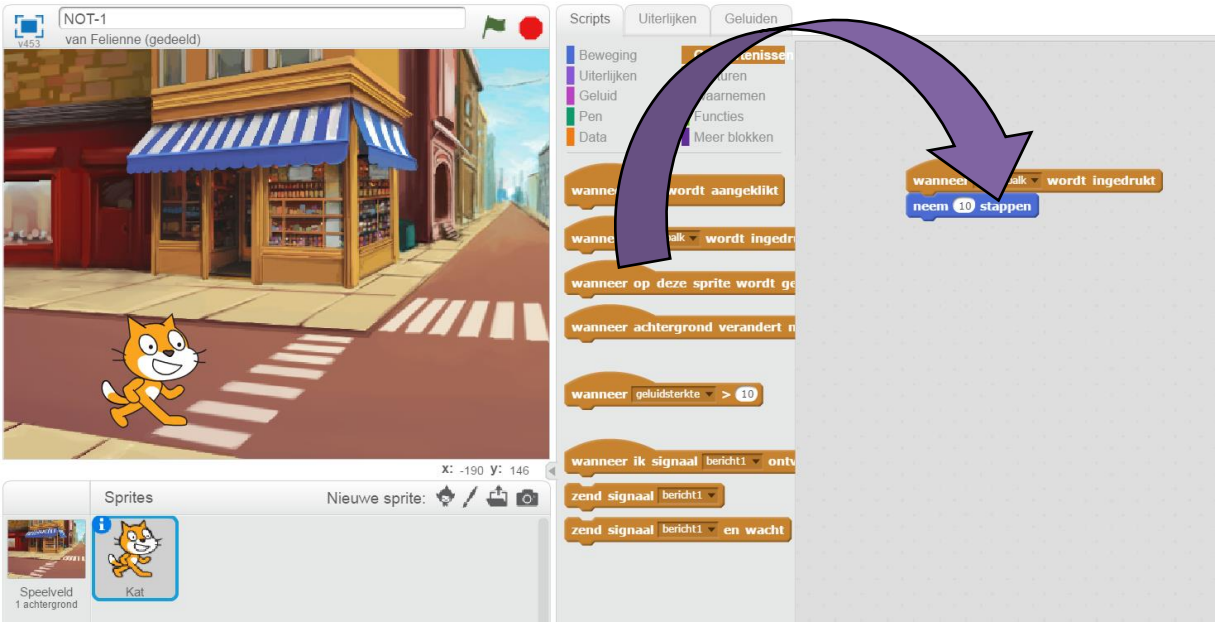


De blokjes met de ronde bovenkant geven aan *wanneer* iets moet gebeuren.

Pak 

en plak die bovenaan het blokje

dat je al had. Dat kan maar op 1 manier.



Druk nu maar eens op de spatiebalk. Nu gaat de kat ook vooruit!

Reken maar!

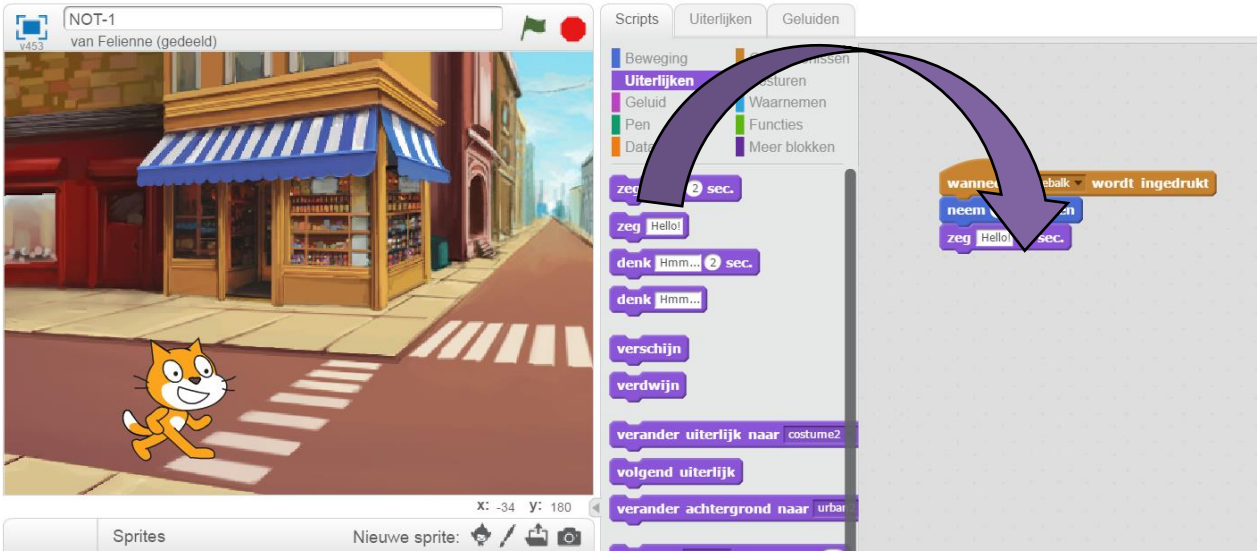
Jullie hebben natuurlijk ook rekenlessen. Maar we kunnen ook rekenen en programmeren tegelijkertijd doen! Dat doen we door met Scratch te rekenen.

Wat je met rekenen vast heel vaak doet, is tafels oefenen. Met tafels kun je ook heel goed programmeren! Het is een echt *algoritme:* dat is een programmeer-woord voor een plan van aanpak.

Als we straks klaar zijn met tafels oefenen, kun je misschien ook nog wel andere dingen verzinnen om te oefenen!

Wat zeggie?

Het eerste wat je moet weten is dat je de figuurtjes in het spel ook iets kan laten zeggen. Ga naar Uiterlijken. Je krijgt nu weer nieuwe blokjes. Pak zeg Hello! 2 sec. en plak 'm onder de blokjes die je al had:



Druk nu nog maar eens op de spatiebalk. Kijk, nu beweegt de kat eerst 10 stappen, en zegt daar, 2 seconden lang, "Hello!"

Nu is het natuurlijk onze bedoeling om de kat de tafels te laten zeggen in plaats van "Hello!". Hiervoor hebben we iets nodig, de variabele!

Variabelen

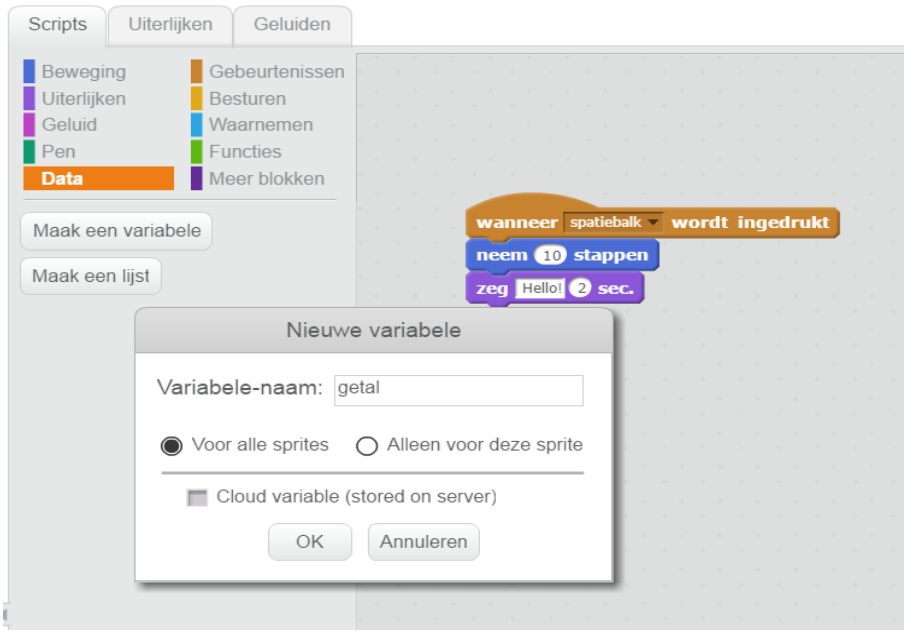
Variabelen kom je veel tegen als je gaat programmeren.

Het is een soort knijper, dat een getal vasthoudt. In een variabele past maar één getal gelijker tijd. Als je er een nieuw getal in stopt gaat de oude weg.

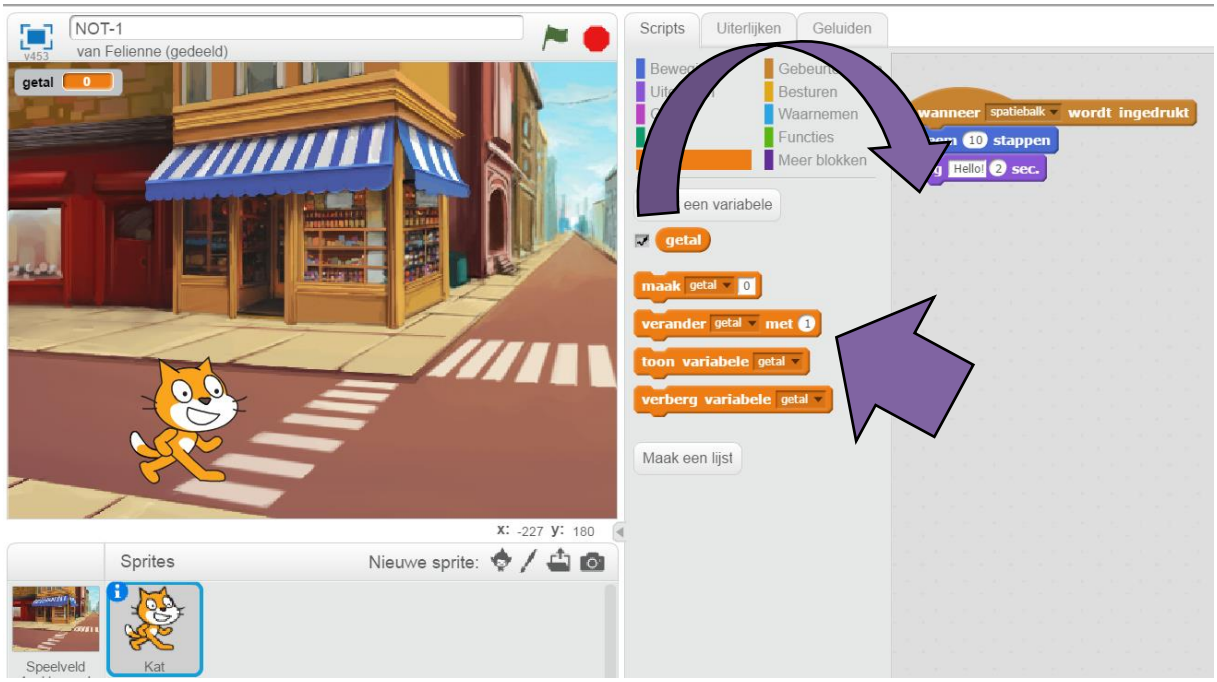
Het werkt zo, kijk maar mee!

Een variabele maak je bij Data. Ga er maar naar toe en klik op Maak een variabele.

Je krijgt nu dit te zien:



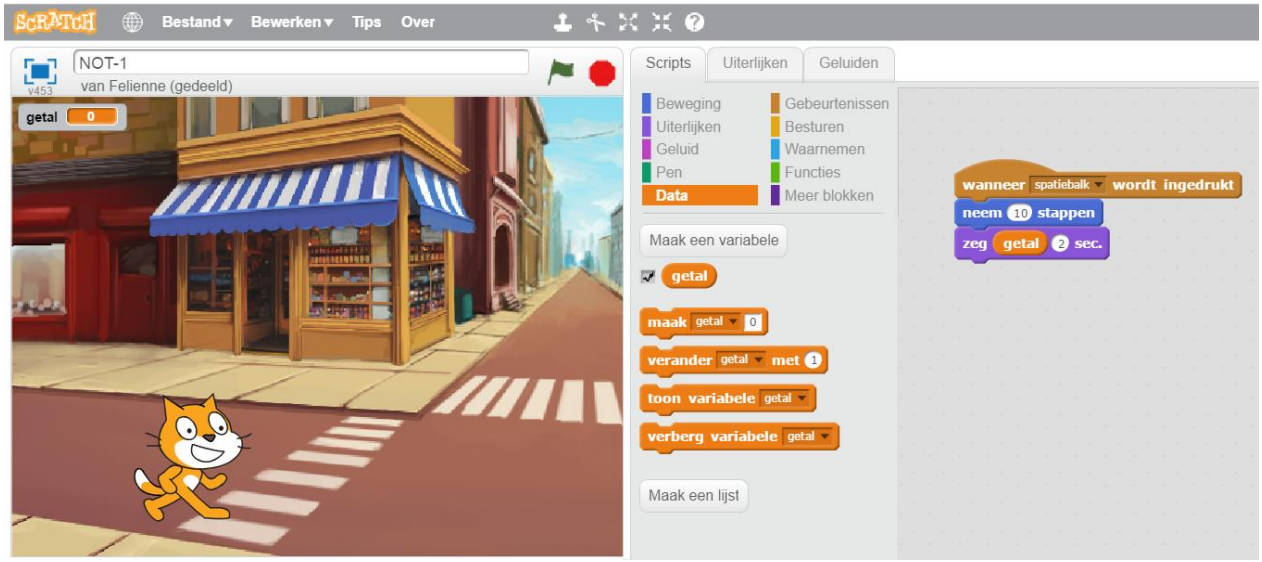
Typ 'getal' en klik daarna op OK.



‘Getal’ is nu een variabele: denk maar aan de knijper. Speciaal daarvoor hebben we er nu een aantal blokjes bij gekregen (kijk bij de rechte pijl).

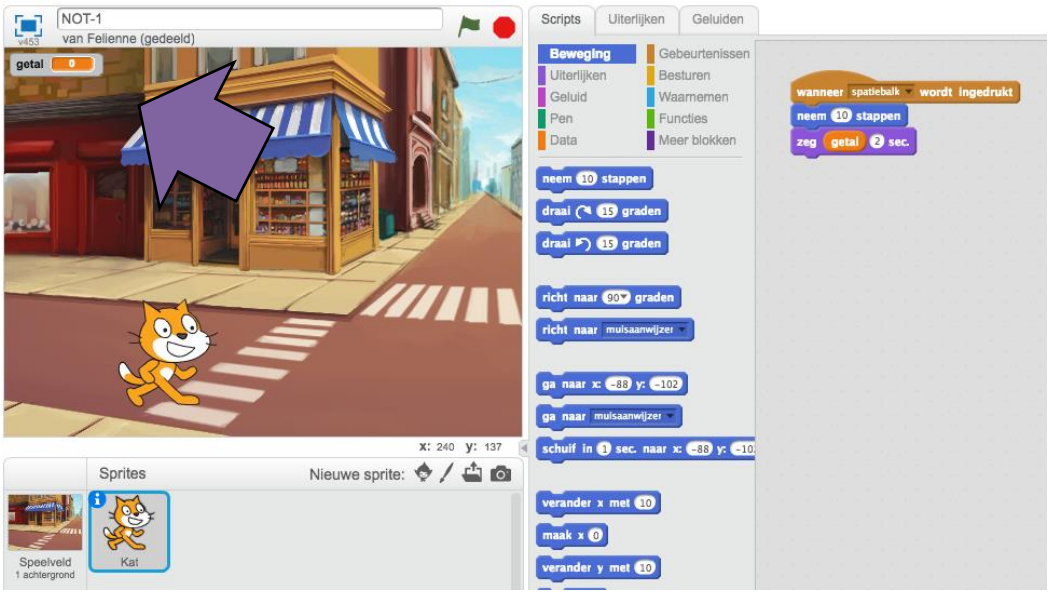
Zullen we de kat het getal eens laten voorlezen? Dat doen we door het oranje, ronde getalblokje op de witte plek in het blok

te stoppen (zoals de ronde pijl aangeeft)
Ondanks dat de een rond is en de ander vierkant, past dat toch!



Opdracht. Wat zou er gebeuren als je nu op de spatiebalk drukt? Raad het eerst en probeer het dan uit.

Hier nog een gratis Scratch tip: je kunt bovenin de hoek de waarde zien die in de variabele is opgeslagen. Kijk bij de pijl:



Aan tafel!

Zo, nu is het tijd om een echte tafel te maken. Zullen we zeggen de tafel van 5?

staat nu op 0, dus het eerste wat we moeten doen is…. 5 erbij.

Dat doen we met een

blok. Zet dit blok boven je

blok:

Verander het getal in het oranje blokje in 5, zoals in de afbeelding hierboven.
Nu krijg je er iedere keer 5 bij als je op de spatiebalk drukt.

In feite hebben we al een tafel he? We kunnen gewoon 10x op de spatie duwen. Maar het leuke aan computers is juist dat ze dingen makkelijker voor je kunnen maken.

Wat we willen is het plus 5 steeds herhalen. Daar heeft Scratch een

blok voor. Ga naar

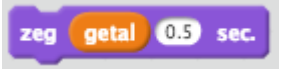
en kies daar de

uit. Sleep die om de drie blokjes heen:

Zet stappen op 30 en seconden op 0.5 zodat dadelijk de kat sneller uit beeld gaat.

Dat gaat eigenlijk vanzelf als je de

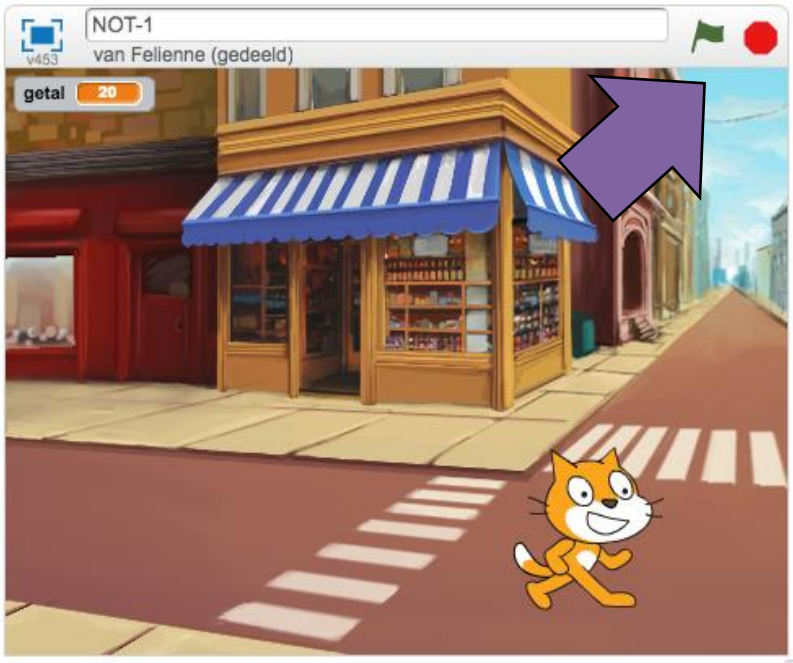
er een beetje in de buurt houdt.

Ik heb ook de seconden in ‘t 

blok op 0.5 gezet anders moet je zo lang wachten :).

Terug naar af

Wat je nu al wel gezien hebt, is dat je tafel niet meer stopt. Je kunt het spel alleen nog stoppen door op de rode knop te klikken:



Steeds begint je getal waar je gestopt was. Dat kun je oplossen door aan het begin

op 0 te zetten met

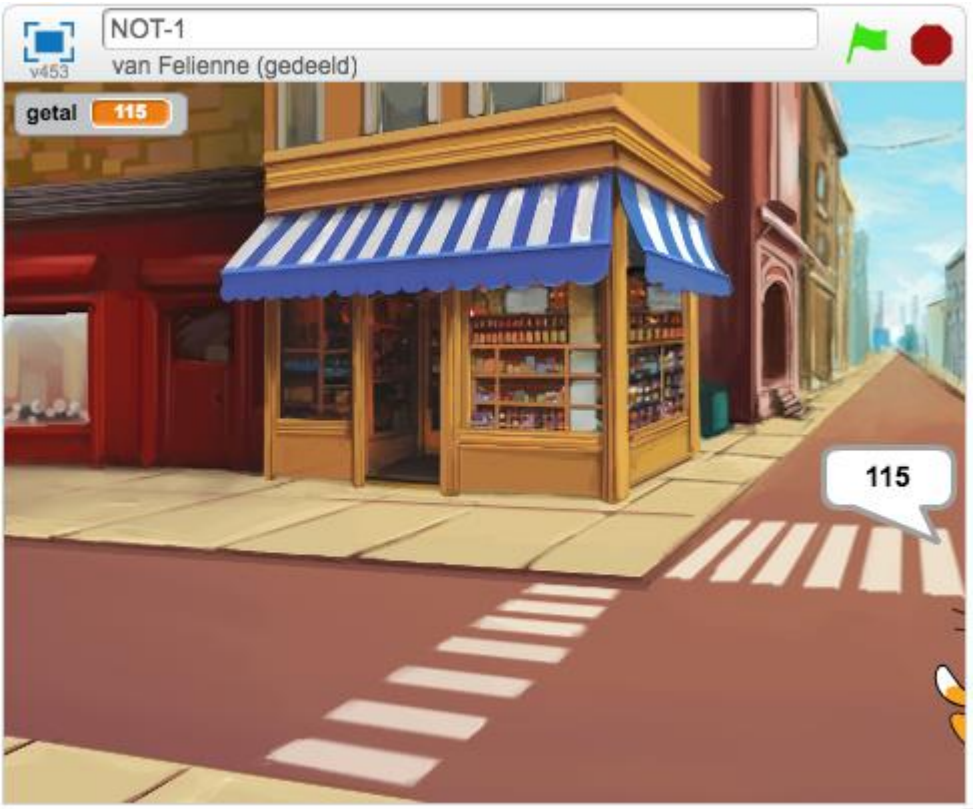
.

Waar zou dat blok moeten komen denk je?

Zo zet je het getal steeds op 0:

Maar, ons programma heeft nog meer problemen! Die noemen we in computertaal *bugs*.

Opdracht. Welke bugs ontdek je in dit plaatje? Het zijn er 2.



Welke problemen ervaar jij zelf en welke zou jij op willen lossen? Op de volgende pagina lees je meer over welke 2 ‘bugs’ ik bedoel.

De twee problemen die we hebben zijn:

- De kat gaat aan het begin niet naar links
- De tafel stopt nooit meer

De kat gaat aan het begin niet naar links

Net zoals we

aan het begin op 0 moeten zetten, zo moeten we ook de kat links neerzetten. In computertaal heet dit initialiseren. Denk aan initialen, die staan ook *voor* je naam.

Ga terug naar

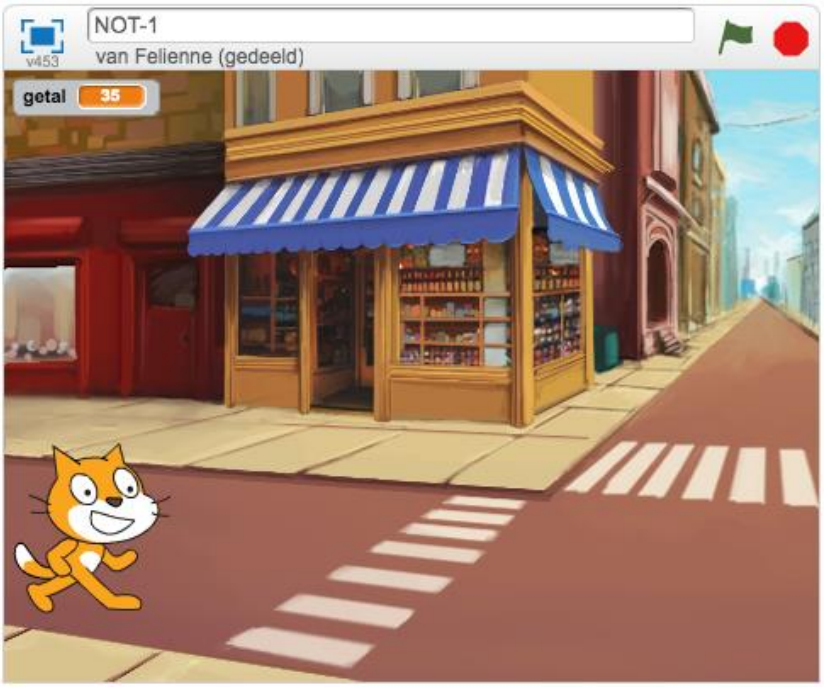
.

Wat nu belangrijk is om te weten, is dat je de kat kan verplaatsen met het

blokje.

In de twee witte ovalen stop je de x en y positie van de kat. Je kunt daar met de hand getallen invullen, probeer dat maar eens uit!

Maar het kan ook handiger. Sleep de kat met je muis naar links, zo:



Je ziet nu dat de getallen in het

blokje veranderd zijn! Ze zullen bij jou ook ongeveer op de getallen -188 en 089 staan.

De getallen worden automatisch op de locatie van de kat gezet. Makkelijk, toch?

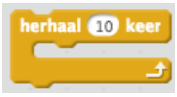
Dit blokje zetten we bij

:

Nu gaat de kat steeds netjes naar links, steeds als je op de spatiebalk drukt.

De tafel stopt nooit meer

Een

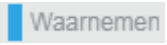
is mooi, maar die gaat eindeloos door, zoals je ziet. Vervang ‘m door een 

en je hebt een keurige tafel!

Nog even verder!

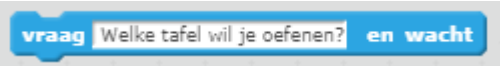
We kunnen natuurlijk zelf telkens een andere tafel kiezen en programmeren in Scratch. Maar, we kunnen ook de speler een tafel laten kiezen!

Daarvoor gebruiken we het

blok. Die staat bij 

.

De vraag die je hier invult, kan de speler beantwoorden. Vraag bijvoorbeeld



.

Nu staat bij Waarnemen een nieuwe variabele: antwoord.

We beginnen met getal op 0. En we veranderen getal met antwoord

Probeer maar eens!

Nu kun jij met Scratch de aller moeilijkste tafels oefenen! Of zelfs je broertje of zusje uitleggen hoe je met tafels moet rekenen!

Bijlage 7 Experiment Les 2

Les 2, Rekenen

In deze les zitten variabelen verwerkt. Voor alle groepen staat dezelfde uitleg met metafoor als in de vorige les: "Het is een soort knijper, dat een getal vasthoudt. In een variabele kan maar één getal gelijker tijd. Als je er een nieuw getal in stopt, gaat de oude weg".
Daarnaast passen ze variabelen toe in Scratch.

Benodigdheden:

- Bord/plankje
- Knijpers
- Stift & Papier.

Herhaling:
Herhaal (met het pdf) “Een variabele is een soort houder, waar je een getal in kunt stoppen. Er past maar één getal gelijker tijd in. Als je er een nieuwe instopt, gaat de oude weg”.

Oefening 1, na uitleg van variabele in de les (P. 5):

“Stel dat deze knijpers variabelen zijn. Ik heb de variabelen vast een naam gegeven: 'Temperatuur'. Als we dan de temperatuur gaan meten, kunnen we de getallen in de variabele stoppen zodat we deze kunnen onthouden. Toen ik vanochtend de temperatuur ging meten, was het … graden buiten" (noem een reëel getal). "dat getal schijf ik op, en stop ik in mijn variabele". Schrijf het op een papier en knijp het vast aan het bord.

"Maar 's ochtends is het altijd een beetje kouder. Als ik nu de temperatuur zou meten, is het misschien wel … graden". (noem een reëel getal). Schrijf het nieuwe getal op, en klem het met een nieuwe knijper op het bord, onder de andere. “Deze waarde klem ik ook aan het bord, we werken van boven naar onder, net als met programmeren”.

"Nu hebben we dezelfde variabele, maar twee waardes. Dat is misschien een beetje verwarrend. Het zit zo: de twee variabelen heten precies hetzelfde, dus is het dezelfde variabele (en hebben dezelfde kleur knijper). We hebben nu dus twéé plekken en twéé waardes voor dézelfde variabele. Maar zoals we weten, kan er maar één getal gelijker tijd in een variabele. Als je er een nieuwe in stopt, gaat de oude weg. Schrijf maar op je wisbordje welk cijfer jij denkt dat de variabele nu heeft".

Controleer de antwoorden. Leg uit dat het onderste cijfer de juiste waarde is, omdat dit de nieuwe is en de oude ‘vergeten’ wordt. Doe dit ook wanneer de (meeste) kinderen het juist hebben. Als je merkt dat de kinderen het alsnog niet goed begrijpen, dan leg je het nogmaals uit.

Haal de knijpers en papieren van het bord. Ga verder met de les.

Oefening 2, Op het einde, na het laatste blok (P. 12):
Benodigdheden:

- Bord
- 2x wasknijper met sticker ‘Ik ben’ (beide kleur 1)
- 2x wasknijper met sticker ‘Ik word’ (beide kleur 2)
- Pen & Papier

“Er gebeurt iets geks in ons programma: we hebben geen getal in onze variabele gestopt, maar er komt er wel een getal uit. Laten we eens kijken hoe dat zit”.

“Ik heb weer vast twee variabelen gemaakt. De ene heet “ik ben” en staat voor de leeftijd die je aan het begin van groep … bent. De andere heet “Ik word” en staat voor de leeftijd die je in groep … word. Aan het begin van groep …. Waren de meesten van jullie (Groep 5: 8 – Groep 6: 9 – Groep 7: 10 – Groep 8: 11)”. Schrijf het op en klem het met de knijper “Ik ben” bovenaan het bord.

“Als je in groep … jarig bent, wordt je …. (getal “ik ben” + 1)”. Schrijf het op en klem het met knijper ‘Ik word’ onder de andere knijper aan het bord. Laat het bord duidelijk aan iedereen zien.

“Stel je bent eindelijk jarig. Dan klopt het getal in de variabele ‘Ik ben’ niet meer. We zouden er een nieuw getal in kunnen stoppen, maar de computer heeft daar ook een trucje voor. We willen hetzelfde getal als in variabele “Ik word” in variabele “ik ben” stoppen”. Wijs ondertussen de juiste knijpers op het bord aan. Wat we dan ook kunnen doen, is variabele ‘Ik word’ in variabele ‘Ik ben’ stoppen”. Knijp wasknijper ‘Ik word’ in wasknijper ‘Ik ben’ aan het bord “En wat er dan eigenlijk gebeurt, is dat het getal van ‘Ik word’ in ‘Ik ben’ gestopt wordt. Als je dit zo ziet, wat is dan nu het getal van ‘Ik ben’? Schijf maar op je wisbordje”.

Oefen nog een keer met andere getallen, bijvoorbeeld 10 en 20. Je hoeft niet eerst andere uitleg te geven.

Laat de code nu nog een keer zien, met andere getallen, leg ondertussen uit wat jullie net geoefend hebben.

Bijlage 8 Les 3

* De pagina lay-out wijkt af van het originele ontwerp i.v.m. ruimtebesparing.

Een gek verhaal

Lesdoel: In deze les mag je zelf (of samen) programmeren met Scratch! In deze les maak je een verhaal. Je geeft de computer dieren, plaatsen en dingen en de computer maakt er een gek verhaal van. Hier zie je bijvoorbeeld een kat die onder water loopt!

Eerst even zonder computer

In de vorige lessen hebben we gewerkt met variabelen, weet je nog? Het zit zo:

> Het is een soort knijper, dat een getal vasthoudt. In een variabele kan maar één getal gelijker tijd. Als je er een nieuw getal in stopt, gaat de oude weg.

In deze les gaan we ook variabelen gebruiken.

We gaan in deze les ook werken met lijsten, dat is nieuw. Lijsten zijn belangrijk voor computers om dingen in op de slaan. Maar, lijsten komen in het dagelijks leven ook veel voor.

Weet jij een paar leuke lijstjes? Denk bijvoorbeeld aan alle kinderen in je klas, al je vriendjes of vriendinnetjes, lekkere soort fruit of je favoriete spelletjes. Welke lijstjes kun jij nog meer bedenken?

In computers kunnen lijstjes veranderen, je kunt bijvoorbeeld dingen weghalen of erbij zetten. Kunnen jouw lijstjes ook veranderen?

Computers kunnen ook vertellen!

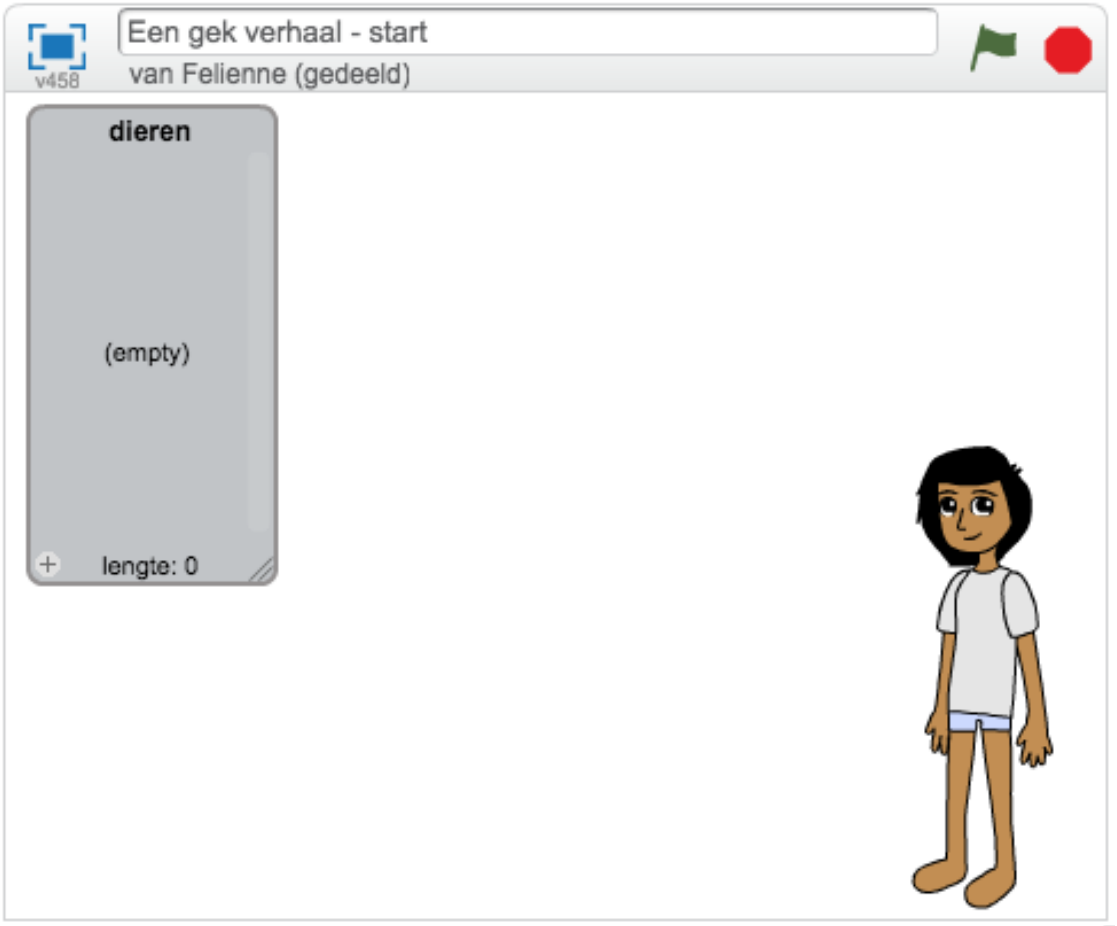
Computers kunnen eigenlijk alles, ook verhalen verzinnen. Maar dat moet jij wel eerst programmeren. Dat gaan we vandaag doen. We 'geven' de computer lijstjes en daaruit kiest de computer steeds een willekeurige optie. Net zoals juf of meester soms een willekeurig kind kiest om een taakje te doen.

We maken lijstjes, met dieren, plaatsen en activiteiten dan zetten we die samen in elkaar. Dan krijg je gekke leuke zinnen! 'Loopt jouw groene pad in het bos?' Of 'graaft jouw hagedis op school?'.

We gaan het nu zien.

Ga naar https://scratch.mit.edu/projects/188964122

Daar staat dit al voor je klaar:



Dani, het poppetje rechts gaat het verhaal voorlezen.

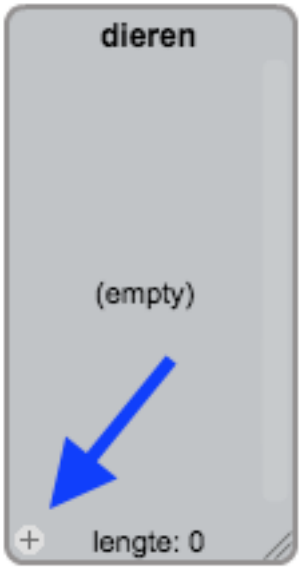
Je ziet ook een grote grijs vlak in beeld: dieren. Dat is een lijst die wij straks gaan vullen.

Dani gaat zich eerst even voorstellen. Laat haar 2 seconden "Hallo" zeggen, of een andere leuke groet. Daarna zegt ze: "Ik kies een dier en een ding en een plek:" Zo weten we wat er gaat gebeuren in het programma. Programmeer jij het zelf?

Zo ziet het eruit:

ScratchTip: aan de kleuren van de blokjes in het voorbeeld kun je zien onder welk kopje in Scratch je de blokjes kunt vinden, maar misschien wist je al waar ze stonden!

Nu is het tijd om aan de dieren te werken. Kies jij een paar leuke dieren uit die aan het verhaal mee mogen doen? Je voegt ze toe in het grijze vlak. Klik eerst op het plusje bij de blauwe pijl:



En typ dan in het oranje vakje. Je kunt dat een paar keer doen. Kies zoveel dieren als je wilt. Ik heb er vier gekozen:

Tijd om te programmeren! Dani gaat een dier kiezen.

Dat doet ze met dit blokje:

Leg het blokje in het grijze veld en klik er maar eens een paar keer op. Wat gebeurt er nu?

Dat blokje kiest altijd het eerste dier.

We willen steeds een ander dier! Misschien weet je nog dat we in de eerst les de stappen en graden van het potlood willekeurig hebben gekozen. Willekeurig betekent dat we het niet kunnen voorspellen, steeds weer anders.

Dat zit ook op een lijst. Klik maar eens op het kleine driehoekje naast de 1 en maak er willekeurig van:

Nu krijg je steeds een ander dier!

Plak het blok in een denk blok en plak het onder wat je al had:

Let op: Ik heb de zeg en denk blokken wel op 1 sec gezet anders moet je wel heel lang wachten!

Dingen en plekken

Zo, nu jij! Kun jij nu de dingen om te doen en de plekken invullen en programmeren?

Ga naar

. Zet nu de vinkjes aan voor 'dingen om te doen' en 'plekken':

Nu komen er twee extra grijzen vlakken in beeld:

Zet nu dingen om te doen in de tweede lijst, en plekken in de derde. Laat Dani daarna iets willekeurigs kiezen. Dat doe je door denkblokken te gebruiken, zoals we net ook hebben gedaan.

Let op: het moet wel mooi in een zin passen! Bij dingen om te doen moet je dus zetten:

voetbalt of loopt of kijkt of danst

Lukt het om iets te programmeren? Als het niet lukt, spiek dan op de volgende pagina.

Dit zijn mijn lijstjes en mijn programma:

Probeer je programma nu uit! Wat was de leukste zin die langskwam?

De mijne was: kat denkt op het strand. Wie denkt er nu op het strand? Daar moet je lekker zonnen.

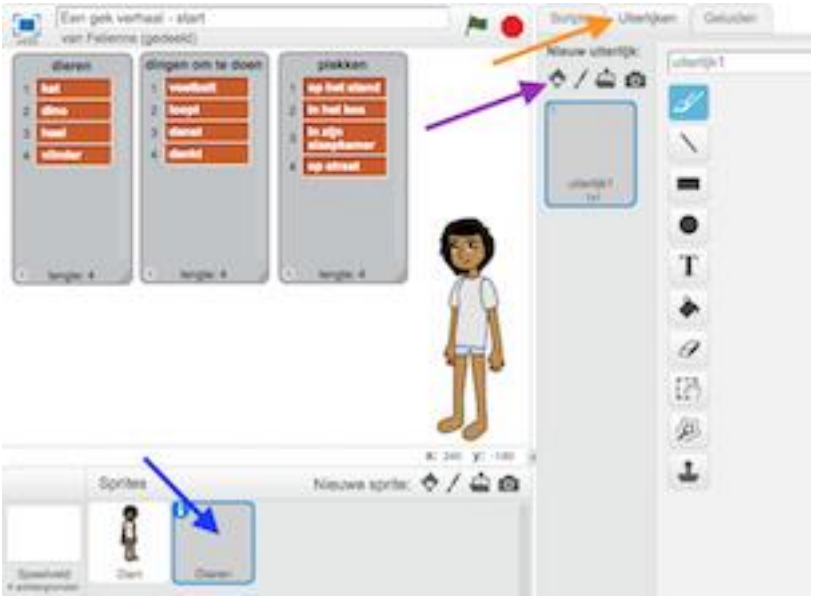
Wat is jouw leukste zin?

Breng je verhaal tot leven

Een verhaal vertellen is leuk, maar uitvoeren is nog leuker!

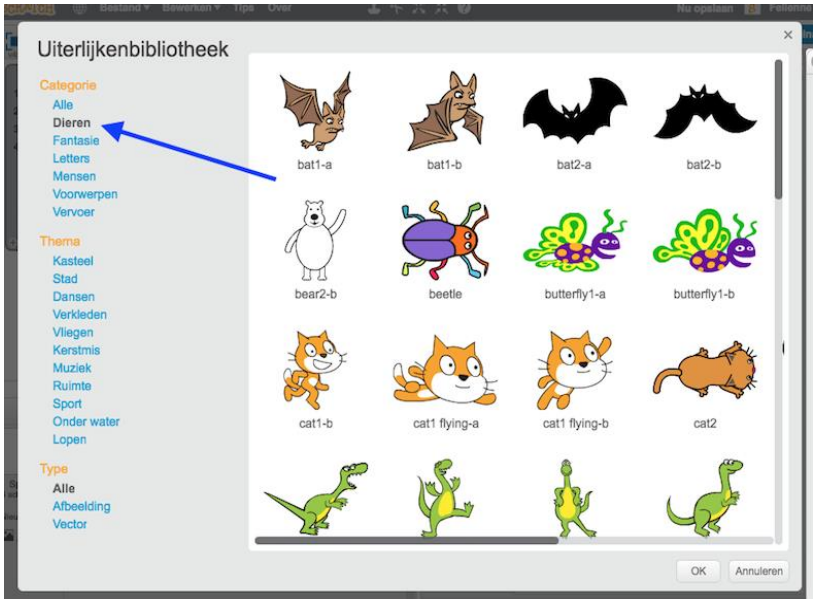
We gaan nu programmeren dat het verhaal echt tot leven komt op je computer. We beginnen met de dieren. Er staat al een sprite Dieren voor je klaar, maar daar zitten nog geen uiterlijken in!

Kies de sprite en kies dan Uiterlijken bovenaan. Kijk goed naar de pijlen. Kies eerst de blauwe en dan de oranje.



Nu gaan we alle dieren uit jouw lijst in de sprite zetten. Klik op kies uiterlijk uit de bibliotheek bij de paarse pijl.

Je krijgt nu alle uiterlijken te zien. Het is handig om op dieren te klikken links (bij de blauwe pijl) dan zie je alleen de dieren.



Let op! Let op!

Let goed op, wat nu komt is heel belangrijk! Je moet de uiterlijken van de dieren nu de namen geven **precies** zoals ze in jouw lijst staan. Namen veranderen kan je bovenaan het scherm (bij de blauwe pijl).

Verander het dier

Nu gaan we het dier ook echt van uiterlijk laten veranderen. Daarvoor gaan we het dier dat Dani kiest in de computer opslaan.

Welk Scratchblokje zouden we daarvoor nodig hebben denk jij?

- variabele
- als-dan-anders
- lijst

En, we moeten het dier ook een seintje geven. Want Dani kiest het dier uit maar het dier moet het zelf wel weten.

Welk Scratchblokje zouden we nu nodig hebben denk jij?

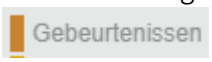
- als-dan-anders
- signaal
- herhaal

De antwoorden staan op de volgende pagina.

De goede antwoorden op de vragen zijn: variabele en signaal. Met een variabele slaan we iets op en met een signaal geven we iets door.

Sla het dier op in een variabele, en stuur een signaal, zo:

Je moet het signaal zelf maken. Het blokje

staat bij 

. Let op, je moet nog wel zelf een signaal schrijven. Klik op het kleine driehoekje en op

.

Als het signaal gemaakt is, moeten de dieren nog reageren. Ga naar de sprite met de Dieren.

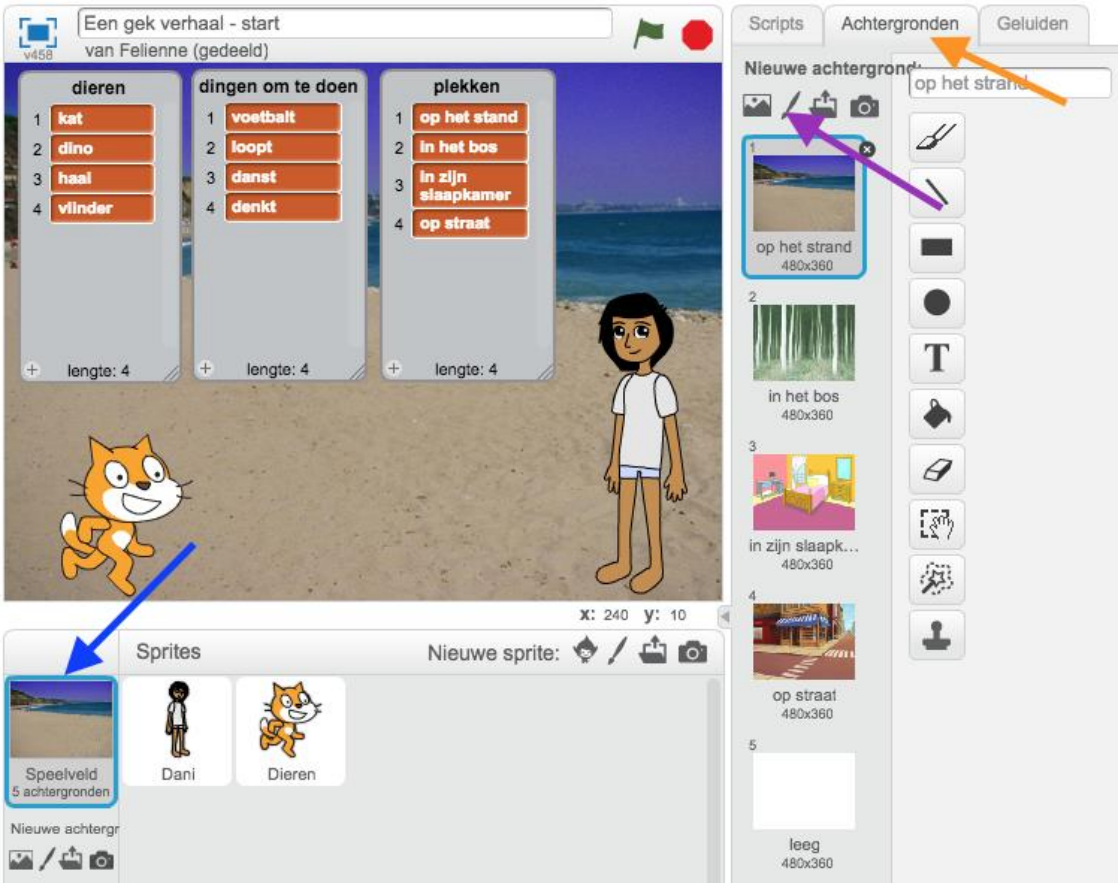
Je hoeft nu alleen maar de variabele in een verander uiterlijk te stoppen:

Dat werkt omdat we de namen zo precies hebben overgeschreven in de uiterlijken.

Nu de achtergrond

We gaan nu hetzelfde doen bij de plekken. Daarvoor veranderen we de achtergronden. De dingen om te doen, die komen hierna.

Een speelveld kan ook uiterlijken hebben, die heten achtergronden. Kies het speelveld bij de blauwe pijl en kies dan Achtergronden, dat zit op dezelfde plek als Uiterlijken bij de oranje pijl:



Voeg ook voor de achtergronden plaatjes toe. Zorg ook weer dat de namen kloppen. Als je een achtergrond niet kunt vinden, kies dan een andere plek in de lijst. Je kunt een woord in de lijst 'plekken' veranderen door op de oranje vakjes te klikken. Je kunt natuurlijk ook een plaatje kiezen dat op je woord lijkt.

Nu moet je dezelfde stappen nemen als bij de dieren, dus:

1. Stop de willekeurig gekozen plek in een variabele
2. Verander het denk blok zodat het de variabele zegt
3. Stuur een nieuw signaal, zoals "achtergrond verander maar"
4. Laat de achtergrond reageren op het signaal

Kun jij het maken?

Als het niet lukt, spiek dan op de volgende pagina.

Zo dat gaat al heel goed! Het is nog mooier als de lijstjes niet meer in beeld staan he? Klik dieren en plekken maar uit:

Nu is het natuurlijk nog leuker als de dieren echt doen wat er in het verhaal staat. Voor iedere activiteit gaan we iets anders doen. De sprites gaan nu echt bewegen.
Welk blokje heb je daarvoor nodig, denk jij?

- Als-dan
- Variabele
- HerhaalJa, dat gaan we met een als-dan doen!

Ga eerst naar de code van Dani, en stuur een signaal naar het dier. Dan weet het dier dat het wat mag gaan doen:

Ga nu weer naar de code van de dieren. Kies een van je dingen om te doen, bijvoorbeeld 'loopt' en stop die in een als blok, met een =. De = staat bij Functies:

Ook nu moeten de woorden precies hetzelfde zijn! Nu moet jij echt creatief worden! Hoe ziet lopen eruit? Of dansen? Of voetballen?

Kies leuke beweegblokken en stop ze in de als:

Het is wel handig als je het dier 'reset' aan het begin, zodat het altijd op dezelfde plek en stand staat. Leg daarvoor deze blokken bij het dier:

Maak jij het verhaal helemaal af? Als je meerdere dingen wilt doen, plak je de alsen onder elkaar, zo:

Zet 'm op met jouw gekke verhaal!

Bijlage 9 Experiment Les 3

Les 3, Een gek verhaal (nader uitwerken)

In deze les worden telkens verschillende waarden uit een lijst, in een variabele gestopt (p. 19). Dat geeft de leerlingen wellicht de indruk dat een variabele meerdere waarden tegelijkertijd vast kan houden. Daarnaast vergroten variabelen met strings de kans op het misconcept (paper 'Thinking out of the box'). Daarom wordt er in deze les geoefend met strings als waarde van variabelen. Het begrip van variabelen wordt hier verbreed.

Uitwerking:

"In de vorige twee lessen hebben we met variabelen gewerkt, weten jullie nog? Het zat zo: Een variabele is een soort knijper, dat een getal vasthoudt. In een variabele kan maar één getal gelijker tijd. Als je er een nieuw getal in stopt, gaat de oude weg".

"Tot nu toe hebben we telkens getallen in de variabelen gestopt. Maar wat ik nog niet verteld heb, is dat je ook woorden en zinnen in een variabele kunt stoppen. Dat is misschien een beetje verwarrend, omdat er maar één waarde in een variabele past. Denk maar eens zo: 1 woord of 1 zin is 1 waarde. Laten we het eens proberen".

Oefening:

Pak een wasknijper met sticker met 'Dier'. Stop er een briefje met een getal in. "Dit is variabele 'Dier' met een getal erin, zoals we het de vorige lessen geoefend hebben. Laten we er een woord in stoppen, een dier". Laat een van de kinderen een dier noemen en schrijf deze op.

Doe de wasknijper open, laat het oude papier op de grond vallen en stop de nieuwe erin. Vertel ondertussen "Ik stop een nieuwe waarde in de variabele, dus de oude valt eruit. Nu hebben we een woord in onze variabele".

"Nu staat er alleen nog maar …. (dier) in onze variabele. Maar ik wil het dier ook graag een naam geven, dan moet ik een woord toevoegen". Verzin een naam voor het dier (bijvoorbeeld Kat Karel, Hond Max of Konijn Pluisje). Schrijf alleen de naam (Karel/Max/Pluisje) op een papier. "Maar let goed op! Als ik nu alleen de naam opschrijf en ik wil deze in mijn variabele stoppen, dan valt het dier eruit". Doe het voor. Houdt de knijper met het dier omhoog, doe hem open om de naam erin te stoppen, en laat het briefje met het dier eruit vallen.
"Als ik dus de zin in mijn variabele wil veranderen, moet ik de hele zin opnieuw schrijven en erin stoppen."

"Laten we nog een voorbeeld oefenen". Schijf op een papier een dier met een naam, bijvoorbeeld: Kat Karel. "In mijn variabele zit de zin: 'Kat Karel'. Ik wil Karel iets laten doen, ik wil hem laten lopen. Dan moet ik het woord 'Loopt' toevoegen aan de zin in de variabele. Schrijf maar op je wisbordje op, wat ik op mijn nieuwe papier moet opschrijven om de zin 'Kat Karel Loopt' in mijn variabele te stoppen". Herhaal wat de kinderen moeten doen.

Laat de kinderen hun antwoorden op de wisbordjes schrijven. Vraag één kind zijn/haar antwoord toe te lichten. Geeft het kind een goed antwoord, herhaal dan alsnog. Geeft het kind een onvolledig antwoord, vul dan aan door het juiste antwoord te herhalen.

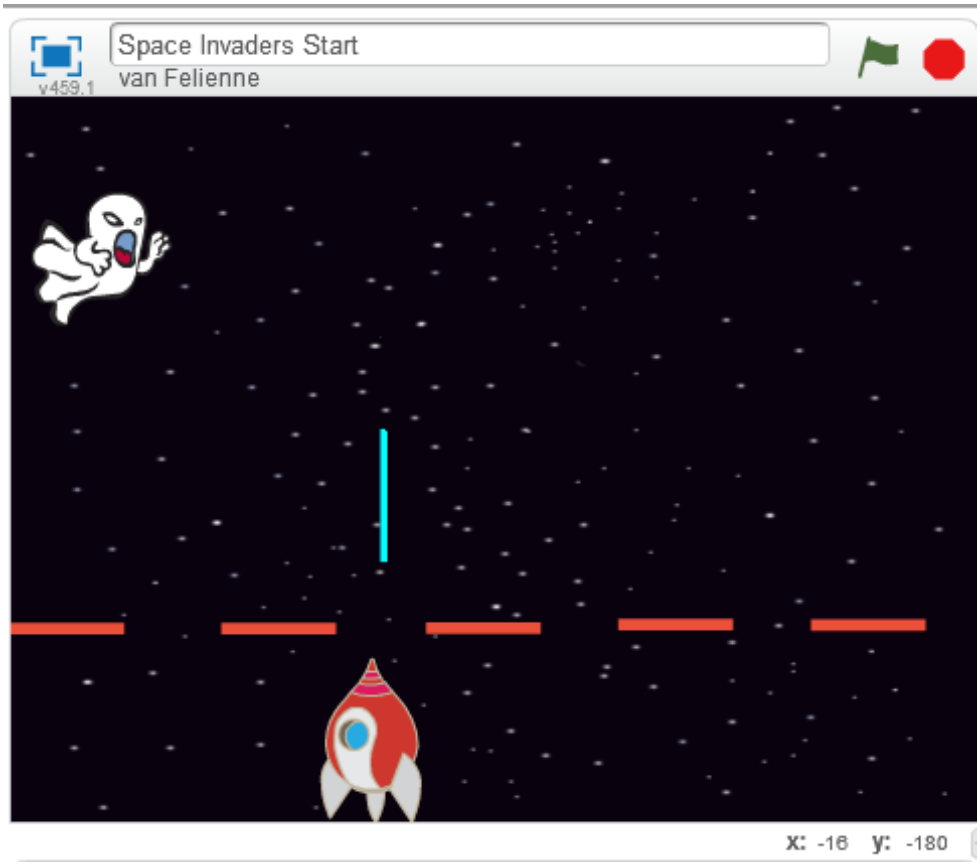
Bijlage 10 Les 4

* De pagina lay-out wijkt af van het originele ontwerp i.v.m. ruimtebesparing.

Scratch les 4: Scratch Invaders

Het ruimteschip

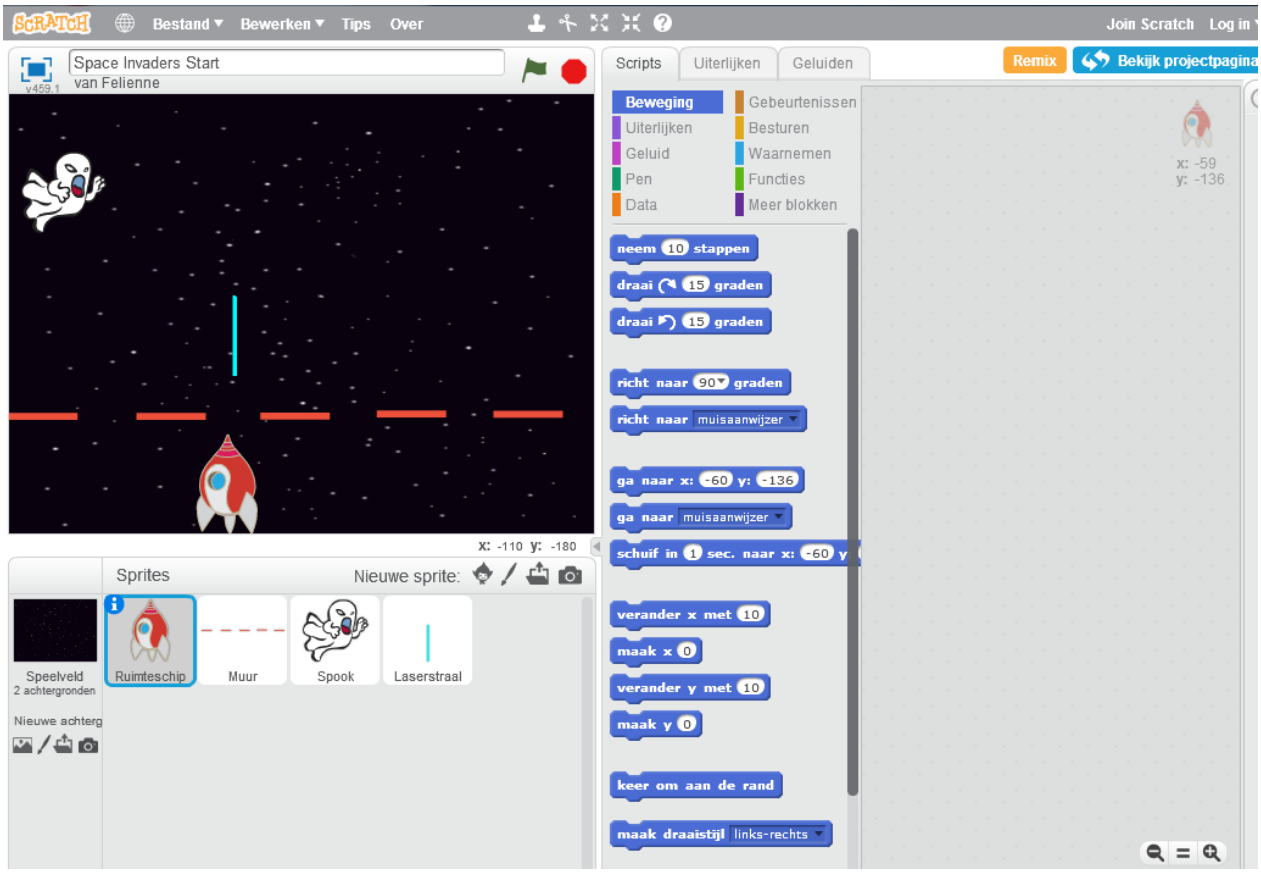
Het ruimteschip is in gevaar! Er komen monsters aan, en die komen steeds dichterbij. Kijk!:

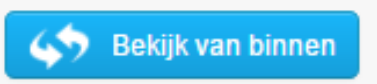


Dit materiaal is gemaakt door Felienne. en bewerkt door Emily Bakker. Het is Creative Commons by-nc-sa-4.0
Simpel gezegd: Je mag het gebruiken in je lessen, aanpassen, uitprinten, kopieren, wat je maar wilt.

Maar: Je moet mijn naam erbij zetten, je mag er geen geld mee verdienen en als je het aanpast, moet je dat ook weer Creative Commons maken. Het linkje bevat alle informatie. Meer info over coding les geven in de klas? Ga naar http://codeuur.nl

Ga naar: https://scratch.mit.edu/projects/133611163.
Daar staat dit voor je klaar:



Als je daar bent, klik dan op 

.

Het ruimteschip besturen

Als het spel straks af is, gaan de monsters jouw ruimteschip aanvallen! Jij kunt schieten en ze proberen te raken. Maar let op, je kunt niet door de rode muur schieten!

Eerst weer een ‘aan-knop’ voor ons programma.

Pak het blok

. Kies nu

.

DDD

Nu moet het ruimteschip nog iets doen als je het pijltje indrukt. Hij moet natuurlijk naar links gaan. Ga naar

. Welk blok moeten we dan gebruiken?

Of

Of

Op de volgende pagina staat het juiste antwoord.

Het blok dat we nodig hebben is:

Weet je nog hoe het zat met de X-as en de Y-as in de alle eerste les? Met de X-as bewogen we naar rechts en naar links, met de Y-as naar boven en naar onder.

Waarschijnlijk heb je nu hetzelfde als ik:

Werkt dat al goed? Nee he? Er moet een minnetje voor de 10! Dat moet je doen door in het rondje te klikken en een minnetje te typen (meestal zit die naast de 0 op je toetsenbord)
Zo:

Kun jij nu ook het pijltje naar rechts programmeren? Probeer het maar! ps. Als het niet lukt, kun je op de volgende pagina spieken.

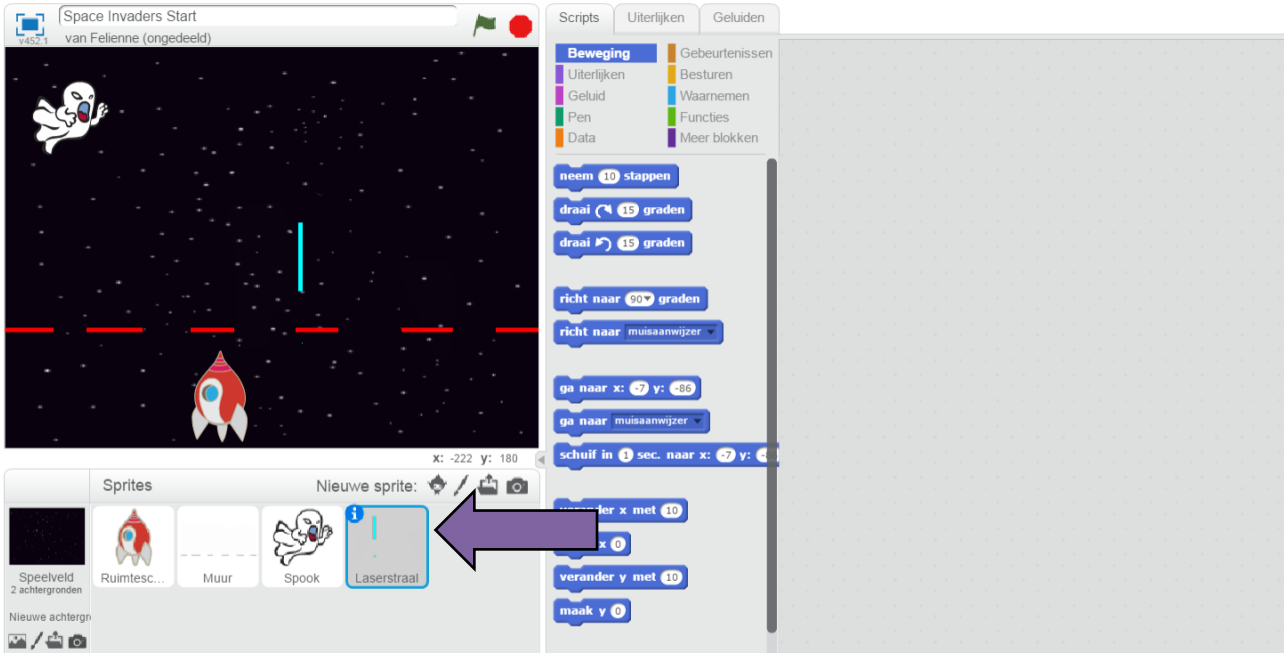
Voordat je verder kunt, moet jij dezelfde code hebben:

Wat gebeurt er nu als je op de pijltjes op je toetsenbord drukt?

Schieten!

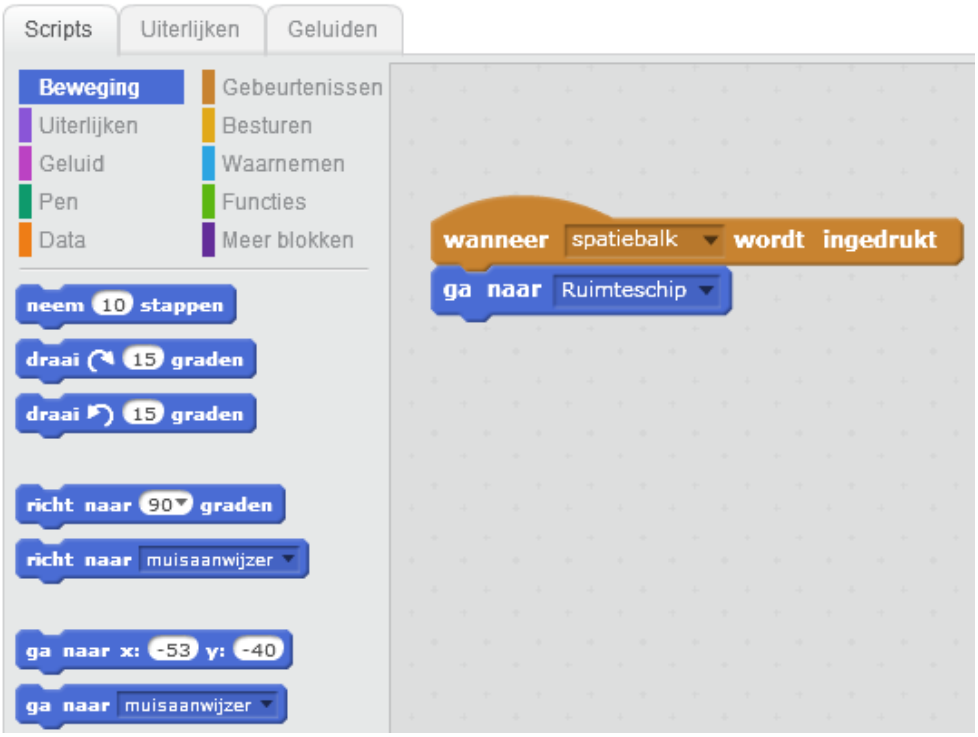
Het ruimteschip moet natuurlijk kunnen schieten. We willen dat als je op de spatiebalk drukt, er een laserstraat verschijnt.
Die moet dan ook nog naar boven gaan.
Als hij boven is, moet hij weer verdwijnen.

Dat gaan we programmeren bij de Laserstraal. Klik er maar eens op:



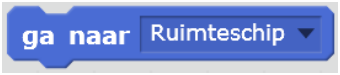
Niet schrikken als je blokjes weg zijn hoor!
Die staan nog veilig bij het ruimteschip.

Maak nu deze blokjes maar eens:



Dat is al best mooi he? De straal komt naar het schip toe. Maar hij moet er juist van weg, anders kunnen we niet schieten. Daar pakken we een blok voor dat we al eens eerder hebben gebruikt:

.

Plak die onder 

. In het

blok moet nu nog wel een blokje, anders gebeurt er niks. De pijl moet telkens omhoog, dus gebruiken we

.

Probeer je spel nu maar eens uit!

Werkt het? Kun je schieten?

Kom terug straaltje!

Dat werkt wel, alleen helaas maar een keer.

Als je aan de rand bent, moet je laserstraal weggaan. Dat doen we zo. Verander eerst de

in een

.

In het gat dat je nu hebt, stop je een

.

Die vind je bij Waarnemen.

Verander het nu in

. Zo dus:

Nu moet de straal nog weggaan. Daarvoor gebruiken we een

. Die zit bij Uiterlijken. Waar zou die moeten denk jij?

En er moet ook nog ergens een

. Anders is de laserstraal voor altijd verdwenen.

Probeer het maar eens!

Op de volgende bladzijde vind je het goede programma.

.

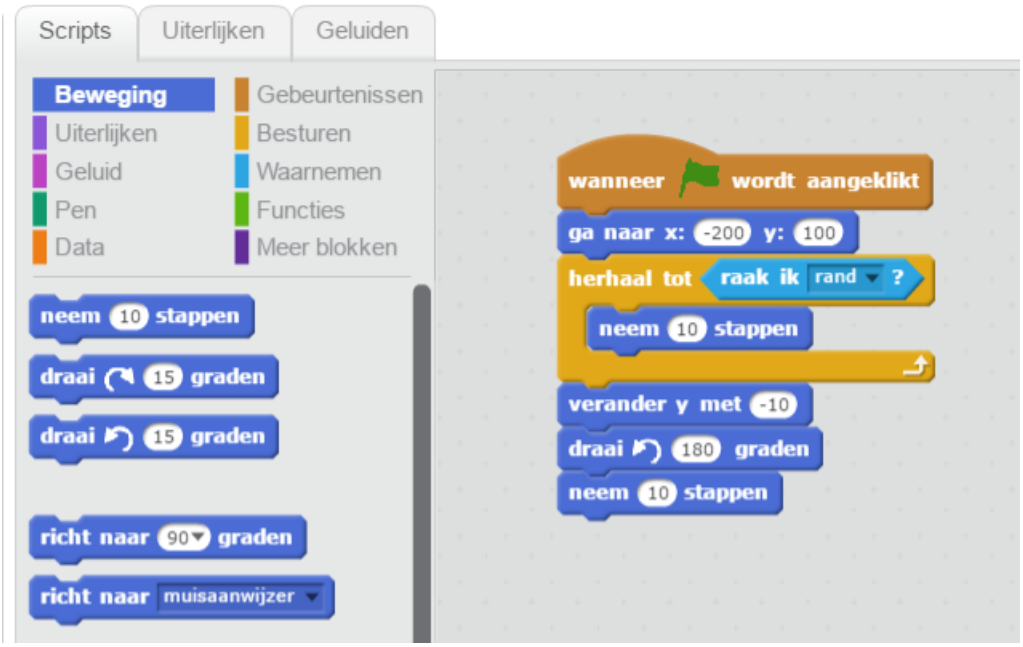
En, werkt jouw programma?

Het monster

Het spook moet opzij bewegen. Als hij tegen de rand komt, gaat hij een stukje naar beneden. Steeds dichterbij. Dat moeten we dus ook programmeren!

Je mag ook een ander poppetje kiezen voor het spook!

Maak deze blokjes maar eens.

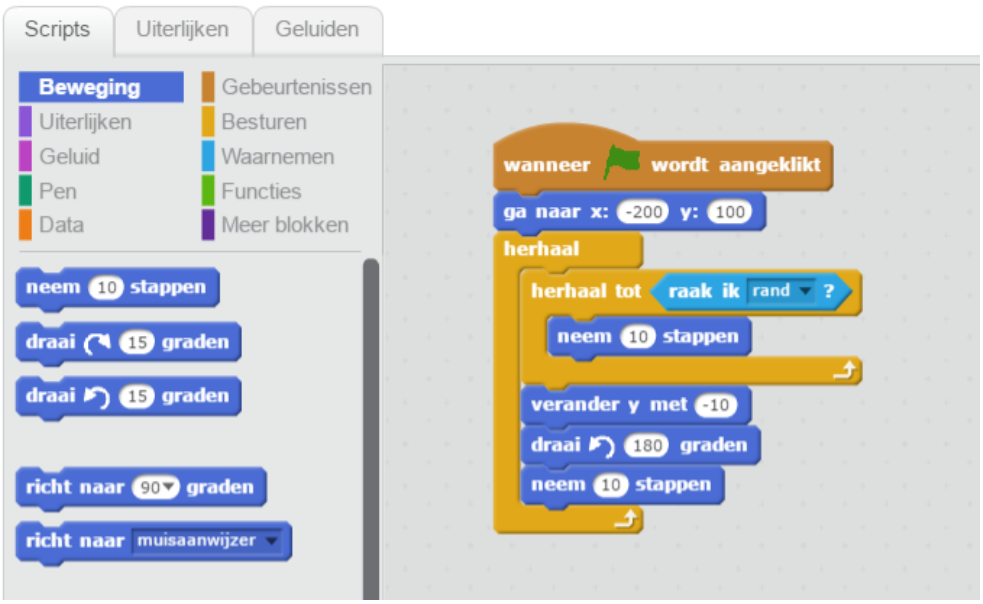


Wat gebeurt er nu als je op de groen vlag klikt? Werkt het al goed?

Niet helemaal he? Er moet nog een blokje omheen.

Weet jij welk blokje?

Ja, kijk, zo is het beter:



Dat enge spook komt op ons af!

Als je nu op het groene vlaggetje klikt, wordt het spook op de goede plek gezet, begint hij te bewegen en komt hij steeds dichterbij. Yèk!

Helaas gebeurt er nog niks als we raak schieten…

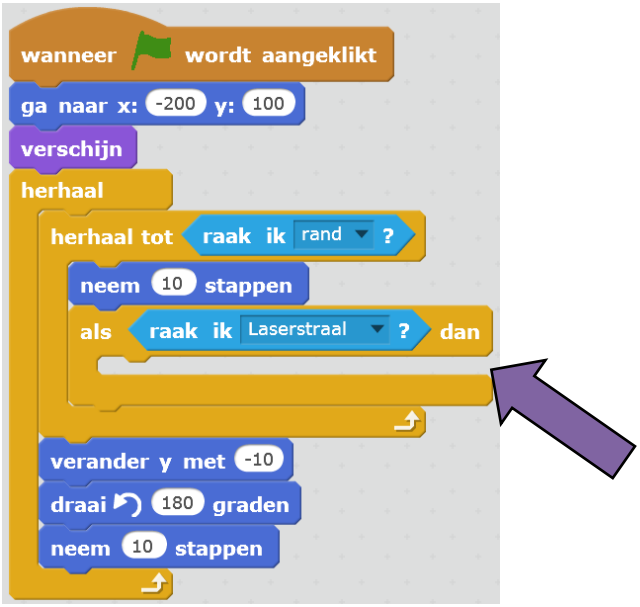
Dat gaan we nu maken, met een

.

Zo:

Ik heb de

even op 2 gezet zodat je makkelijker raak schiet.



Weet jij wat er nu bij de pijl moet komen? Probeer het maar eens! Ik heb het zo:

Game Over

Als het spook bij het ruimteschip is gekomen, is het spel voorbij!

Zet deze blokjes bij het ruimteschip:

We hebben al een best leuk spel maar…. We kunnen nog veel verder! Kijk gauw op de volgende pagina.

De muur maakt het moeilijker

Het is nog wel wat te makkelijk misschien he?

We gaan zorgen dat de muur in de weg zit. De laserstraal moet stoppen als hij de rand of de muur aanraakt. Daar is een blokje voor!

Kijk bij

en pak de

. Zet die in het

.

Zo:

We laten de muur ook nog eens bewegen, dan wordt het nog moeilijker. Geef de muur dit script:

Punten scoren

We hebben al een super gave game! Maar we kunnen ook nog de score bij houden, iedere keer dat we het spook raken, krijgen we een punt. Dat doe je met variabelen, weetje nog? Het zat zo:

Het is een soort knijper, dat een getal vasthoudt. In een variabele past maar één getal gelijker tijd. Als je er een nieuw getal in stopt, gaat de oude eruit.

Weet jij nog een je een variabele moet maken? Het moet bij

. Nu kun je er een maken. Ik heb die van mij

genoemd.

We gaan programmeren bij het Spook. Iedere keer dat we het Spook raken, krijg je punt. De variabele

moeten dan dus veranderen. Gebruik daarvoor dit blok:

.

Weet jij waar het blok moet staan? Kijk maar gauw op de volgende pagina:

.

Nu kun je kiezen hoe veel punten je krijgt als je het spook raakt. Gewoon 1? Of misschien wel 5 of 10?

Nu zijn we echt heel ver! We gaan nog één ding doen: een reset-button maken. Dan kun jij het aantal punten resetten, wanneer je wil!

Maak een nieuwe sprite, welke afbeelding wordt jouw reset-button? Die van mij ziet er zo uit:

.

Maak deze code:

.

En nu… Gamen maar!

MMM

Bijlage 11 Experiment 4

* De pagina lay-out wijkt af van het originele ontwerp i.v.m. ruimtebesparing.

Experiment Les 4, ScratchInvaders

Dit is de laatste les in Scratch die de kinderen van de TU Delft krijgen. In de vorige drie lessen hebben de kinderen uit de experimentele groep elke keer uitleg gehad over variabelen, gevolgd door een oefening. Dit maal zullen zij alleen uitleg krijgen. Ze hebben al drie keer op verschillende wijzen geoefend, dus daar kan altijd naar gerefereerd worden. Daarnaast moet de les zo min mogelijk tijd in beslag nemen, omdat de leerlingen ook nog 20 minuten bezig zullen gaan met het invullen van de vragenlijst.

Benodigdheden:

- Het pdf op groot scherm.

Uitwerking op de les

De kinderen gaan zelfstandig met de les aan de slag. Pas helemaal op het einde van de les komen variabelen aan bod (p. 15). Om ervoor te zorgen dat de kinderen dit op de juiste wijze nogmaals doornemen, wordt dit klassikaal aangepakt.

Start de les zoals gewoonlijk met een korte samenvatting en de introductie van de opdracht van vandaag. Zorg ervoor dat in die herhaling variabelen worden genoemd. Scroll in het pdf door naar pagina 15. Hier staat de uitleg van variabelen. Vertel dat als de kinderen snel kunnen werken, zij vandaag ook variabelen gaan maken. En dat je nu vast gezamenlijk nog even kijkt hoe het ook alweer zat.

Lees de definitie van variabelen klassikaal hardop voor. Lees daarna gezamenlijk met de leerlingen de definitie nog een keer voor.
Refereer daarna terug naar de voorbeelden uit de vorige lessen:
Les 1: We stopten een getal in de knijper, als we er een nieuwe in deden, viel de oude eruit.
Les 2: Hier hebben we twee dingen gedaan. Eerst lieten we een variabele twee keer zien. Ondanks dat één variabele twee keer gemaakt was, paste er maar één waarde in. Dat was altijd de laatste die we toevoegden.
We hebben ook nog een variabele in een variabele gestopt. Als we dat doen, gaat de waarde altijd mee.
Les 3: Toen hebben we woorden en zinnen in een variabele gestopt. Als we een woord wouden toevoegen, moesten we alles opnieuw opschrijven en in 1 keer veranderen.

Nu gaan we verder met de les.

Bijlage 12 Procedure Vragenlijstafname

Procedure vragenlijst afname

Voor het invullen van de vragenlijst moeten de kinderen zeker 20 minuten de tijd krijgen. Begin dus ruim 25 minuten voor het einde van de les met alles klaarzetten.

1. Stoppen met de les
Begin hiermee dus ruim 25 minuten voor het einde van de les. Vertel de kinderen dat ze Scratch afsluiten. De meesten hebben het spel nog niet af, maar dat is niet erg. Ze het via scratchles.nl afmaken. De kinderen die een account hebben, kunnen het project opslaan.

2. De link naar de vragenlijst
Laat alle kinderen naar de website link.shareyourwork.nl gaan. Deze staat in de werkbalk favorieten als 'Onderzoek Emily' (o.i.d.). Zeg dat ze heel goed moeten luisteren voor wat ze nu gaan doen. Dadelijk noem je een lijst namen op. Dit zijn de namen van de kinderen die wel een toestemmingsformulier hebben ingeleverd.

Vertel dat als je vraag wel wordt opgenoemd, je op cijfer 1 klikt. Als je naam niet wordt opgenoemd, klik je op cijfer 5.

3. Uitleg alternatieve opdracht
De kinderen die de vragenlijst niet invullen maken een alternatieve opdracht. Zij gaan niet verder met Scratch, een spelletje spelen, of iets voor henzelf doen, want dat leidt de andere kinderen te veel af en zorgt er wellicht voor dat de andere kinderen de vragenlijst gaan afraffelen.

De alternatieve opdracht staat achter knop 5 op de site. Dit is een pdf/word bestandje dat ze zelfstandig kunnen maken. Als ze klaar zijn mogen ze iets lezen, of oefeningen voor taal of rekenen op de computer maken, maar ze gaan niet gamen, filmpjes kijken of andere dingen doen.

Laat ze aan de slag gaan. Ze mogen niet overleggen of praten.

4. Uitleg vragenlijst
Tijdens de uitleg en het maken van de vragenlijst, is het helemaal stil in het lokaal. Benoem van te voren dat kinderen die niet stil zijn, de computer uit moeten zetten en buiten het lokaal gaan wachten.

Lees de instructie van de vragenlijst voor, en laat de kinderen op hun scherm meelezen.
"Dit was alweer de laatste programmeerles van de TU Delft. Samen hebben we een kunstwerk, een rekenmachine, een verhaal en een game geprogrammeerd. Weet je nog?

- afbeeldingen van de lessen –

We hopen dat je het leuk vond en dat je veel hebt geleerd! Maar wij kunnen ook nog veel van jou leren. Je helpt ons door de vragen te maken. Het zijn er wel meer dan 20, probeer er zo veel mogelijk te maken!

Er zijn geen goede of foute antwoorden bij deze vragen, het gaat om hoe jij over de vragen denkt. Denk niet te lang na over een antwoord, vul maar in wat er in je opkomt. Als je het antwoord écht niet weet, kies je het minst slechte antwoord.

De antwoorden die je geeft zijn anoniem: dat betekent dat wij niet weten welke antwoorden van jou zijn. De eerste vragen vullen we samen in:

Klik op het pijltje rechts onderin."

"Bij de eerste vraag klik je op 'De Nicolaasschool, Rotterdam"
"Bij de tweede vraag klik je op (groep) (6) "
"Bij de derde vraag klik je op *les1: 'C' *Les 2: 'A' * les3: 'B'".

"Klik maar weer rechts onderin op de pijl. Nu maken we 1 oefenvraag. Let goed op, want alle vragen lijken op deze".

Lees de vraag voor:
'Kijk goed naar de blokken. Wat zal Karel zeggen en doen als we op de groene vlag klikken'?.
'We stoppen 'Hoi' in een variabele, dan zetten we 10 stappen en dan noemen we de variabele. Klik het antwoord aan dat jij denkt dat erbij hoort. Daarna ga je zelf verder'.

Algemeen:
Alle kinderen sluiten Scratch af.
Geef de kinderen met de alternatieve opdracht eventueel een a4 om hun antwoorden op te schrijven.

!! Het is helemaal stil tijdens de uitleg en het invullen van de vragenlijst. Kinderen die niet stil zijn, sluit hun computer af en wachten op de gang.

Leidt het toch af dat de kinderen met de alternatieve opdracht vast aan de slag zijn, laat ze bij de andere groepen dan wachten tot de uitleg van de vragenlijst klaar is.

Bijlage 13 Vragenlijst

* De pagina lay-out wijkt af van het originele ontwerp i.v.m. ruimtebesparing.
** In deze versie staat in rode letters per antwoord het bijbehorende misconcept genoteerd.

Instructie

* Klassikaal door te nemen en in te vullen

Dit was alweer de laatste programmeerles van de TU Delft, samen hebben we een kunstwerk, een rekenmachine, een verhaal en een game geprogrammeerd. Weet je nog?

- Afbeeldingen van lessen -

We hopen dat je het leuk vond en dat je veel hebt geleerd! Maar wij kunnen ook nog heel veel van jou leren. Wij hebben een aantal vragen over programmeren gemaakt. Het zijn er meer dan 20, dus let goed op de tijd. Er zijn geen goede of foute antwoorden, het gaat erom hoe jij over de vragen denkt. Als je het antwoord écht niet weet, kies je het minst slechte antwoord. Dadelijk doen we een oefenvraag.
De antwoorden die je geeft zijn anoniem: dat betekent dat wij niet weten welke antwoorden van jou zijn. Denk bij het invullen van de vragen niet te lang na over je antwoord, vul maar in wat er in je opkomt.
De eerste vragen vullen we samen in:

School

Op welke school zit je?

• De Nicolaasschool in Rotterdam •Het Blokhuus in Hoevelaken

Groep *Klas*

Ik zit in groep…

Groep	Klas
•5	•A
•6	•B
•7	•C
•8	•D
	•Combinatie 6/7

Oefenvraag:

Kijk goed naar de blokken. Wat zal Karel zeggen wanneer we op de groene vlag klikken?

•Neemt 10 stappen,

Correct

•

, neemt 10 stappen Fout

• Neemt 10 stappen,

Fout

•Neemt 10 stappen,
Fout

Leeftijd

4. Hoe oud ben je?

•8 •9 •10 •11 •12 •13

Geslacht

5. Ik ben een…

•Jongen •Meisje •Zeg ik liever niet

Aanwezigheid

6. Bij hoe veel Scratchlessen van de TU Delft was je aanwezig?

•1 •2 •3 •4

Lessen

6A. Welke lessen heb je gehad? *MC*

•Les 1 Spirograaf •Les 2 Tafels •Les 3 Een gek verhaal •Les 4 Doolhof

Ervaring

7. Ik had voor de programmeerlessen op school ook al eens geprogrammeerd

•Ja •Nee

Ervaring_bij

7A. Ik heb eerder geprogrammeerd…. *MC*

•op school •thuis •met vriendjes/vriendinnetjes •*open*

Ervaring_met

7B. Ik heb geprogrammeerd met…. *MC*

•Scratch •Lego Mindstorms •Codeuur •*open*

Testvraag1

15. Kijk goed naar de blokken. Wat zal Karel zeggen wanneer we op de groene vlag klikken?

-

Correct
-

Fout

Testvraag2

16. Kijk goed naar de blokken. Wat zal Karel zeggen wanneer we op de groene vlag klikken?

-

Fout
-

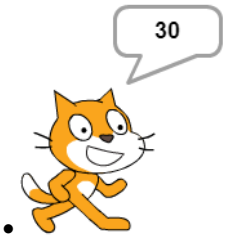
Correct

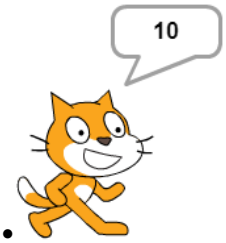
P1

Kijk goed naar de blokken. Wat zal Karel zeggen wanneer we op de groene vlag klikken?

-

correct
-

multiple
- 

optellen
- 

4

P2

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

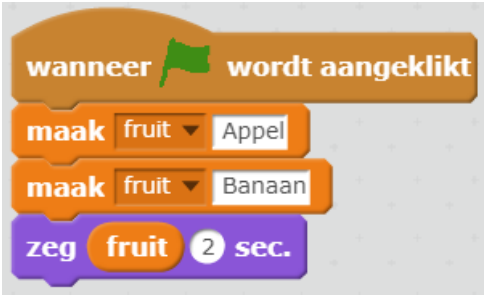
-

correct
-

logisch
-
klopt
- Karel zegt een willekeurig groot getal aangepast

P3

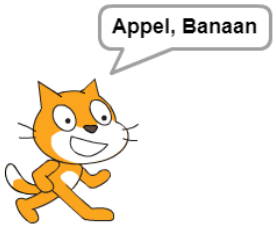
Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?



-

correct
-

volgorde

•1 

alles

-

en dan

dubbel

P4

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

-

correct
-

en dan

interpreteren

UUU

-

en dan

omzetten

P5

Kijk goed naar de blokken. Wat zullen de bananen zeggen als we op de groene vlag klikken?

-

correct
-

werkelijkheid
-

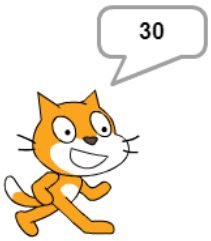
toegepast
-
overslaan

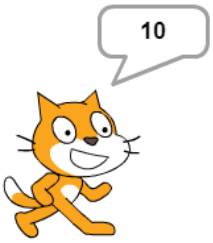
P6

Kijk goed naar de blokken. Wat zal Karel zeggen wanneer we op de groene vlag klikken? *Variabelen en warden*.

-

correct
-

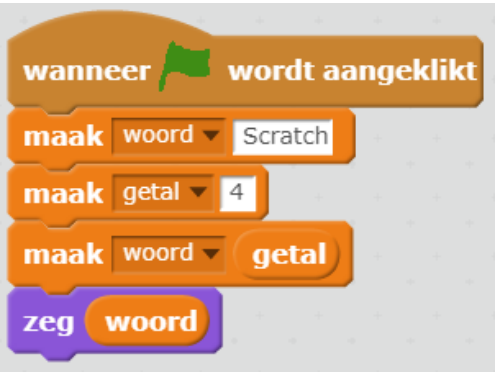
multiple
- 

optellen
- 

fixed

P7

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?



-

correct
-

logisch

- klopt
- Karel kiest zelf een woord aangepast

P8

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

- correct
- volgorde
- alles
- en dan dubbel

P9

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

-

correct 1
-

interpreteren 2
-

omzetten 3

P10

Kijk goed naar de blokken. Wat zal de leeuw zeggen als we op de groene vlag klikken?

-

correct
-

werkelijkheid
-

toegepast
-
overslaan

P11

Kijk goed naar de blokken. Wat zal Karel zeggen wanneer we op de groene vlag klikken?

-

en dan

correct
-

en dan

multiple
-

en dan

optellen
-

en dan

fixed

ZZZ

P12

Kik goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

-

correct
-

Logisch
-
klopt
- Karel kiest zelf een naam aangepast

P13

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

-

correct
-

volgorde
-

alles
-

en dan

dubbel

P14

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

-

correct
-

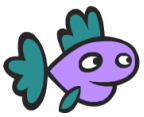
en dan

interpreteren
-

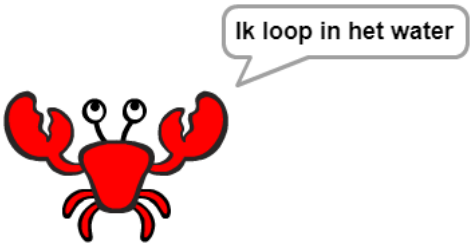
en dan

omzetten

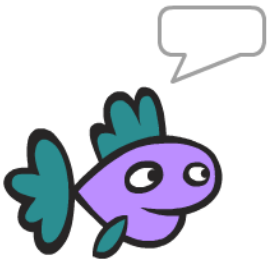
P15

Kijk goed naar de blokken. Wat zal de vis  zeggen als we op de groene vlag klikken?

-

correct
- 

werkelikheid
-

toegepast
- 
overslaan

P16

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?
Variabelen en waarden.

-

correct
-

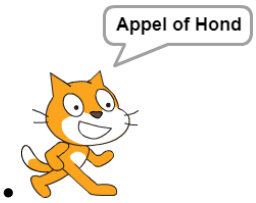
multiple
-

optellen
-

fixed

P17

Kijk goed naar de blokken. Wat zal Karel zeggen als we op de groene vlag klikken?

- 

en dan

correct
-

en dan

logisch
-
en dan
klopt
- Karel kiest een willekeurig soort fruit en dan een willekeurig dier aangepast

DDDD

P18

Kijk goed naar de blokken. Wat zal Karel zeggen en doen als we op de groene vlag klikken?

-

, neemt 10 stappen,

, neemt 10 stappen,

correct
-

,neemt 2 keer 10 stappen,

volgorde
-

, neemt 10 stappen,

,

, neemt 10 stappen,

alles
-

,neemt 2 keer 10 stappen,

dubbel

P19

EEEE

Kijk goed naar de blokken. Wat zal de donut zeggen als we op de groene vlag klikken?

-

correct
- Een donut kan helemaal niet praten of rekenen werkelijkheid
-

toegepast
-
overslaan

A1
Programmeren is leuk
•Niet waar/oneens •Gewoon/geen mening •Waar/eens

A2
Programmeren is voor jongens
•Niet waar/oneens •Gewoon/geen mening •Waar/eens

A3
De lessen waren leuk
•Niet waar/oneens •Gewoon/geen mening •Waar/eens

A4
De lessen waren moeilijk
•Niet waar/oneens •Gewoon/geen mening •Waar/eens

A5
Programmeren is voor meisjes
•Niet waar/oneens •Gewoon/geen mening •Waar/eens

A6
Mijn ouders/verzorgers vinden programmeren goed
•Niet waar/oneens •Gewoon/geen mening •Waar/eens

A7
Mijn vriendjes/vriendinnetjes vinden programmeren (denk ik) leuk

FFFF

•Niet waar/oneens •Gewoon/geen mening •Waar/eens

A8
Wat kun je met programmeren maken/doen? Noem minstens 1 ding.

Open

GGGG

Bijlage 14 Planning Lessen

Weeknr.		11	12	13	14	15	16	17	18	19	20	21	22
		2	1	2	1	2	1	2	X	X	1	2	1
Dinsdag		13-3	20-3	27-3	3-4	10-4	17-4	24-4	1-5	8-5	15-5	22-5	29-5
08.30-09.15		6C Luffher&Emily	5B Lufther	6C Luffher&Emily		6C Luffher&Emily	5B Emily	6C Luffher		vakantie	5B Luffher		5B Luffher
10.00-10.45		6A Luffher&Emily	5A Luffher	6A Luffher&Emily		6A Luffher&Emily	5A Emily	6A Luffher		vakantie	5A Luffher		5A Luffher
12.45-13.30		5D Luffher	5C Luffher	6B Luffher&Emily		6B Luffher	5C Emily	6B Luffher		vakantie	5D Luffher		5C Luffher
14.15-15.00		6B Emily					5D Emily			vakantie	5C Emily		5D Emily
Woensdag		14-3	21-3	9-4	4-4	11-4	18-4	25-4	2-5	9-5	16-5	23-5	30-5
08.30-09.30		8A Emily		11-12: 8A Emily		8A Emily		8A Emily		vakantie			
10.00-11.00		8C Emily		13-14: 8C Emily		8C Emily		8C Emily		vakantie			
11.30-12.30		8B Emily		14-15: 8B Emily		8B Emily		8B Emily		vakantie			
Vrijdag		16-3	23-3	30-3	6-4	13-4	20-4	27-4	4-5	11-5	18-5	25-5	1-6
08.30-09.30			7C Emily	vakantie	7C Emily		7C Emily			vakantie	7C Emily		
10.30-11.30			7A Emily	vakantie	7A Emily		7A Emily			vakantie	7A Emily		
14.00-15.00			7B Emily	vakantie	7B Emily		7B Emily			vakantie	7B Emily		

Groep 5 en 6: Locatie Korfmakersstraat 12
Groep 7 en 8: Locatie Schiedamseweg 227

Les 1
Les 2
Les 3
Les 4

Les 1 en 2 worden op locatie in het eigen klaslokaal gegeven.
Les 3 en 4 worden in het computerlokaal aan de schiedamseweg gegeven.

Experimentele groepen: onderstreepte groepen; extra uitleg en oefeningen (169 leerlingen)
Controle groepen: de overige groepen; geen extra uitleg en oefeningen (147 leerlingen)

Planning Lessen Het Blokhuus te Hoevelaken
Alle lessen worden gegeven door **Emily**
Groep 6/7 van 13.00-14.00, groep 6 van 14.00-15.00

Les 1	27-mrt
Les 2	3-apr
Les 3	10-apr
Les 4	24-apr

HHHH

Bijlage 15 Plan van Aanpak

Leerlijn Programmeren

In het programmeeronderwijs mist nog een leerlijn met opbouwende lessen en doorlopende leerdoelen. Naar aanleiding van dit onderzoek wordt een opzet voor een model gemaakt, gebaseerd op de klassieke theorie van Piaget en Neo-Piagetian theorieën. Implicaties voor lesmateriaal, leerdoelen en het aantal lessen worden nog achterwege gelaten.

Volgens de klassieke theorie van Piaget doorloopt ieder mens fasen in de cognitieve ontwikkeling, de doorstroming naar een fase gebeurd op basis van rijping van het brein. Leeftijd bepaalt dan dus de cognitieve fase. Volgens Neo-Piagetian theorieën doorloopt een mens inderdaad deze fasen, maar niet op basis van leeftijd. De fasen zouden per domein doorlopen worden. Wanneer iemand de laatste fase heeft afgerond, is hij/zij deskundig in het domein. Beide theorieën kennen (voor programmeren) waarheid. Ervaring hangt negatief samen met het aantal misconcepten. Er kan de aanname gemaakt worden dat kinderen met meer programmeerervaring in een verdere fase zitten (Neo-Piagetian). Anderszijds kan een kind van 7 niet deskundig worden in programmeren (Klassiek).
In het volgende model worden beide gecombineerd. Er worden levels opgesteld die enigszins samenhangen met de klassieke theorie. Per level varieert de mate waarin een fase voorkomt (Neo-Piagetian). Er is verscheidene literatuur te vinden die programmeertaken koppelt aan deze fasen. Deze moet dan toegespitst worden op de belevingswereld van de leerling.

Model

Level 1

Groepen 1 en 2

Klassieke theorie: pre-operationele fase tot 6 jaar.
Kinderen in deze groepen kunnen nog niet/nauwelijks lezen en zijn veelal gericht op ervarend leren (doen!). Mate waarin fasen naar voren moeten komen:

- Sensomotorische fase: 75%
- Pre-operationele fase: 25%

Level 2

Groepen 3 en 4

Kinderen in deze groepen kunnen beginnend lezen en schrijven en leren artificiële concepten2.

- *Sensomotorische fase:* 40%
- *Pre-operationele fase*: 40%
- *Concreet-operationele fase*: 15%
- *Formeel-operationele fase*: 5%

Level 3

Groepen 4 en 5

- *Sensomotirsche fase:* 10%
- *Pre-operationele fase:* 20%
- *Concreet-operationele fase:* 55%
- *Formeel-operationele fase:* 15%

Level 4

Groepen 7 en 8

Klassieke theorie: formeel operationele fase vanaf 11 jaar

- *Sensomotorische fase:* 10%
- *Pre-operationele fase:* 10%
- *Concreet-operationele fase:* 30%
- *Formeel-operationele fase*: 50%

Programmeertaken

Er moet per fase beschreven worden welke programmeertaken uitgevoerd kunnen worden. De taken moeten gekoppeld worden aan leeftijd en zo bepalen in welke mate (%) de fase in een level voorkomt. De capaciteiten van kinderen uit groepen 1 en 2 zijn nog heel basaal, waardoor af te vragen is in hoeverre het relevant is om taken uit de laatste twee fasen naar de belevingswereld te vertalen.
In de andere levels komen wel alle fasen voor. De taken behorend bij de fase worden dus ieder level vertaald naar de belevingswereld. Hierdoor wordt waarschijnlijk in ieder level aspecten van de taak overgeslagen. Door de fase in ieder level terug te laten komen, kunnen er verschillende aspecten behandeld worden en wordt de fase geheel doorlopen.
In iedere fase komen ook hogere fasen voor, om het kind te prikkelen en snelle leerders gemotiveerd te houden.

Lesmateriaal

Het uiteindelijke doel zou zijn het aanbieden van een ruime hoeveelheid lesmateriaal, waarin de docent ook keuzes kan maken. Om het programmeeronderwijs gevarieerd te houden, moet didactiek gevarieerd blijven. Denk aan plugged en un-plugged materiaal, klassikaal en individueel programmeren en concepten binnen en buiten computer context plaatsen.

Een belangrijk aspect voor de sensomotorische fase: leren hoe de computer werkt, voor de basiskennis.
Voor level 1 kan nog leuk en logisch Hello Ruby materiaal gebruikt worden.