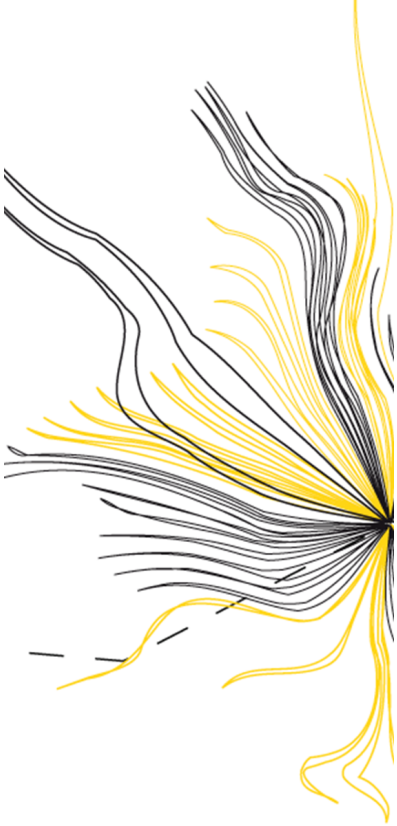


UNIVERSITY OF TWENTE.

Faculty of Behavioural, Management & Social Sciences

Vernieuwing en Plezier in Programmeeronderwijs in de Bovenbouw

Ekambaranathan, A & van der Zaag, H

Onderzoek van Onderwijs Verslag
Januari 2018

Begeleiders:
dr. L.E.I. Breymann
dr. J.T. van der Veen

Universiteit Twente
ELAN, vakgroep Docentontwikkeling
Ravelijn
P.O. Box 217
7500 AE Enschede
Nederland

Contents

1 Samenvatting

Programmeren is een essentieel onderdeel van informaticaonderwijs. Echter, veel lesmateriaal op dit gebied is verouderd en past niet bij het huidige tijdsbeeld. We hebben daarom een nieuwe programmeermodule ontworpen waarin gebruik wordt gemaakt van de taal *Processing*. Deze taal maakt het eenvoudig om animaties te programmeren en hiermee spelen we in op de belevingswereld van de leerlingen. Leerlingen krijgen visueel feedback op wat ze hebben geprogrammeerd en kunnen animaties makkelijk aanpassen door parameters in de code te wijzigen. De lesmodule is vervolgens geïmplementeerd in de klas en achteraf geëvalueerd. Het ontwerp blijkt effectief en toegankelijk voor de leerlingen en ze hebben de module over het algemeen plezier ervaren. De module kan verbeterd worden door in de HAVO-variant meer aandacht te besteden aan de maatschappelijke relevantie van programmeren en door in de VWO-variant de opdrachten uitdagender te maken.

2 Inleiding

Het informaticaonderwijs in Nederland begint langzaam populairder te worden en programmeren is hier een belangrijk onderdeel van. Programmeren wordt vaak gezien als een lastig onderwerp dat weggelegd is voor technisch ingestelde mensen. We hebben onderzoek gedaan naar het ontwerp en de implementatie van een lesmodule waarin programmeren op een plezierige en interactieve wijze wordt geïntroduceerd aan de leerlingen.

2.1 Aanleiding

Gedurende onze lerarenopleiding (informatica) is er veel discussie geweest over programmeren en hoe dit het beste te leren is. Ook over de keuze in programmeertaal is veel gesproken. Wat wij, Anirudh Ekambaranathan en Heleen van der Zaag opviel, is dat veel beschikbaar lesmateriaal al sterk verouderd is. Dit was ook te merken op onze stageschool, Lyceum de Grundel in Hengelo. De school werkt met de methode Enigma Online. De module programmeren hierin is echter achterhaald en spreekt niet tot de verbeelding van de leerlingen. In de module programmeren van Enigma Online is gekozen voor de programmeertaal Java met een focus op de AWT en Swing Library voor grafisch programmeren.

Echter, in de praktijk, voor grafisch programmeren, zal hier zelden gebruik van gemaakt worden. In de industrie wordt voor grafisch ontwerpen al snel gebruik gemaakt van talen als OpenGL en DirectX, of software als Unity3D en AutoCAD. Deze grafische programmeertalen vergen echter specifieke technische kennis, bijv. matrix algebra, en zijn geen goede opties als een introducerende taal tot programmeren.

Daarentegen zijn er veel makkelijker aanleerbare talen die voortborduren op traditionele talen (C++ en Java) waarmee leerlingen de basisprincipes van programmeren kunnen leren. Voorbeelden van zulke talen zijn *Processing* en *NodeBox*. Met deze talen krijgen leerlingen direct visueel feedback, dit helpt de leerlingen om duidelijk het effect van wat ze programmeren te zien. Hierdoor kunnen ze makkelijker de technische concepten leren terwijl ze de programmeertaal gebruiken om hun creativiteit te uiten.

We hebben op onze stageschool de ruimte gekregen om de programmeerlessen zelf in te vullen. We hebben gekozen om een lesmodule te ontwerpen om de leerlingen de basisconcepten van programmeren bij te brengen. Hiervoor maken we gebruik van een programmeertaal die de leerlingen

direct visueel feedback geeft. Wij hopen hiermee programmeren te combineren met creativiteit waarbij het hopelijk voor de leerlingen leuker en makkelijker word. Hierdoor hopen wij dat de leerlingen gemotiveerd raken en blijven om te leren programmeren.

2.2 'Uitvoering'

Gedurende de eerste weken van onze stage hebben wij onze module voorbereid, waarbij Anirudh Ekambaranathan zich inzette voor de HAVO-klas en Heleen van der Zaag voor de VWO klas. De module is vanaf november 2016 uitgevoerd over een periode van acht weken. Helaas was er ook lesuitval (buiten onze schuld om) waardoor er geschoven moest worden in het programma. Het effect hiervan was dat wij minder lessen hadden dan wij van te voren hadden gepland. Hier hebben wij de module op aangepast. De module is afgesloten met een project van twee weken, waarna de leerlingen een evaluerende enquête hebben ingevuld.

2.2.1 Wie heeft wat gedaan

Tijdens onze stage was, zoals hierboven genoemd, Anirudh verantwoordelijk voor de HAVO-klas, en Heleen voor de VWO-klas. Dit gold ook voor al het materiaal dat voor de verschillende klassen ontwikkeld moest worden. Gezamenlijk werd het ontwerp en de richting bepaald, en apart werd dit verder uitgewerkt voor beide klassen. Hierdoor kon er gericht rekening gehouden worden met het niveauverschil.

In dit verslag is Anirudh hoofdverantwoordelijke geweest voor de hoofdstukken Literatuur, Lesontwerp, Discussie en Conclusie. Heleen is dit geweest voor de hoofdstukken Inleiding, Evaluatiemethode en Resultaten. Het is wel zo dat wanneer er informatie specifiek voor een klas geschreven moest worden, dit gedaan is door degene die voor deze klas verantwoordelijk was.

2.3 Organisatie

Om de module te ontwerpen (hoofdstuk 3) hebben wij gebruik gemaakt van literatuur (paragraaf 3.2), waaruit een aantal ontwerpeisen zijn afgeleid (paragraaf 3.4). In hoofdstuk 4 wordt uitgelegd hoe deze eisen gebruikt zijn om tot een lesontwerp te komen. Hierna wordt in hoofdstuk 5 op basis van de ontwerpeisen de evaluatie eisen en -methoden gedefinieerd. In hoofdstuk 6 worden de resultaten gepresenteerd en daarna in hoofdstuk 7 bediscussieerd. In hoofdstuk 8 worden de conclusies van het onderzoek gepresenteerd.

3 Ontwerpopdracht

In dit onderzoek proberen we de volgende onderzoeksvraag te beantwoorden:

- Hoe kan programmeren als een lesmethode worden aangeboden, zodat het toegankelijk, plezierig en maatschappelijk relevant is voor vierdejaars leerlingen op de middelbare school?

3.1 Deelvragen

Hiervoor definiëren we de volgende deelvragen:

- Hoe kan programmeren als een lesmethode worden aangeboden aan vierdejaars informaticaleerlingen zodat de moeilijkheidsgraad, de laagdrempeligheid en technische methode van de lesmethode juist is voor technisch ingestelde en minder technisch ingestelde leerlingen?
- Hoe kan programmeren als een lesmethode worden aangeboden aan vierdejaars informaticaleerlingen zodat de maatschappelijke relevantie van programmeren duidelijk wordt waardoor zij inzien hoe programmeren bruikbaar en nuttig is?
- Hoe kan programmeren als een lesmethode op een manier worden aangeboden aan vierdejaars informaticaleerlingen zodat ze het als plezierig ervaren?

3.2 Literatuur

3.2.1 Huidig informaticaonderwijs.

In het Nederlands onderwijsstelsel is informatica een keuzevak in de bovenbouw voor de richtingen HAVO en VWO. Het curriculum was geïnspireerd door het UNESCO/IFIP curriculum van 1994 (Van Weert, 1998) en opgezet in 1998 met het idee dat het toegankelijk moest zijn voor leerlingen met verschillende achtergronden. Bovendien was informatica geen eis tot toelating bij vervolgopleidingen, wat er toe leidde dat er geen centraal examen aan gekoppeld was (Barendsen, Grgurina, & Tolboom, 2016).

Doordat informatica een snel ontwikkelende discipline is, verouderde het curriculum van 1998 binnen enkele jaren. In 2007 heeft er een re-evaluatie plaatsgevonden (Schmidt, 2008), maar er waren niet veel veranderingen doorgevoerd. Een sterk curriculum is belangrijk omdat informatica een grote rol zal spelen in de verschuiving van werkgelegenheid. Digitalisering en automatisering zal in de komende 30 jaar 47% van de banen in de Verenigde Staten bedreigen (Frey & Osborne, 2017). Dit betekent dat informatici nodig zijn en zo heeft de Association for Computing Machinery (ACM) de voorspelling gedaan dat in de toekomst 50% van de banen op technisch gebied in de ICT zal zijn (Kaczmarczyk & Dopplick, 2014).

Er is daarom veel zorg geuit door informaticadocenten in Europa en Amerika (Gander et al., 2013; Kaczmarczyk & Dopplick, 2014) en binnen Nederland door de Koninklijke Nederlandse Akademie van Wetenschappen 2013 en door SLO Schmidt (2008). Deze zorgen gingen vooral over "de vergrijzing van de docentenpopulatie in combinatie met de (nog) geringe deelname aan de eerstegraads lerarenopleidingen" Schmidt (2008). Als gevolg hiervan heeft het Ministerie van Onderwijs een onderzoek laten uitvoeren (Tolboom, Krüger, & Grgurina, 2014) waaruit naar voren is gekomen dat een advies voor een nieuw examenprogramma is gewenst.

In 2016 is een nieuw examenprogramma uitgebracht (Barendsen et al., 2016) waarbij de nadruk is gelegd op de duurzaamheid van het programma. In het concept-contextparadigma (Bruning & Michels, 2013) wordt sterk gewerkt vanuit een conceptbenadering. Het idee hierachter is dat concepten minder snel veranderen, in tegenstelling tot contexten, die zich erg snel kunnen ontwikkelen. In het nieuwe programma zijn concepten dus concreet uitgewerkt, terwijl contexten generiek zijn beschreven.

Het kernprogramma bevat een domein waarin de algemene vaardigheden zijn opgenomen, met daarnaast 5 kennisdomeinen. Naast het kernprogramma zijn er keuzethema's bestaande uit 12 domeinen. HAVO-leerlingen kiezen twee keuzethema's en VWO leerlingen kiezen er 4. Tabel 1 laat een schematisch overzicht zien van deze domeinen.

Kernprogramma	Keuzethema's
Domein A Vaardigheden	Domein G Keuzethema Algoritmiek, berekenbaarheid en logica
Domein B Grondslagen	Domein H Keuzethema Databases
Domein C Informatie	Domein I Keuzethema Cognitive computing
Domein D Programmeren	Domein J Keuzethema Programmeerparadigma's
Domein E Architectuur	Domein K Keuzethema Computerarchitectuur
Domein F Interactie	Domein L Keuzethema Netwerken
	Domein M Keuzethema Physical computing
	Domein N Keuzethema Security
	Domein O Keuzethema Usability
	Domein P Keuzethema User experience
	Domein Q Keuzethema Maatschappelijke en individuele invloed van informatica
	Domein R Keuzethema Computational science

Table 1: Domeinen van het informaticacurriculum

3.2.2 Programmeren in informaticaonderwijs.

Programmeren is een belangrijk onderdeel van informatica in het kernprogramma. Dit betekent dat elke informaticaleerling tijdens het bovenbouwtraject in aanraking zal komen met programmeren. Naast het feit dat het de leerlingen een platform geeft om problemen mee op te lossen, is begrip van programmeerconcepten vaak het belangrijkste onderdeel in assessment van de leerlingen in verschillende onderwerpen binnen informatica (Decker, 2007; Mead et al., 2006; Tew & Guzdial, 2010; Whalley & Lister, 2009). Kennis van programmeren kan dus verreist worden in onderwerpen als 'hardware' (bijvoorbeeld programmeren met Arduino's) of 'web programmeren'.

Volgens onderzoek van het SLO blijkt dat docenten programmeren ook een belangrijk onderdeel vinden van het curriculum (Tolboom et al., 2014). Er is echter geen advies uitgebracht over welke programmeertaal gehanteerd moet worden. Docenten zijn dus vrij om te kiezen welke programmeertaal ze willen gebruiken in de lessen. Dit kan een belangrijke keuze zijn omdat sommige talen het begrip van programmeerconcepten beter faciliteren dan andere talen (Good, Brna, & Cox, 1997; Wiedenbeck & Ramalingam, 1999; Wiedenbeck, Ramalingam, Sarasamma, & Corritore, 1999). Zo hebben leerlingen bijvoorbeeld meer moeite met onderwerpen als 'pointers' en 'geheugen allocatie' (Milne & Rowe, 2002), welk terugkomen in talen als C en C++. Aan de andere kant hebben leerlingen ook moeite met Object Geörienteerde (OO) principes in talen zoals Java. In de wetenschap is er echter geen overeenkomst over welke taal een leerling als eerste zou moeten leren (Culwin, 1997). Docenten geven echter wel aan dat leerlingen die eerst een OO taal leren, minder moeite hebben met lastigere principes uit procedurele talen (Milne & Rowe, 2002).

Omdat er geen centraal examen is en docenten vrij zijn om een een programmeertaal te kiezen, lopen de talen die leerlingen leren uiteen. Zo is het mogelijk dat op sommige scholen een webprogrammeertaal wordt gekozen, bijv. JavaScript, en op een andere school een OO taal als Python of Java.

3.2.3 Onderwijsgerelateerde uitdagingen

Naast verschillen in programmeertalen en scholen, moet de lesmodule rekening houden met verschillen tussen leerlingen, onder andere veroorzaakt door de structuur van het onderwijsstelsel. Zo zijn er bijvoorbeeld verschillen tussen HAVO- en VWO-leerlingen, maar ook tussen de leerlingen binnen elke klas. Deze verschillen worden hieronder kort aangestipt.

Differentiatie HAVO/VWO SLO heeft uitgebreid beschreven wat de verschillen tussen HAVO- en VWO-leerlingen (moeten) zijn (Michels, 2006). Op het gebied van informatica beschrijft SLO HAVO-leerlingen als *denkende doeners* en VWO-leerlingen als *doende denkers* (Barendsen & Tolboom, 2016). Ruwweg betekent dit dat HAVO-leerlingen meer praktisch ingesteld zijn, terwijl VWO-leerlingen dieper ingaan op de theorie.

Met dit verschil moet rekening gehouden worden tijdens het ontwerp van de lesmodule. We kunnen niet dezelfde module presenteren aan zowel HAVO als VWO. Dit leidt tot een eerste ontwerpeis:

> *Ontwerpeis 1:* De lesmodule moet rekening houden met het verschil tussen HAVO en VWO.

Differentiatie profielen SLO geeft ook aan dat het verschil tussen de leerlingen met verschillende profielen een grotere impact kan hebben dan het verschil tussen de leerroutes.

In de bovenbouw moeten leerlingen een profielkeuze maken. Er zijn in totaal vier profielen waar de leerlingen uit kunnen kiezen en bij sommige profielen ligt de focus op technische vakken terwijl bij andere profielen de focus ligt op sociale vakken.

Zo hebben leerlingen met een C&M profiel een heel andere technische achtergrond dan leerlingen met een N&T profiel. Zoals later uitgelegd zal worden, kan dit een probleem vormen omdat sommige leerlingen geen sterke wiskundige achtergrond zullen hebben.

3.2.4 Moeilijkheden in het leren programmeren.

Programmeren is moeilijk om te leren en het wordt gezegd dat het 10 jaar kan duren voordat een beginner een expert wordt (Soloway, 2013). Veel onderzoek is gedaan naar de karakteristieken van beginnende programmeurs om zo lesmethodes te verbeteren. Omdat talen als C++ en Java op universiteiten vaak als eerste worden aangeleerd, is recent ook gekeken naar de verschillen in moeilijkheid tussen procedurele en object geörienteerde talen.

Beginnende programmeurs. Beginnende programmeurs zijn programmeurs die niet de vaardigheden en kennis hebben van experts. In deze paragraaf wordt ingegaan op karakteristieke eigenschappen die beginnende programmeurs onderscheidt van experts.

Kenmerkend voor beginnende programmeurs is dat ze maar oppervlakkige kennis hebben van programmastructuren en als gevolg hiervan benaderen ze programmeren "regel per regel", terwijl ervaren programmeurs het van een macroperspectief benaderen (Winslow, 1996). Experts hebben efficiënt georganiseerde kennisschema's die ze toepassen op functionele karakteristieken van een algoritmisch probleem in plaats van op details als bijvoorbeeld syntax of taal (Guindon, 1990). Beginners spenderen weinig tijd aan het ontwerp en het testen van de code, en corrigeren hun programma's door kleine veranderingen aan te brengen in plaats van hun ontwerp te re-evalueren (Kölling & Rosenberg, 1996).

In een survey-onderzoek van Milne en Rowe 2002 werden leerlingen en docenten gevraagd een enquête in te vullen na het volgen van een C++ cursus. Hierin gaven leerlingen aan dat ze minder moeite hadden dan dat hun docenten aangaven. Leerlingen hadden de indruk dat ze het materiaal begrepen hadden, maar de docenten konden de tekortkomingen van de programma's van de leerlingen zien. Leerlingen hebben dus moeite om hun eigen tekortkomingen te zien.

Verder spelen de vaardigheden en de mindset van de leerlingen ook een rol in het leren van programmeren (Gomes & Mendes, 2007). Logische en wiskundige vaardigheden helpen leerlingen bij het programmeren (Gomes & Mendes, 2007). Leerlingen hebben moeite om tekstuele problemen te vertalen naar een wiskundig model of een andere abstracte entiteit (Gomes, Carmo, Bigotte, & Mendes, 2006). Een relatie is aangetoond tussen wiskundige vaardigheden en het niveau van programmeren (Byrne & Lyons, 2001). Leerlingen die dus geen sterke wiskunde of technische achtergrond hebben, wat mogelijk is, zullen meer moeite hebben met het leren programmeren. Omdat informatica toegankelijk is voor alle leerlingen, is het mogelijk dat leerlingen met een C&M profiel (en dus wiskunde C) het vak volgen. Dit vormt een reëel probleem in het huidig informaticaonderwijs.

> *Ontwerpeis 2: De lesmodule moet rekening houden met verschillen in wiskundige achtergronden van de leerlingen en moet toegankelijk zijn voor leerlingen die geen sterke wiskundeachtergrond hebben.*

Gomes and Mendes (2007) geven aan dat leerlingen ook niet precies weten hoe een probleem opgelost moet worden. Ze kunnen moeite hebben om (i) het probleem te begrijpen, (ii) kennis te relateren aan het probleem door de juiste analogieën toe te passen, (iii) te reflecteren over het probleem en de oplossing (iv) en/of snel opgeven omdat ze het antwoord niet snel vinden. Ze geven aan dat dynamische concepten (bijv. hoe het geheugen verandert) vaak statisch wordt overgebracht (bijv. op het bord in plaats van met een animatie). De focus van een beginnende programmeercursus moet liggen op de programmeervaardigheden van de leerlingen en niet op de syntactische details.

> *Ontwerpeis 3:* In de lesmodule moet de nadruk liggen op het toepassen van programmeerconcepten en niet op de syntactische details van de programmeertaal.

Motivatie speelt ook een rol bij aspecten van informatica. Doordat een programmeur een publiek imago heeft van een 'nerd' en programmeren wordt opgevat als een moeilijke vaardigheid, missen sommige leerlingen de intrinsieke motivatie om het te leren (Jenkins, 2002). Zonder intrinsieke motivatie is het erg lastig om succesvol te zijn (Ng & Bereiter, 1991). Door een vertekend publiek imago hebben leerlingen een slecht beeld van de maatschappelijke bijdrage van het vak programmeren. In een vernieuwde lesmodule hebben we de verantwoordelijkheid om een beeld te schetsen van het vak in een bredere context.

> *Ontwerpeis 4:* De lesmodule laat zien wat de maatschappelijke relevantie is van programmeren.

Bij het leren van programmeren kunnen twee soorten leerlingen worden onderscheiden. Perkins, Hancock, Hobbs, Martin, and Simmons (1986) maken gebruik van de termen 'movers' en 'stoppers'. 'Stoppers' geven het in een moeilijke situatie snel op en geloven niet dat ze het probleem kunnen oplossen. 'Movers' blijven in moeilijke situaties nieuwe dingen proberen door bijvoorbeeld de code aan te passen en geven het dus niet zo makkelijk op. Het verschil tussen effectieve en ineffectieve

beginners, gegeven dat hun kennis uniform is verdeeld, zijn dus bestaande attitudes en strategieën (Robins, Rountree, & Rountree, 2003).

Hieraan voegen Gomes and Mendes (2007) toe dat ook lesmethodes en leerstrategieën een rol spelen. Hierbij geven ze als problemen dat docenten niet altijd gepersonaliseerd les kunnen geven en verschillende leerstijlen niet ondersteunen, wat resulteert in leerlingen die niet de juiste leerstrategieën gebruiken. Zoals Jenkins (2002) al aangeeft, het is de verantwoordelijkheid van de docenten om er voor te zorgen dat leerlingen de juiste leerstrategieën gebruiken. Met andere woorden, de lessen moeten gepersonaliseerd worden voor de leerlingen.

Ontwerpeis 5: De module moet gepersonaliseerd leren faciliteren.

Programmeeraspecten. Leren programmeren betekent niet alleen dat je de syntax van een taal begrijpt. Het brengt een aantal verschillende aspecten samen; zo moet een leerling bijvoorbeeld ook leren hoe een programma ontworpen wordt en hoe je fouten het beste kunt analyseren.

Veel voorkomende misconcepties en moeilijkheden voor beginnende programmeurs is beschreven in (Pane & Myers, 1996; Soloway, 2013). Leerlingen hebben bijvoorbeeld meer moeite met het declareren van een variabele dan het aanpassen van een variabele na declaratie. Fouten in “loops” (lussen) en ”if-statements” (condities) zijn veel voorkomend. Leerlingen hebben ook moeite met notaties die erg veel op elkaar lijken, zoals het verschil tussen een tekst variabele (x = ”20”) en een decimale variabele (x = 20) en ook met constructies waarbij kennis nodig is van de werking van het geheugen (Milne & Rowe, 2002), zoals het gebruik van pointers.

Echter, Robins et al. (2003) geven aan dat leerlingen de meeste moeite hebben met het ontwerpen van het programma. Een leerling kan de *programmeerkennis* hebben, maar niet weten welke *strategie* gebruikt moet worden (Davies, 1993). Het onderscheid is belangrijk, want leerlingen begrijpen vaak wel de syntax en de semantische betekenis van individuele regels code, maar kunnen dit niet combineren tot een geheel programma (Winslow, 1996). Ook al weten ze hoe ze een probleem op papier moeten oplossen, vinden ze het lastig om dit te vertalen naar een computerprogramma.

Ontwerpeis 6: De module moet leerlingen begeleiden bij het ontwerpen van hun code en programma’s.

Object Geörienteerd of Procedureel? Docenten hebben de keuze om een programmeertaal te kiezen voor hun leerlingen. Er was lang gedacht dat object geörienteerde talen op een natuurlijke en intuïtieve wijze problemen van de echte wereld beschrijven, echter studies bevestigen dit niet altijd (Robins et al., 2003). Hoewel de klassen van object geörienteerde taal leerlingen helpen om echte entiteiten te modelleren (Sajaniemi, 2002), maakt de onderliggende complexe controlestroom (control flow) het moeilijk voor beginners een mentale representatie te vormen van het programma (Ala-Mutka, 2004). Veel docenten in de studie van Milne and Rowe (2002) geven aan dat object geörienteerd programmeren pas gebruikt moet worden nadat leerlingen een beter begrip hebben van procedureel programmeren.

Op middelbare scholen is er vaak ook niet genoeg tijd om een hele object geörienteerde taal te behandelen in de lessen. Daarom wordt vaak gekozen voor een webprogrammeertaal, PHP of JavaScript, waar het object geörienteerde deel wordt weggelaten. Immers, een taal als Java wordt in het hoger onderwijs vaak aangeboden als een vak dat een kwartiel of een semester lang is.

Door het object geörienteerde deel weg te laten kan de leerling zich richten op het toepassen en creëren van programma’s. Leerlingen kunnen op deze manier kennis maken en ervaring op doen

met programmeren. Lastigere OO principes kunnen eventueel uitgelegd worden aan leerlingen die meer uitgedaagd willen worden.

Visualiseertechnieken binnen programmeren. Technieken waar programmastructuren worden gevisualiseerd zijn vaak onderzocht en worden ook toegepast in het onderwijs. Veel onderzoek heeft betrekking op visualisatie van algoritmes, zoals recursie (Du Boulay, 1986; Soloway, 2013), en de interne werking van het geheugen (Rowe & Thorburn, 2000). Leerlingen kunnen zien hoe het geheugen verandert terwijl het programma wordt uitgevoerd.

Elenbogen, Maxim, and McDonald (2000) zijn een stapje verder gegaan en hebben een interactieve webapplicatie ontwikkeld voor de taal C++. Hoewel het niet erg grafisch is, hebben de leerlingen de mogelijkheid om (bestaande) code en templates aan te passen om zo kijken hoe het gedrag van het programma verandert. Dit laatste is wel een belangrijk element. Leerlingen die interactie kunnen hebben met een visualisatietechniek scoren beter dan leerlingen die het passief observeren (Hundhausen, Douglas, & Stasko, 2002).

Het idee is daarom om een programmeertaal te kiezen die interactie met visualisaties faciliteert. Hierbij is het resultaat van het programma een visualisatie en leerlingen kunnen deze visualisaties aanpassen door parameters in de code te wijzigen. Naps et al. (2002) definiëren zes verschillende soorten van gebruikersinteractie als het aankomt op visualisatie. Hierbij maken wij gebruik van het soort "aanpassen". De 6 soorten staan hieronder.

- **Niet kijken.** Dit gebruikt geen visualisatie.
- **Kijken.** Leerlingen zien een visualisatie van bijvoorbeeld een programma, maar kunnen hier geen interactie mee hebben. Dit is een kernmethode van visualisatie omdat alle andere vormen hiervan afhankelijk zijn.
- **Responsief.** Leerlingen zien een visualisatie worden vragen gesteld over wat er gaat gebeuren (voorspellen) of welke code erachter zit (programmeren).
- **Aanpassen.** Leerlingen kunnen de visualisatie veranderen door, bijvoorbeeld, de invoer van het algoritme te wijzigen.
- **Presenteren.** Leerlingen krijgen een visualisatie gepresenteerd, door de docent, en vervolgens is er ruimte voor discussie of vragen.

Ontwerpeis 7: De lesmodule maakt gebruik van een programmeertaal die interactie met visualisaties faciliteert.

Met deze ontwerpeis hebben we het niet over visuele talen zoals Blockly (*Blockly*, n.d.) of Scratch (*Scratch*, n.d.), waarin programma's gemaakt kunnen worden door bijvoorbeeld blokjes te verschuiven met de muis. In ons geval wordt een taal bedoeld waarin visualisaties geprogrammeerd kunnen worden. Leerlingen krijgen dus direct het resultaat te zien en kunnen deze visualisatie makkelijk aanpassen door de code te wijzigen.

3.3 Ervaring van de leerlingen

Tot zo ver zijn de eisen geformuleerd gebaseerd op de inhoud van het lesmateriaal. Hieronder wordt kort gekeken naar twee psychologische elementen die ook een rol spelen bij het implementeren bij een lesmodule: plezier en tevredenheid.

3.3.1 Plezier

Het belangrijk dat leerlingen de module positief en plezierig ervaren. Dit klinkt erg logisch en vanzelfsprekend, maar wordt ook ondersteund door voorgaand onderzoek. Al in de jaren vijftig beargumenteerden wetenschappers dat plezier leidt tot effectievere leren (Huizinga & Hull, 1949). Recenter heeft Robinson and Kakela (2006) aangetoond dat plezier en vertrouwen in het klaslokaal betrokkenheid, leren, en begrip van de leerlingen bevordert.

Plezier heeft ook fysieke en neurologische gevolgen. Plezier en positieve emoties maken dopamine vrij in de hersenen en dit in gevolg leidt weer tot verhoogde niveaus van creativiteit en motivatie (Willis, 2006).

Over het algemeen gaat men ervan uit dat leerlingen beter leren wanneer ze meer plezier hebben in het leerproces (Tews, Jackson, Ramsay, & Michel, 2015).

Ontwerpeis 8: Leerlingen moeten de lesmodule plezierig ervaren.

3.3.2 Tevredenheid

Tevredenheid van de leerlingen is altijd belangrijk geweest voor onderwijsinstellingen (Orpen, 1990), omdat het een indicatie geeft over de succes en effectiviteit van de instelling. Door de tevredenheid van de studenten te meten is het mogelijk om de kwaliteit van een module te verbeteren (Wiers-Jenssen, Stensaker, & Grogaard, 2002). Het helpt om de kwaliteit van de module te meten en om om het lesmateriaal van de docenten te verbeteren (Kulik, 2001). Dit leidt tot onze laatste ontwerpeis.

Ontwerpeis 9: Leerlingen moeten tevreden zijn over de lesmodule.

3.4 Eisen aan het ontwerp

Gebaseerd op de literatuuranalyse in de vorige sectie hebben we een programma van eisen kunnen opstellen waaraan de lesmodule moet voldoen. Deze eisen staan hieronder kort samengevat.

1. Differentiatie HAVO/VWO. De lesmodule moet rekening houden met het verschil tussen HAVO en VWO.

2. Toegankelijkheid. De lesmodule moet rekening houden met verschillen in wiskundige achtergronden van de leerlingen en moet toegankelijk zijn voor leerlingen die geen sterke wiskundeachtergrond hebben.

3. Nadruk ligt op toepassing. In de lesmodule moet de nadruk liggen op het toepassen van programmeerconcepten en niet op de syntactische details van de programmeertaal.

4. Maatschappelijke relevantie. De lesmodule laat zien wat de maatschappelijke relevantie is van programmeren.

5. Gepersonaliseerd leren. De module moet gepersonaliseerd leren faciliteren.

6. Begeleiding bij het plannen en ontwerpen. De module moet leerlingen begeleiden bij het ontwerpen van hun code en programma's.

7. Visualisatie. De lesmodule maakt gebruik van een programmeertaal die interactie met visualisaties faciliteert.

8. Plezier. Leerlingen moeten de lesmodule plezierig ervaren.

9. Tevredenheid. Leerlingen moeten tevreden zijn over de lesmodule.

4 Lesontwerp

Gebaseerd op de eisen die hierboven staan beschreven is een lesontwerp samengesteld. De uiteindelijke lesmodule staat in bijlage 1. Hieronder staan de ontwerpkeuzes toegelicht.

4.1 Programmeertaal

Er is gekozen voor een programmeertaal waarbij leerlingen direct visueel feedback krijgen en waar leerlingen interactie kunnen hebben met de visualisaties. Door code aan te passen inspecteren leerlingen hoe het gedrag van het programma verandert. Daarnaast wilden we in principe kiezen voor een procedurele taal, maar we wilden ook ruimte bieden voor talentontwikkeling en verkenning van object geörienteerde principes.

Om deze redenen is gekozen voor de programmeertaal *Processing*, welk wordt gebruikt om visualisaties te creëren. Dit is een taal ontwikkeld voor leerlingen als eerste programmeertaal en is geïnspireerd door talen als *BASIC* en *Logo*.

Het voordeel hiervan is dat leerlingen direct visualisaties kunnen programmeren en kunnen experimenteren met de code om het gedrag van programma's te analyseren.

4.1.1 Visualisatie

Processing is een programmeertaal waar leerlingen direct beginnen met het creëren van animaties. Zo kunnen ze figuren en vormen met enkele regels code op het scherm laten verschijnen. Door parameters aan te passen zien ze bijvoorbeeld hoe een rechthoek groter wordt en hoe de kleur verandert. Zoals Hundhausen et al. (2002) aangeven, leren leerlingen meer doordat ze de visualisaties kunnen aanpassen.

4.1.2 Nadruk op toepassing

In de lessen is het de bedoeling dat leerlingen leren programmeren door actief bezig te zijn. De lessen zijn daarom zo ingericht dat ongeveer 10 minuten wordt besteed aan de uitleg van de theorie, waarna leerlingen zelfstandig of in tweetallen de theorie kunnen toepassen.

Ze krijgen de kans om de theorie toe te passen tijdens een praktische opdracht waarin ze bijvoorbeeld een simpele animatie moeten programmeren. Van leerlingen wordt dus niet verwacht dat ze de details van de programmeertaal uit hun hoofd leren, maar ze leren de taal gaandeweg door ze toe te passen.

4.1.3 Toegankelijkheid

Zoals aangegeven is het belangrijk dat de module toegankelijk is voor leerlingen met verschillende profielen en verschillende technische achtergronden. Voor deze module is daarom geen voorkennis

vereist. Er wordt ook niet ingegaan op complexe algoritmes of concepten als recursie of optimalisatie.

Verder is technische kennis niet vereist om een eerste programma te creëren. Moeilijkere concepten worden gaandeweg geïntroduceerd en bouwen voort op voorgaande lessen. Kennis van concepten uit de wiskunde, zoals statistiek of combinatoriek is niet vereist. Daarentegen is het wel zo dat leerlingen die al eerder geprogrammeerd hebben het minder moeilijk zullen hebben omdat ze al bekend zullen zijn met programmeerconcepten.

4.2 Gepersonaliseerd leren

Zoals Gomes and Mendes (2007) aangeven, is het belangrijk dat lessen worden gepersonaliseerd voor de leerlingen. In de lessenseries wordt dit gedaan door de theorie op verschillende manieren te presenteren aan de leerlingen. Zo wordt er rekening gehouden met de verschillen in leerstijlen.

- **Klassikaal.** Alle theorie wordt klassikaal uitgelegd aan de leerlingen. Dit zorgt ervoor dat er duidelijkheid is over wat de leerlingen minimaal moeten leren. Verder worden ze niet meteen in het diepe gegooid wanneer complexere concepten geïntroduceerd worden.
- **Individueel.**
  - *Theorie.* Voor de leerlingen die liever zelfstandig leren is de theorie ook beschikbaar. Ze kunnen gebruik maken van het boek over *Processing* (Shiffman, 2009), waarvan de docent een aantal exemplaren kan meenemen. Ze kunnen ook gebruik maken van de website, de slides van de docenten, of van de theorie die de docent online eventueel beschikbaar stelt. Leerlingen kunnen zelfstandig en in hun eigen tempo de stof doornemen.
  - *Zelfstandig.* Als laatste zijn er ook vaak leerlingen die bijvoorbeeld al eerder hebben geprogrammeerd en met behulp van de API een behoorlijk ver komen. Voor die leerlingen wordt er doorverwezen naar de website waar de API beschikbaar is. Ze kunnen dan experimenteren met de nieuwe mogelijkheden van *Processing*.

4.2.1 Differentiatie HAVO/VWO

In de module wordt onderscheid gemaakt tussen HAVO en VWO. Zoals eerder aangeven, worden VWO'ers beschouwd als *doende denkers* en HAVO'ers als *denkende doeners*. Er worden een aantal verschillen gehanteerd tussen de VWO en HAVO leerlingen.

- Het tempo bij de VWO leerlingen ligt iets hoger dan bij de HAVO leerlingen. Als gevolg behandelen de VWO leerlingen meer theorie dan de HAVO leerlingen.
- De HAVO leerlingen worden meer begeleid bij de opdrachten en krijgen vaak een stappenplan aangereikt. Deze stappenplan begeleidt ze bij het ontwerpen van een programma, zodat ze zich kunnen richten op het toepassen van de theorie. VWO leerlingen worden op weg geholpen maar krijgen geen stappenplan aangereikt.

4.2.2 Ontwerpen code

Zoals eerder aangegeven is het belangrijk dat leerlingen aandacht besteden aan het ontwerpen van hun code. Bij de eindopdracht is hier speciaal aandacht aan besteed. De leerlingen moeten eerst een formulier met hun code-ontwerp inleveren voordat ze verder kunnen met de opdracht.

4.2.3 Talentontwikkeling

Sommige leerlingen hebben al ervaring met programmeren of pakken het erg snel op. Voor deze leerlingen bieden we ruimte voor talentontwikkeling door ze een extra uitdaging te geven in de opdrachten.

4.3 Maatschappelijke relevantie

Tijdens de lessen is kort aandacht besteed aan hoe technologie en programmeren wordt gebruikt in de huidige samenleving. Vooral in de VWO klas is hier aan het begin van elke les aandacht aan besteed door een leuk voorbeeld te presenteren.

5 Evaluatiemethode

5.1 Eisen voor de evaluatie

In 3.4 is omschreven hoe de eisen voor het ontwerp gedestilleerd zijn uit de theorie, een aantal van deze eisen (toepassingsgerichtheid, begeleiding bij ontwerpen en visualisatie) hebben we alleen gebruikt om het ontwerp van de lesmodule vorm te geven. Deze onderdelen zullen wij niet toetsen, hier zijn twee redenen voor. Ten eerste was er geen controlegroep om deze eisen te toetsen en ten tweede hebben we door tijdsgebrek keuzes moeten over welke eisen we wel niet wilden toetsen. In tabel 2 wordt aangegeven hoe de verschillende eisen worden getoetst.

Table 2: Evaluatie eisen

Item/Meetmethode	**Expert**	**Enquête**	**Resultaten opdrachten**	**Eigen Observaties**
Differentiatie	x			x
Toegankelijkheid	x	x	x	x
Maatschappelijke relevantie		x	x	
Gepersonaliseerd leren	x			x
Plezier	x	x		x
Tevredenheid		x		x

5.2 Respondenten

Voor dit onderzoek hebben wij twee klassen aan respondenten gehad. Deze twee klassen waren de 4 HAVO- en de 4 VWO-klas van het Lyceum de Grundel. De leerlingen in deze klassen kregen de ontworpen module als onderdeel van hun informatica onderwijs.

Beide klassen hadden ongeveer hetzelfde niveau, zij begonnen net aan hun informatica onderwijs en hadden geen tot nauwelijks voorkennis. Hier zaten uitzonderingen bij, er zaten in beide klassen leerlingen die waren blijven zitten, en in de VWO-klas zaten een aantal leerlingen die als hobby al

eerder hadden geprogrammeerd. Tijdens het geven van de module is hier zo veel mogelijk rekening mee gehouden.

Kwa leeftijd zaten de leerlingen ook op ongeveer hetzelfde niveau. Beide klassen waren ook ongeveer hetzelfde kwa formaat (16 leerlingen in de VWO-groep, 20 in de HAVO-groep). Ook zaten in beide klassen overwegend meer mannelijke leerlingen, 11 tegen 5 in de VWO-groep en 18 tegen 2 in de HAVO-groep.

Zoals hierboven genoemd kregen deze leerlingen de ontworpen module als onderdeel van hun informatica onderwijs. Zij hadden hierin geen keuze om deze module niet te volgen. Dit is in overleg gegaan met de docent die verantwoordelijk was. De leerlingen gingen dat kwartiel toch aan de slag met programmeren, hierin hebben de onderzoekers erg veel vrijheid gekregen van de docent.

Er is tijdens deze module zo veel mogelijk rekening gehouden met de verschillen tussen (het niveau van) de klassen en de individuele leerling. Om deze reden hebben beide klassen hetzelfde materiaal gekregen, maar op een andere manier aangeboden gekregen.

5.3 Procedure

In oktober werd bekend dat de mogelijkheid zich aanbood dat een module programmeren ontworpen en gegeven werd op Lyceum de Grundel in de klassen zoals hierboven omschreven. Hiervoor zou een tijd van ongeveer acht weken beschikbaar zijn waarin de leerlingen werden verwacht te leren programmeren. Taal en methode waren vrij in te vullen.

Deze module is toen zo snel mogelijk ontworpen aangezien deze vanaf november gegeven moest worden. Hierin is zo veel mogelijk voorbereid, maar zijn er ook tijdens het geven van de module aanpassingen gedaan. De grootste aanpassing, zoals ook te lezen is in bijlage 11.1, is dat de originele planning is ingekort om rekening te houden met uitvallende lessen en onverwachte wijzigingen van de lessen.

Terwijl deze module werd gegeven zijn de resultaten van de leerlingen in de gaten gehouden door middel van wekelijkse opdrachten. De begeleidende docent heeft ook constant feedback geleverd waarop het lesontwerp is aangepast. Hiernaast is ook een externe expert gevraagd om feedback te geven op het ontwerp.

Aan het eind van de module is de leerlingen gevraagd om een enquête in te vullen. Deze enquête was gericht op de ervaringen van de leerlingen tijdens de module.

5.4 Instrumenten

Hierboven zijn de verschillende variabelen besproken die onderzocht zijn en gebruikt zijn om de lesmethode te ontwikkelen. Om de benodigde informatie te verzamelen worden de onderstaande methodes gebruikt. Deze methodes worden hieronder kort uitgelegd. In tabel 2 is een overzicht te zien waarin weergegeven wordt welke variabelen met welke methodes onderzocht zijn.

5.4.1 Expert

Een expert, Erik Barendsen, is gevraagd om de lesmethode te bestuderen en te beoordelen op een aantal punten. Hij is als expert gevraagd omdat hij een van de auteurs is van het nieuwe informatica curriculum Barendsen et al. (2016). De expert is door middel van open vragen gevraagd om te letten op de moeilijkheid, laagdrempeligheid, technisch niveau, bruikbaarheid en effectiviteit van het ontwerp. Ook is de expert gevraagd om daarnaast commentaar te leveren op wat hem verder is opgevallen. Dit heeft hij gedaan op basis van bijlage 11.1.

Daarnaast is ook de begeleidende docent, Wim Nijhuis, vanuit Lyceum het Grundel gevraagd om zijn professionele mening te geven over de module. Hierin is hij gevraagd om de ontworpen module te vergelijken met hoe hij anders programmeren had gegeven in deze klassen. Vanuit zijn ervaring, als docent en met deze klassen, geloven wij dat hij veel nuttige informatie en verdieping kan geven.

5.4.2 Enquête

De leerlingen hebben, na deelname aan de lesmodule, een enquête ingevuld. In deze enquête zijn vragen gesteld om de maatschappelijke context, moeilijkheidsgraad, laagdrempelighid, bruikbaarheid, tevredenheid en plezier die zij ervaren hebben tijdens hun deelname aan de lesmethode te onderzoeken. Het ontwerp van deze enquête is gebaseerd op de enquête die op de universiteit wordt gebruikt om vakken te evalueren. Van deze vragenlijst zijn een aantal vragen gekozen die aansluiten bij de beoogde onderdelen. Deze vragenlijst is uitgebreid met de System Usability Scale. (Brooke, 2013). Deze vragenlijst is ontworpen om een snelle indicatie te geven van de bruikbaarheid van een systeem. In dit onderzoek is deze vragenlijst gebruikt om de bruikbaarheid van de lesmodule programmeren te onderzoeken. Dit is niet waar de vragenlijst specifiek voor ontworpen is (het testen van het gebruik van systemen), dus de vragenlijst kan alleen een indicatie geven.

De enquête is toegevoegd als bijlage 11.2.

5.4.3 Eigen Observatie

Gedurende de module hebben wij de lessen gegeven en de opdrachten beoordeeld. Hierdoor hebben wij ook eigen observaties gedaan. Deze zullen wij meenemen als resultaat. Daarin zullen wij er wel op letten dat onze observaties subjectief zijn. Om deze reden zullen wij altijd meer dan een methode gebruiken om te rapporteren over de evaluatie eisen. Wel kunnen onze observaties conclusies versterken, uitleggen, of twijfels oproepen.

5.5 Analyse

5.5.1 Expert review

De opzet van de lesmodule (bijlage 11.1) is opgestuurd naar onze expert, Erik Barendsen. Onze expert heeft hierover via de mail feedback gegeven. De resultaten zullen per vraag besproken worden, hier zullen de relevante opmerkingen van meneer Barendsen meegenomen worden. De volledige reactie van meneer Barendsen is te vinden in bijlage 11.3.

Ook is onze begeleidend docent, Wim Nijhuis, van Lyceum de Grundel gevraagd om de lesmodule te beoordelen. Dit is gebeurd door hem, na de module, een mail te sturen met vragen erin. Op deze mail heeft meneer Nijhuis gereageerd met zijn antwoorden. Ook de relevante opmerkingen hieruit zullen per vraag besproken worden. De volledige reactie van meneer Nijhuis staat in bijlage 11.4.

5.5.2 Enquête

De analyse van de enquête wordt opgedeeld in drie onderdelen. Hieronder wordt per onderdeel uitgelegd hoe deze geanalyseerd zal worden. De analyse zal per klas uitgevoerd worden omdat de aanpak per klas erg verschilde.

Kwaliteitsvragen - Likert Voor veel vragen is 5 punt Likert-type schaal gebruikt. Deze vragen zullen beoordeeld worden op een kwantitatieve manier. Er wordt gekeken naar de gemiddelde score

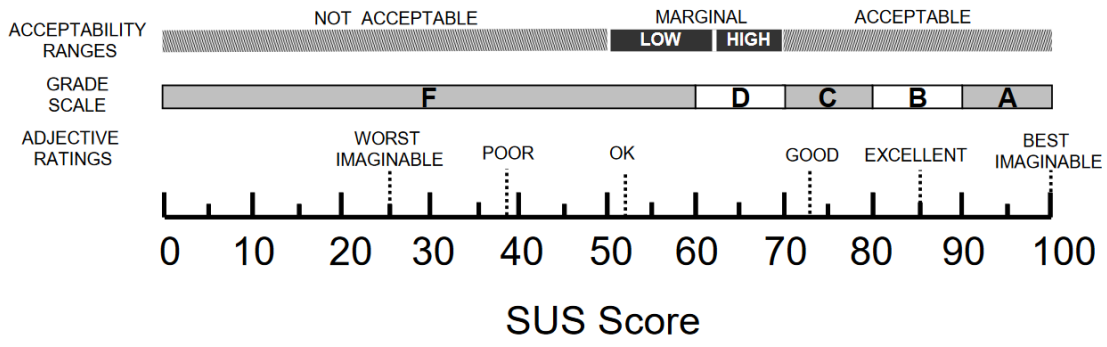
per vraag en het meest veelvoorkomend antwoord. Per vraag wordt dan gekeken naar wat de betekenis van een score is. Bij vragen waar er gevraagd wordt of leerlingen het eens of oneens zijn met een stelling wordt een gemiddelde score van 3,5 (beetje mee eens) als een succes gezien. Bij een stelling waarbij de leerlingen aangeven of iets te laag of te hoog was (bijvoorbeeld, het niveau van de module), wordt een score van 3 als ideaal genomen. Hoe dichter bij deze score, hoe succesvoller deze vraag scoort. Hierbij wordt alles tussen de 2 en de 4 als een succes meegenomen. Er wordt per vraag niet alleen gekeken naar het gemiddelde, ook de modus en het bereik worden besproken.

Een algemeen overzicht van de antwoorden zijn te vinden in bijlage 11.5.5 voor HAVO en bijlage 11.5.6 voor vwo. Per onderzoekseis zullen de relevante vragen behandeld worden.

Kwaliteitsvragen - Open De antwoorden op deze vragen zullen beoordeeld worden op een kwalitatieve manier. Hierbij wordt gekeken naar de verschillende resultaten en elk antwoord als individueel antwoord behandeld. Er wordt aandacht besteed aan constructieve feedback of vragen van de leerlingen. Al het commentaar ingeleverd door de leerlingen zal bekeken en meegenomen worden. De resultaten hiervan zullen per onderzoekseis worden gepresenteerd.

SUS De System Usability Scale (SUS) heeft een gespecificeerde manier om de resultaten te analyseren. (Bangor, Kortum, & Miller, 2008; Brooke, 2013) Deze manier zal aangehouden worden. Uit Bangor, Kortum, and Miller (2009) komt figuur 1, deze zal gebruikt worden om aan te geven wat de scores betekenen. De resultaten hiervan zullen onder onderzoekseis Plezier verder uitgewerkt worden.

Figure 1: SUS resultaten analyseren uit Bangor et al. (2009)



5.5.3 Resultaten Opdrachten

De opdrachten die ingeleverd zijn door de leerlingen zullen beoordeeld worden op begrip van de maatschappelijke relevantie en de toegankelijkheid van de lesmodule.

Maatschappelijke relevantie Tijdens de eindopdracht werden de leerlingen gevraagd om ook aan te geven wat de maatschappelijke context was van hun opdrachten en hoe dit hun opdracht heeft beïnvloed. Deze uitleg zal hierin meegenomen worden.

Toegankelijkheid De leerlingen hebben een korte en intensieve cursus gehad. Vooral in de eerste weken zal te zien zijn of leerlingen makkelijk mee konden komen met de opdrachten (laagdrempeligheid). De tussentijdse opdracht van de leerlingen zal bekeken worden om de toegankelijkheid te bepalen. Hierin wordt goed gekeken naar of de leerlingen de opdracht konden vervullen en hoe moeilijk dit voor ze was.

6 Resultaten

In tabel 2 staat een overzicht van welke resultaten we voor elk onderdeel gaan presenteren. Hierin is een poging gedaan om het overzichtelijk te houden ondanks de grote variatie aan soorten data. Om deze reden zijn de interviews met de experts en de grafieken verplaatst naar bijlagen 11.3 en 11.4 voor de experts en bijlage 11.5 voor de enquête resultaten.

6.1 Differentiatie

De lesmodule moest rekening houden met de verschillen tussen HAVO en VWO.

6.1.1 Expert reviews

Erik Barendsen

Deze expert geeft aan dat het goed is dat de HAVO-versie van de module niet afhankelijk is van lastigere programmeerconcepten die wel in VWO zijn behandeld. Verder geeft hij aan dat er geprobeerd is om op een "kansrijke manier" te differentiëren.

Wim Nijhuis

Deze expert geeft aan dat de VWO-versie van de module meer uitdaging aan had gekund.

6.1.2 Eigen Observaties

HAVO

Er waren een aantal verschillen te observeren tussen de HAVO en VWO leerlingen. Daarom is uit observatie besloten om de modules aan te passen voor de twee groepen.

De HAVO-leerlingen leken minder goed om te kunnen gaan met vrijheid rondom de opdrachten. Als de leerlingen een opdracht kregen waarin ze zelf mochten kiezen wat ze wilden maken, hadden veel leerlingen moeite om om te gaan met deze vrijheid. Twee redenen hadden hiermee te maken. Ten eerste, sommige leerlingen waren niet gemotiveerd om te werken. Als gevolg misbruikten ze de vrijheid om weinig in te leveren. In hun ogen hadden ze de vrijheid om te doen wat ze wilden, en dus deden ze zo min mogelijk. Ten tweede wisten sommige leerlingen niet goed wat ze moesten doen. Hoewel ze kennis hadden van de programmeerprincipes, wisten ze niet goed wat er van hen verwacht werd en wat ze moesten maken.

Om deze redenen hebben de HAVO-leerlingen een concreet handvat gekregen in de vorm van een stappenplan voor de opdrachten. De leerlingen moesten nog steeds wel nadenken over de implementatie en organisatie, maar hoefden minder na te denken over de macro-organisatie van de programma's.

Verder was initieel geprobeerd om het tempo van VWO aan te houden, maar al snel bleek dat dit niet mogelijk zou zijn. De leerlingen hadden meer tijd nodig om de concepten te verwerken. Daarom is in de VWO klas meer theorie behandeld dan in de HAVO-klas.

VWO

De leerlingen moesten duidelijk wennen aan de open invulling van de opdrachten. Voor sommige leerlingen betekende dit dat ze niet zo goed wisten wat ze moesten doen, waardoor er niet zo veel gebeurde. Andere leerlingen waren juist erg blij met de vrijheid en gingen gelijk aan de slag met de wildste ideeën. Deze tweedeling zorgde ervoor dat de docent aan de slag kon met de leerlingen die er meer moeite mee hadden terwijl de andere leerlingen lekker bezig waren.

Terwijl de leerlingen in de eerste weken steeds bezig waren met het uitbreiden van hun initieel ontwerp, waren er leerlingen die na een aantal weken klaar waren met hun ontwerp en graag met iets nieuws verder wilden. Deze vrijheid was er voor deze leerlingen.

Inhoudelijk was er veel vrijheid bij de VWO groep. De leerlingen pakten het programmeren snel op en er was wat tijd om extra aandacht te besteden aan de leerlingen die extra begeleiding nodig hadden. Hierdoor kon het tempo hoger liggen dan in de HAVO groep. Aan het eind van de module is een blok theorie optioneel aangeboden aan de leerlingen, leerlingen die graag die uitdaging aan wilden gaan konden aan de slag met functies schrijven. De andere leerlingen waren nog bezig met hun opdrachten af maken. Deze manier van differentiatie was erg fijn voor de stagiaire.

Met differentiatie is meer gedaan tijdens deze lesmodule, er waren in de VWO groep leerlingen die al ervaring hadden met programmeren. Met deze leerlingen is een andere planning gemaakt met eigen uitdagingen. Deze leerlingen hebben voor de kerstvakantie hun eindopdracht gedaan, en zijn na de kerst verder gegaan met een fysieke aansturing van hun eindopdracht met Arduino. Hierdoor waren deze leerlingen ook actief bezig.

6.2 Toegankelijkheid

Binnen dezelfde klas kunnen leerlingen een verschillende wiskundige achtergrond hebben en de module moet toegankelijk zijn voor leerlingen die minder sterk zijn in wiskunde.

6.2.1 Expert reviews

Erik Barendsen

Deze expert is niet specifiek op dit onderwerp ingegaan.

Wim Nijhuis

Wim Nijhuis heeft de indruk dat de leerlingen voldoende begrip hebben voor de basisprincipes van programmeren. Ook vind deze expert de combinatie van programmeren en creativiteit mooi. Hij vind de module niet vergelijkbaar met voorgaande jaren, toen was programmeeronderwijs meer vanuit de beta-kant benaderd. De opzet van de module vind hij verfrissend. Wel vind de expert dat er een vervolg op moet komen, het is een introductie die ook gedaan kan worden door leerjaar 2, en meer diepgang is gewenst.

6.2.2 Enquête

Gesloten vragen

Twee van de gestelde vragen, "Het niveau van de module was te laag/te hoog" en "Ik had voldoende voorkennis om deze module te volgen", sloten aan bij toegankelijkheid, dit waren de vragen over het niveau van de module en of de module goed aansloot bij de voorgaande kennis. In figuur 3 staat

een overzicht van de antwoorden voor de HAVO en de VWO groep. In de HAVO groep vonden de leerlingen het niveau goed tot hoog, bij de VWO leerlingen was dit goed en voor een enkele leerling te hoog. De antwoorden voor de vraag over voorkennis waren redelijk verspreid bij de HAVO groep, en middel tot mee eens voor de VWO groep.

Open vragen

De open vragen gaven geen informatie specifiek op het gebied van toegankelijkheid.

6.2.3 Resultaten opdrachten

De resultaten van de opdrachten is een indicatie van de toegankelijkheid. Er zijn twee opdrachten die hierbij een rol spelen: de tussentijdse opdracht en de eindopdracht. De resultaten per leerling van de opdrachten staan in figuren 4a en 4b. Tijdens de module hadden leerlingen ook huiswerkopgaven, wanneer deze opdrachten niet werden ingeleverd werd 0.5 punten van het eindcijfer afgetrokken. De figuren bevatten ook de cijfers zonder deze aftrek.

Tussentijdse Opdracht

In de HAVO-klas hebben 7 van van de 20 leerlingen een onvoldoende gescoord en hiervan hadden 4 leerlingen de opdracht niet ingeleverd. Van de 3 leerlingen die het wel hebben ingeleverd hebben 2 een voldoende gehaald voor de eindopdracht.

In de VWO-klas hebben alle leerlingen een voldoende gehaald.

Eindopdracht

In de HAVO-klas heeft 8 leerlingen een onvoldoende gehaald voor de eindopdracht, waarvan 1 leerling het niet heeft ingeleverd.

In de VWO-klas hebben alle leerlingen een voldoende gehaald voor de eindopdracht, waarbij driekwart van de leerlingen hoger dan een 7.5 heeft gescoord.

6.2.4 Eigen Observaties

HAVO

De module was ontwikkeld met het idee dat het toegankelijk moet zijn voor leerlingen die geen voorkennis hebben in programmeren. Uit observatie is gebleken dat leerlingen weinig moeite hadden om de eerste paar regels te coderen. Hoewel ze soms een aantal syntactische details vergaten tijdens het programmeren, zoals het afsluiten van een regel code met een puntkomma, begrepen ze hoe ze simpele programma's moesten maken.

Bij de eerste huiswerkseries waren de leerlingen vrij gelaten om zelf te bedenken wat ze wilden programmeren. De leerlingen gingen niet goed om met deze vrijheid en begonnen al snel te experimenteren met code die ze bijvoorbeeld online vonden. Andere leerlingen misbruikten deze vrijheid om zo min mogelijk te doen. Ook was opgemerkt dat leerlingen niet altijd wisten hoe ze moesten beginnen met een programma en hoe ze meerdere technieken konden samenvoegen om een één coherent programma te maken. Nadat de leerlingen een stappenplan was gegeven ging het veel beter. Ze hadden meer richting en wisten wat er van hen verwacht werd.

Er waren een aantal leerlingen die alsnog moeite hadden of niet precies wisten wat er moest gebeuren. Initieel dachten we dat deze leerlingen extra moeite hadden, maar later bleek dat ze tijdens de practica afgeleid waren door andere dingen. Het was dus niet dat ze moeite hadden met programmeren per se, maar moeite hadden met motivatie of concentratie.

Bij de eindopdracht hadden sommige leerlingen moeite om op gang te komen. Hoewel ze de individuele concepten beheersten, wisten ze niet helemaal zeker hoe ze deze concepten moesten toepassen om een groter programma te ontwerpen en te programmeren. We moesten vaker tips geven over hoe ze het beste konden beginnen.

VWO

Tijdens de eerste lessen viel op dat bijna alle leerlingen goed meekwamen in de lessen. Er waren er maar een aantal die moeite hadden met de stof. Door de zelfstandige opbouw van de lesmodule was er veel ruimte voor de docent om de tijd te pakken om met de leerlingen die het ingewikkelder vonden te gaan zitten. Bij een specifieke leerling heeft de docent de leerling gevraagd om extra materiaal te lezen. Hierna ging het de leerling een stuk makkelijker af.

Gedurende de module waren veel leerlingen goed bezig. Hierin was het effect van positieve feedback erg belangrijk. De leerlingen die veel positieve feedback kregen, vaak de leerlingen die het in het begin wat lastiger hadden, werden hierdoor extra gemotiveerd waardoor ze ook veel lieten zien.

6.3 Maatschappelijke relevantie

Initieel was er het idee dat tijdens de lessen het belang van programmeren benadrukt zou worden in de lessen en tijdens de opdrachten. In de VWO lessen is aan het begin van elke les hier een aantal minuten aandacht aan besteed. Zo zijn bijvoorbeeld filmpjes gepresenteerd van maatschappelijke toepassingen waar programmeren een grote rol bij speelt.

In de HAVO-klas is hier minder aandacht aan besteed. Ook bij de opdrachten bij beide klassen is dit niet aan bod gekomen. Oorspronkelijk was het plan dat de leerlingen moesten uitleggen wat de maatschappelijke relevantie van hun project was. Dit is niet gebruikt, de leerlingen hebben in plaats van focussen op een maatschappelijk probleem (een initieel idee) een spel gemaakt. Hiervoor is gekozen omdat dit concept (een spel) een stuk dichter bij de belevingswereld van de leerlingen is, hierdoor was het makkelijker voor de leerlingen om mee aan de slag te gaan.

6.3.1 Enquête

Twee van de vragen uit de enquête sloten aan bij de maatschappelijke relevantie van de module. De leerlingen hebben aangegeven hoe oneens (licht) of eens (donker) ze het waren met de stellingen "Deze module helpt mij te begrijpen hoe programmeren terugkomt in de samenleving." en "Ik heb een idee van wat ik met programmeren zou kunnen doen.". De resultaten van deze vragen staan in figuur 5. In beide groepen werd er rond het midden gescoord op de vraag over programmeren in de samenleving, waarbij HAVO positiever was dan VWO. Op de vraag "Ik heb een idee van wat ik met programmeren zou kunnen doen" werd er redelijk positief geantwoord, waarbij de VWO groep positiever was dan de HAVO groep.

6.3.2 Resultaten Opdrachten

Zoals aangegeven is hier in de opdrachten geen aandacht aan besteed. Daarom is dit niet terug te zien in de resultaten van de opdrachten.

6.4 Gepersonaliseerd leren

6.4.1 Expert review

Erik Barendsen

In de HAVO-klas zijn de opdrachten stapsgewijs voorgedragen aan de leerlingen. Erik Barendsen geeft aan dat stapsgewijze opdrachten misschien niet voor iedereen zal werken. Het is een aanbeveling om extra reflectie in te bouwen in de opdrachten. Leerlingen moeten dan nadenken over wat er gaat gebeuren, wat er gebeurt op dit moment en waarom een bepaalde resultaat naar voren is gekomen. Dit voorkomt dat leerlingen niet blindelings de stappen volgen, maar ook actief nadenken over de opdrachten.

Wim Nijhuis

Deze expert gaat niet specifiek in op dit onderwerp.

6.4.2 Eigen Observaties

HAVO

Zoals beschreven in sectie 4.2 is gepersonaliseerd leren gefaciliteerd op drie verschillende manieren. Leerlingen konden leren door de klassikale lessen bij te wonen, gebruik te maken van de theorie die samengevat op Its Learning stond, en zelfstandig de website door te nemen. Uit observatie bleek dat de meeste leerlingen hier geen gebruik van maakten, leerlingen leerden vooral uit de klassikale lessen en raadpleegden de theorie zelden. Er waren 2 of 3 leerlingen die gebruik maakten van de *Processing* website om te experimenteren met nieuwe technieken.

Hoewel verschillende leerstijlen waren gefaciliteerd, maakten de leerlingen niet overal gebruik van.

VWO

Tijdens de lessen werd er elke les klassikaal uitleg gegeven. Daarnaast werden de leerlingen tijdens de lessen aangespoord om zelfstandig op zoek te gaan naar informatie. Wanneer leerlingen bij de docent kwamen met vragen die de leerlingen ook zelfstandig konden opzoeken, spoorde de docent de leerlingen actief aan om dit op te zoeken. Dit deed de docent door vragen terug te stellen. Hierdoor werden de website van Processing en Google ook redelijk vaak gebruikt in de les. De samenvatting van de theorie op papier is weinig bekeken tijdens de les.

6.5 Plezier

6.5.1 Expert Review

Erik Barendsen

Deze expert is niet specifiek op dit onderwerp ingegaan.

Wim Nijhuis

Deze expert geeft aan dat de lessen ongedwongen waren en dat leerlingen het erg leuk hebben gevonden.

6.5.2 Enquête

Twee van de gestelde vragen sloten aan bij het plezier van de leerlingen, de eerste was "Ik vond het leuk om te leren programmeren." en de tweede was "Ik vond het leuk om aan de opdrachten te werken.". De resultaten van de leerlingen uit de verschillende groepen staan in figuren 6a en 6b weergegeven.

Deze resultaten laten zien dat in zowel HAVO als VWO er 1 leerling was die het niet leuk vond om aan de opdrachten te werken. Tweederde van de leerlingen vond het leuk om te leren programmeren en om aan de opdrachten te werken. De rest van de leerlingen is hier neutraal over.

6.5.3 Eigen Observaties

HAVO

Er was veel verschil tussen de leerlingen en hoeveel plezier ze beleefden uit de module. Een grote groep leerlingen vond het erg leuk om te programmeren en ze waren hier enthousiast over. Deze leerlingen probeerden creatieve programma's te maken en wilden meer concepten verwerken dan in de les is besproken.

Andere leerlingen waren niet helemaal gemotiveerd om te werken. Soms waren ze niet aanwezig in de lessen en op andere momenten waren ze niet aan het opletten. Deze leerlingen hadden ook vaak hun huiswerk niet ingeleverd.

Een kleine groep leerlingen was vrij neutraal tegenover de module. Ze maakten de opdrachten allemaal, maar lieten niet zien dat ze hier extra enthousiast over waren.

Van de HAVO-klas was het lastig om feedback te krijgen, ze waren tijdens de uitleg soms erg stil en lieten niet veel van zich horen. Daarom was het af en toe lastig om hun emoties te peilen.

VWO

Over het algemeen hebben de leerlingen het leuk gehad tijdens de les. Veel leerlingen kwamen trots laten zien wat ze hadden gemaakt en hoe ze ervoor stonden. Ook de leerlingen die vooruit konden werken waren blij met wat ze gemaakt hadden en vonden het leuk om andere leerlingen te helpen.

Bijzonder opvallend was een leerling die programmeren in het begin van de module erg lastig vond. Deze leerling was aan het eind erg positief over zijn/haar eindproduct en daarnaast ook erg tevreden over de gehele module, ondanks dat het toch wel een lastig onderwerp was voor deze leerling.

Wat minder plezierig werd ervaren door de leerlingen was de constante tijdsdruk, deze tijdsdruk ontstond doordat leerlingen elke week een opdracht moesten inleveren. Veel leerlingen werden hierdoor soms ontmoedigd.

6.6 Tevredenheid

6.6.1 Enquête

Vraag

Een van de vragen in de enquête vroeg de leerlingen of ze aan wilden geven hoe oneens of eens de leerlingen waren met de stelling: de lessen waren goed. De resultaten van de antwoorden staan in figuur 7 weergegeven in de bijlage. Over het algemeen waren de leerlingen positief over deze vraag, waarbij de HAVO leerlingen een stuk positiever waren dan de VWO leerlingen.

Rapportcijfer Module

Tijdens de enquête zijn de leerlingen ook gevraagd om aan te geven welk rapportcijfer ze de module zouden geven. In figuur 8 staat een overzicht van de antwoorden van de leerlingen. Het rapportcijfer ligt tussen de 5 en de 9 voor HAVO en tussen 6 en 9 voor VWO.

SUS Score

Zoals omschreven in hoofdstuk 5.5.2 is door middel van een vragenlijst van 10 vragen, die elke leerling heeft ingevuld, een SUS score bepaald. In figuur 9 staat een overzicht van de SUS scores voor de havo en de VWO-groep. De SUS scores liggen tussen 25 en 80 voor HAVO en tussen 30 en 85 voor VWO.

6.6.2 Eigen Observaties

HAVO

Aan het begin van de module moesten de leerlingen relatief eenvoudige programma's maken. Bij het tekenen van een vierkant op het scherm wisten ze niet altijd waar dit voor diende. Dus hoewel het hen lukte om de opdracht uit te voeren, waren ze niet zeker hoe ze het resultaat konden toepassen. Pas in de latere lessen, wanneer ze interactieve programma's begonnen te maken, kregen ze een gevoel van voldoening en tevredenheid. Leerlingen begonnen toen een eigen draai aan hun programma's te geven en vroegen om tips om het creatiever te maken.

Leerlingen die meer moeite hadden raakten soms gefrustreerd omdat het hen niet altijd lukte. Echter, aan het einde van de module waren de meeste leerlingen in staat om een spelletje te maken in *Processing*. De laatste opdracht viel helaas in de toetsweek van de leerlingen, waardoor sommige leerlingen gefrustreerd raakten vanwege de hoge werkdruk. Desondanks waren ze blij met wat ze konden bereiken en over het algemeen tevreden met wat ze hadden geleerd.

VWO

De leerlingen hebben het tijdens de module erg afwisselend gehad. Er was veel frustratie wanneer het niet lukte, maar ook veel trots en tevredenheid wanneer de opdrachten succesvol afgerond konden worden. Hierin heeft de probleem-gerichte aanpak van de lesmodule erg geholpen, doordat

er veel korte opdrachten waren hadden de leerlingen het gevoel dat ze successen hadden. Deze kleine overwinningen hebben denk ik erg geholpen met het overall gevoel wat de leerlingen hadden over deze module. Tijdens de lessen waren er meerdere momenten (per les) waarin een leerling trots wilde laten zien wat hij/zij had gemaakt of werkend had gekregen.

7 Discussie

7.1 Analyse

7.1.1 Differentiatie

Op het gebied van differentiatie geven de experts twee verschillende dingen aan. Wim Nijhuis geeft aan dat de VWO leerlingen meer uitdaging aangekund zouden hebben. Deze opmerking wordt enigszins weerspiegeld in de cijfers van de VWO leerlingen, waarin alle leerlingen voor alle opdrachten een voldoende hebben gehaald.

Erik Barendsen geeft aan dat de module genoeg ruimte biedt voor differentiatie en dat het goed is dat de HAVO-leerlingen niet afhankelijk zijn van lastigere concepten. In de eigen observatie was ook inderdaad gebleken dat HAVO-leerlingen meer tijd nodig hadden dan de VWO leerlingen om programmeerconcepten te begrijpen.

Verder is er uit de observatie naar voren gekomen dat de VWO leerlingen beter om konden gaan met de gegeven vrijheid dan de HAVO-leerlingen. De HAVO-leerlingen hadden dus meer begeleiding nodig. Deze begeleiding komt terug in de module in de vorm van een stappenplan.

Uit deze resultaten blijkt dat de VWO klas iets meer uitgedaagd kan worden door ze bijvoorbeeld moeilijkere opdrachten te geven. Verder is het een goed idee geweest om de HAVO-klas niet afhankelijk te laten zijn van complexere programmeerconcepten zoals functies met parameters. Het is ook goed dat het tempo van de HAVO-klas lager ligt dan de VWO klas, want zij hebben meer tijd nodig om de concepten te verwerken.

Hieruit kan geconcludeerd worden dat differentiatie een positieve invloed heeft gehad op de leerlingen. Dit onderdeel kan verbeterd worden door het niveau voor VWO beter aan te passen aan de leerlingen.

7.1.2 Toegankelijkheid

Erik Barendsen heeft geen commentaar gegeven op dit gebied, maar Wim Nijhuis heeft hier uitgebreid op gereageerd. Wim Nijhuis geeft aan dat de module zeer toegankelijk is voor de leerlingen. Hierbij loert het gevaar dat het te makkelijk wordt voor de leerlingen als het *te* toegankelijk is. In de vorige subsectie is hier al het een en ander over opgemerkt, waarbij is aangegeven dat de VWO klas meer uitdaging aangekund had.

Uit de enquête is er een verschil te zien tussen de VWO en de HAVO-klas. De meerderheid van de VWO-leerlingen geeft aan dat het niveau van de module niet te hoog en niet laag lag en dat ze geen voorkennis misten om de module te volgen. Daarentegen zijn de meningen van de HAVO-klas iets verdeeld. De meerderheid van de leerlingen geeft aan dat de module op niveau was, of zelfs iets aan de hogere kant. Dit lijkt aan te geven dat ze voldoende zijn uitgedaagd. De vraag of ze genoeg voorkennis hadden is gelijk verdeeld over de leerlingen.

Uit de resultaten van de opdrachten blijkt ook inderdaad dat de VWO leerlingen geen onvoldoendes hebben gehaald. Dit komt overeen met de opinie van de leerlingen dat de module niet

te moeilijk was. Het geeft ook aan dat leerlingen niet onderuit zijn gegaan vanwege een tekort aan voorkennis zoals ook blijkt uit de eigen observatie. Hieruit blijkt dat VWO leerlingen genoeg voorkennis hebben gehad om de module te volgen.

Bij HAVO-leerlingen hebben een aantal leerlingen aangegeven dat ze het minder toegankelijk vonden en voorkennis misten om de module te volgen. Daarentegen vonden de meeste leerlingen het niveau van de module niet te hoog. Het kan niet meteen geconcludeerd worden dat leerlingen de module niet toegankelijk vonden. Uit de cijfers van de opdrachten blijkt ook dat leerlingen de tussentijdse opdracht voldoende hebben afgerond. Slechts 3 leerlingen hebben feitelijk een onvoldoende gehaald hiervoor en van deze leerlingen hebben 2 leerlingen een voldoende gehaald voor de eindopdracht, wat laat zien dat ze het basisniveau beheersen.

Uit de observatie is ook gebleken dat een aantal leerlingen niet actief hebben meegedaan met de module en hierdoor hebben zij moeite gehad met de opdrachten. Voor deze leerlingen zal gelden dat ze de module moeilijk hebben gevonden. Bovendien hebben de experts niet aangegeven dat de module niet toegankelijk is voor de HAVO-leerlingen.

Uit deze analyse kan geconcludeerd worden dat leerlingen de module als toegankelijk hebben ervaren.

7.1.3 Maatschappelijke relevantie

Aan maatschappelijke relevantie is in de VWO klas aandacht besteed door middel van filmpjes. De stagiaire en later de leerlingen presenteerden filmpjes van producten of maatschappelijke diensten waarin programmeren een grote rol speelt. De stagiaire, Heleen, vond dit een belangrijk onderdeel van de module onder andere omdat ze een socio-technische achtergrond heeft.

In de HAVO-klas is het niet gelukt om dit te benadrukken. De stagiair, Anirudh, heeft hier minder aandacht aan besteed en heeft zich meer gefocust op de technische kant. Dit onder andere omdat hij een technische achtergrond heeft. Dit is ook weerspiegeld in de enquête. Een aantal leerlingen geven aan te weten hoe programmeren terug komt in de samenleving, echter de meerderheid heeft moeite om dit te formuleren of weet hier niks over te zeggen. Veel leerlingen geven echter wel aan te weten wat ze met programmeren zouden kunnen doen.

In de opdrachten is de maatschappelijke relevantie niet naar boven gekomen, er zijn dus geen opdrachten ontwikkeld waar leerlingen een programma moeten maken waar de samenleving nut van heeft. Dit is eventueel een verbeterpunt voor de volgende keer. Er moet echter wel rekening mee gehouden worden dat de opdrachten enigszins vallen in de belevingswereld van de leerlingen. Maatschappelijk belang is niet voor alle leerlingen even belangrijk. Daarentegen vinden ze over het algemeen actiespelletjes wel weer interessant. Er is dus altijd een balans die gevonden moet worden tussen in hoeverre het maatschappelijke belang wordt benadrukt en in hoeverre de opdrachten rekening moeten houden met de belevingswereld van de leerlingen. Als er alleen gefocust wordt op spelletjes dan krijgen leerlingen een eenzijdig beeld over programmeren en denken ze dat programmeren alleen wordt toegepast in informaticagerelateerde applicaties. Daarentegen als er alleen wordt gefocust op maatschappelijke relevantie, dan kunnen de leerlingen interesse verliezen in informatica en programmeren.

Interessant zou zijn om te kijken naar naar een overlap van beide aspecten. Een toepassing hiervan is bijvoorbeeld 'serieus gaming'. Op deze manier is het mogelijk om beide aspecten aan bod te laten komen.

Aan de andere kant, zoals bij VWO naar voren is gekomen, lijkt het niet noodzakelijk om maatschappelijke relevantie in de opdrachten te integreren, maar is het genoeg om het kort aan te

stippen tijdens de presentaties, bijvoorbeeld door middel van video's. Voor de leerlingen was dit genoeg om een beeld te krijgen over hoe programmeren ingezet kan worden en hoe het terugkomt in de samenleving.

Er kan geconcludeerd worden dat maatschappelijk belang van programmeren benadrukt kan worden door voorbeelden te laten zien in de klas en door het te integreren in de uitleg. De resultaten laten zien dat door hier aandacht aan te besteden het degelijk een verschil maakt voor de leerlingen. In de HAVO-klas was hier minder aandacht aan besteed, wat dus een verbeterpunt is voor de volgende iteratie van de module.

7.1.4 Gepersonaliseerd leren

In de eigen observaties wordt bij VWO en HAVO aangegeven dat leerlingen het meeste gebruik maakten van de klassikale presentaties om te leren. Ze maakten dus niet vaak gebruik van de theorie of de website.

Dit kan een aantal redenen hebben. Ten eerste kan het laten zien dat leerlingen genoeg hadden aan de theorie die klassikaal was uitgelegd en dat alle leerlingen dit een prettige methode vonden om te leren. Dit is echter zeer dubieus, want het is onwaarschijnlijk dat alle leerlingen alles hebben geleerd van de klassikale uitleg.

Wat waarschijnlijker is, is dat leerlingen niet precies wisten hoe ze de theorie of website moesten raadplegen en konden gebruiken tijdens het maken van hun programma's. Dit is namelijk niet expliciet in de lessen uitgelegd of in de opdrachten teruggekomen.

De expert geeft daarnaast ook aan dat de stapsgewijze methode om opdrachten te maken in de HAVO-klas misschien niet voor iedereen zou werken en dat het beter is om reflectie in te bouwen in de opdrachten. Voor de volgende versie is het daarom een idee om vragen aan de opdrachten toe te voegen zodat leerlingen actief moeten nadenken over wat ze implementeren.

Al met al, de module kan verbeterd worden op twee vlakken. Ten eerste kan er in de les uitgelegd worden hoe leerlingen gebruik kunnen maken de verschillende bronnen van informatie. Ten tweede kunnen de opdrachten op verschillende wijzen worden gepresenteerd zodat het de verschillende leerstijlen faciliteert. Dat laatste is al ingevoerd door HAVO-leerlingen een stappenplan aan te bieden, dit kan verder uitgewerkt worden voor individuele personalisatie.

7.1.5 Plezier

Uit observaties, expert review en de enquête blijkt dat leerlingen de module over het algemeen positief hebben ervaren. Uit de HAVO-observatie blijkt dat sommige leerlingen minder gemotiveerd waren om te werken, desondanks hebben geen leerlingen aangegeven dat ze het niet leuk vonden om te programmeren. Er was 1 leerling bij HAVO en 1 leerling bij VWO, die hebben aangegeven dat het minder leuk was om aan de opdrachten te werken.

Hieruit kan geconcludeerd worden dat de leerlingen de lesmodule over het algemeen als plezierig hebben ervaren.

7.1.6 Tevredenheid

We hebben tevredenheid beoordeeld door de leerlingen te vragen naar hun mening. Ze zijn gevraagd om de module te beoordelen door middel van een rapportcijfer en een SUS score. Het gemiddelde rapportcijfer van HAVO en VWO is 7,2. De spreiding bij de HAVO-klas is iets groter dan bij VWO. In de HAVO-klas waren er 3 leerlingen die de module een 5 hebben gegeven. Deze leerlingen vonden

vooral dat de module te lastig was. Hun commentaar was: "Niveau een stukje omlaag doen" en "Minder moeilijk". Deze leerlingen hebben tevens ook aangegeven dat het niveau van de module te hoog was (een score van 4 op een schaal van 1 - 5) en dat ze niet genoeg voorkennis hadden. De leerlingen die gemiddeld de module een 9 hebben gescoord, geven daarentegen aan dat ze voldoende voorkennis hadden om de module te volgen en vonden het niveau niet te hoog. Leerlingen die de module laag hebben gescoord hadden moeite met het niveau.

Bij VWO geldt praktisch hetzelfde waar één leerling de module een 6 heeft gegeven. Deze leerling geeft aan geen voorkennis te hebben om de module te volgen.

Leerlingen zijn over het algemeen dus tevreden met de module. De leerlingen die minder tevreden waren vonden de module minder toegankelijk.

7.1.6.1 SUS Score

De SUS Scores staan in figuur 9, hieronder zijn deze toegevoegd aan figuur 1, zoals omschreven in de methode. Zoals te zien is, zijn de scores voor de module erg laag. Met een gemiddelde van 57 hebben de VWO leerlingen de module net "OK" gescoord. De HAVO groep was met een gemiddelde van 49 iets minder positief over de module. De meningen van de leerlingen lopen ver uiteen (van ergst mogelijk tot goed bij HAVO en van zwak tot excellent bij VWO), hier kan uit geconcludeerd worden dat er veel te verbeteren valt volgens de leerlingen.

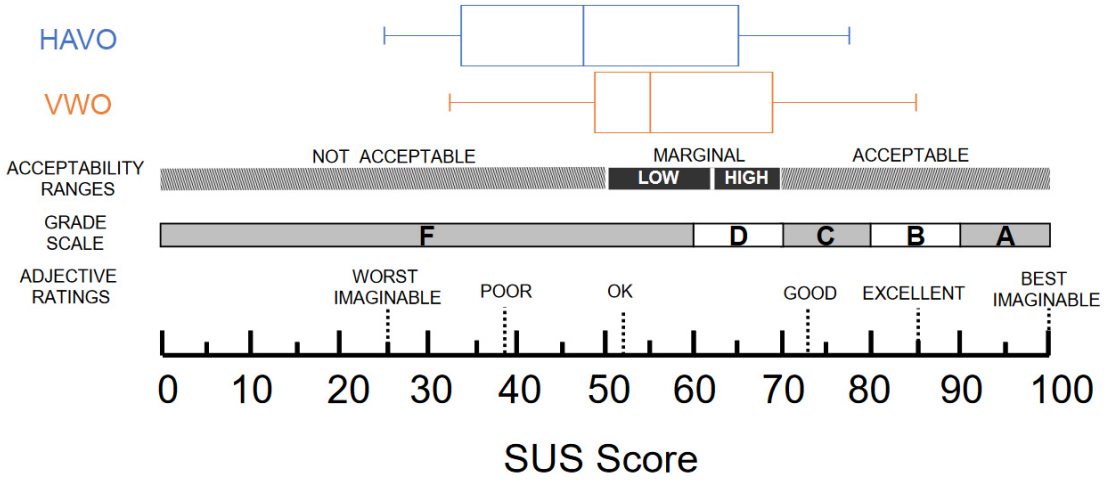


Figure 2: De SUS Scores voor HAVO (blauw) en VWO (oranje) in de SUS Scale

De SUS methode is niet de handigste manier om tevredenheid te bepalen voor onze module. Bijvoorbeeld de statement "Ik moest veel leren voordat ik aan de slag kon met programmeren.", voor een hoge SUS score wordt een lage Likert score verwacht. Veel leerlingen geven een redelijk hoge score aan deze vraag, wat volledig acceptabel is voor het leren programmeren. Programmeren is niet iets wat veel mensen intuïtief oppakken. Dit geldt ook voor de vraag "Ik denk dat ik de hulp van iemand anders nodig zal hebben bij het programmeren.", het is volledig acceptabel dat de leerlingen nog begeleiding nodig hebben. Ze hebben pas een aantal weken kunnen wennen aan programmeren, en het is logisch dat ze nog wat begeleiding nodig zullen hebben voordat ze volledig

zelfstandig aan de slag kunnen. Het is ook erg lastig om op deze vraag een oordeel te vellen, sommige leerlingen vinden het juist fijn om die begeleiding te krijgen, terwijl anderen juist zelfstandig aan de slag willen. Voor een hoge SUS score, wordt op deze vraag een lage score verwacht (dicht bij 1), terwijl dat niet per se gewenst is wat betreft deze vraag.

Hierdoor kunnen we concluderen dat de SUS scores negatief uitvallen vergeleken met het rapportcijfer van de leerlingen. Dit komt waarschijnlijk doordat de SUS score niet geschikt is voor het meten van het succes van een onderwijssysteem.

7.1.7 Vakinhoudelijke voortgang

Daarnaast is tijdens de evaluatie van dit onderzoek naar boven gekomen dat leerlingen niet ver zijn gekomen in de theorie. VWO is immers niet eens toegekomen aan loops. Op dit punt is de module nog zeker te verbeteren.

De verklaring hiervoor is dat we als beginnende docenten nog veel moesten verkennen en leren op het gebied van lesgeven. We wisten van te voren niet zeker welk tempo we konden hanteren en bovendien hebben we te maken gehad met veel lesuitval. Een verbeterpunt voor de volgende versie is om meer theorie te behandelen en op een hoger tempo te werken. Dit past ook bij eerdere bevindingen.

7.2 Restricties

Er zijn een aantal aspecten die de uitvoering hebben beperkt.

Toen deze module was ontworpen waren we net begonnen met de lerarenopleiding, dus onze ervaring op het gebied van lesontwerp en docentschap was beperkt. Hierdoor zijn er niet altijd optimale keuzes gemaakt. Zoals opgemerkt ontbreken bijvoorbeeld specifieke leerdoelen per les. Verder zijn veel lessen door ons voor het eerst gegeven, waardoor veel technieken voor ons experimenteel waren.

Verder was het voor ons niet mogelijk om de standaard lessenseries te presenteren aan een controlegroep. Er is dus geen materiaal om onze module en resultaten te vergelijken met een klas die de standaardmodules gebruikt. Het was niet mogelijk om de klassen op te splitsen in twee en elke groep andere theorie en opdrachten voor te leggen. Dit zou leiden tot ethische problemen en mogelijk protest van de ouders. Bovendien was er niet genoeg tijd om twee lesmodules voor te bereiden. Het was ook niet mogelijk om HAVO en VWO verschillende modules te geven, want de resultaten van de HAVO-klas zullen erg anders zijn dan van de VWO-klas.

Hoewel bij een praktijkonderzoek onderzocht wordt in hoeverre de ontwerpdoelen zijn behaald, zou een vergelijkend onderzoek kunnen bevestigen of de nieuwe module beter is dan de oude module die is vervangen.

7.3 Aanbevelingen en vervolgonderzoek

De volgende verbeterpunten kunnen in de module worden aangebracht:

- De VWO leerlingen kunnen meer uitgedaagd worden. Zoals blijkt uit de analyse van de cijfers en door de expert review kunnen de VWO leerlingen meer uitdaging gebruiken. De opdrachten kunnen bijvoorbeeld ingaan op algoritmes of geparametriseerde functies.
- De maatschappelijke relevantie van programmeren kan versterkt worden, vooral in de HAVO groep, door dit te integreren in de instructie. Initieel was er het idee om maatschappelijke

relevantie van programmeren te integreren in alle lessen zodat de leerlingen zouden begrijpen hoe programmeren terugkomt in de samenleving. Bij de HAVO-klas is hier uiteindelijk niet veel aandacht aan besteedt, wat in de toekomst wel gedaan zou kunnen worden. Verder is het een idee om de opdrachten zo te ontwerpen dat ze maatschappelijk relevant zijn. Leerlingen kunnen bijvoorbeeld gevraagd worden om programma's te maken die senioren in de ouderenzorg zouden kunnen helpen.

- Beter uitleggen hoe de leerlingen de verschillende bronnen kunnen gebruiken. Uit observatie is gebleken dat leerlingen vooral gebruik hebben gemaakt van de PowerPoint slides uit de klassikale presentaties. De informatie op deze slides is vaak beperkt en leerlingen kunnen daarom ook beter gebruik maken van de beschikbaar gestelde theorie of de website.
- De opdrachten op verschillende manieren presenteren. Uit de observatie was ook gebleken dat sommige leerlingen de opdrachten moeilijk vonden, terwijl andere leerlingen de opdrachten makkelijk vonden. Het is dan een idee om de opdrachten op verschillende manieren aan de leerlingen te presenteren. Leerlingen die een uitdaging willen krijgen minder voorgeschreven bij de uitleg. Leerlingen die het moeilijker vinden krijgen bijvoorbeeld meer tips.
- Zoals aangegeven is deze module niet uitgevoerd naast een controlegroep. In een vervolgonderzoek zou dit wel gedaan kunnen worden om zo resultaten te verkrijgen ter vergelijking.
- De module moet ook nog getoetst worden op de ontwerpeisen toepassingsgerichtheid, begeleiding bij ontwerpen en visualisatie.
- Bij opdrachten die stapsgewijs zijn beschreven is het een goed idee om reflectievragen in te bouwen zodat leerlingen actief moeten nadenken over code die ze implementeren.

7.4 Vervolg

Zoals opgemerkt is er ruimte voor verbetering en verandering in de huidige module. In een volgende iteratie en vervolgonderzoek kunnen onder andere de verbeteringen aangebracht worden die hierboven staan beschreven. Al het materiaal, theorie, lesontwerp, opdrachten, PowerPoint slides, etc. is beschikbaar voor komende onderzoekers om te gebruiken.

Vakinhoudelijk zullen de onderzoekers programmeerkennis nodig hebben. Daarna vereist het niet veel tijd en moeite om *Processing* te begrijpen en te leren. Voor studenten kan het lastig zijn om deze module exact te implementeren. Dit komt omdat er een kans is dat programmeren niet een onderwerp is dat tijdens de stageperiode behandeld wordt. Verder is het vaak afhankelijk van de vakcoach welke onderwerpen in de lessen worden behandeld.

8 Conclusies

In dit ontwerponderzoek is een lesmodule programmeren met *Processing* ontworpen voor 4 HAVO en VWO. Er is gepoogd om een lesmodule te ontwerpen waarin de leerlingen op een interactieve en plezierige wijze kennismaken met programmeren. Hiermee is geprobeerd om de volgende hoofdonderzoeksvraag te beantwoorden: Hoe kan programmeren als een lesmethode worden aangeboden, zodat het toegankelijk en maatschappelijk relevant is voor vierdejaars leerlingen op de middelbare school?

Uit het onderzoek zijn resultaten verzameld over de meningen van de leerlingen en van verschillende experts. Deze resultaten zijn geanalyseerd om te bepalen of onze lesmodule voldoet aan de vooraf gestelde ontwerpeisen en om drie deelvragen te beantwoorden. Deze vragen worden hieronder beantwoord.

- Hoe kan programmeren als een lesmethode worden aangeboden aan vierdejaars informaticaleerlingen zodat de moeilijkheidsgraad, de laagdrempeligheid en technische methode van de lesmethode juist is voor technische en non-technische leerlingen?

Om de lesmethode toegankelijk te maken voor alle leerlingen is ten eerste gedifferentieerd tussen HAVO en VWO. De module voor VWO behandelt meer theorie en gaat dieper in op het materiaal. HAVO-leerlingen hebben meer aanwijzingen en een stappenplan gekregen tijdens de opdrachten. De VWO-leerlingen daarentegen hebben tijdens de opdrachten meer vrijheid gekregen. De resultaten laten zien dat deze differentiatie heeft gewerkt, maar dat het verbeterd kan worden door de VWO-leerlingen meer uit te dagen.

Naast differentiatie is gekozen voor een programmeertaal waarin direct visueel feedback wordt gegeven. Op deze wijze krijgen de leerlingen direct te zien wat ze coderen en kunnen ze de animaties aanpassen door de code te wijzigen. Ook is gekozen om geen technische en complexe programmeerprincipes, zoals recursie en geparametriseerde functies, te behandelen. De resultaten laten zien dat leerlingen de module toegankelijk vonden.

- Hoe kan programmeren als een lesmethode worden aangeboden aan vierdejaars informaticaleerlingen zodat de maatschappelijke relevantie van programmeren duidelijk wordt en zodat programmeren bruikbaar en nuttig wordt in hun dagelijkse leven?

Maatschappelijke relevantie en bruikbaarheid is toegevoegd in de module door aan het begin van de les actualiteiten te laten zien waarin programmeren een belangrijke rol speelt. Zo zijn voorbeelden van VR applicaties en tastbare interfaces besproken, en wat de rol en effect is van dergelijke applicaties in de samenleving . Hoewel dit minder in HAVO is toegepast, zijn de resultaten in de VWO klas positief. Door bijvoorbeeld filmpjes te laten zien in de les over hoe programmeren in de maatschappij terugkomt, hebben de leerlingen een goed beeld kunnen vormen over hoe programmeren ingezet kan worden in de samenleving.

- Hoe kan programmeren als een lesmethode op een manier worden aangeboden aan vierdejaars informaticaleerlingen zodat ze het als plezierig ervaren?

Om de module plezierig te maken voor de leerlingen zijn opdrachten gekozen die goed passen bij de belevingswereld van de leerlingen. Zo hebben de leerlingen een eigen spelletje mogen maken voor de eindopdracht. Bij VWO mochten de leerlingen zelf animaties bedenken en creëren voor de tussentijdse- en huiswerkopdrachten. Uit de resultaten blijkt dat leerlingen de module over het algemeen positief hebben ervaren.

Met deze drie deelvragen is nu ook de hoofdvraag beantwoord:

- Hoe kan programmeren als een lesmethode worden aangeboden, zodat het toegankelijk, plezierig en maatschappelijk relevant is voor vierdejaars leerlingen op de middelbare school?

Met de module is aangetoond dat, door het kiezen van een geschikte programmeertaal en de opdrachten op een goede manier aan te bieden, een programmeermodule toegankelijk, maatschappelijk relevant en leuk kan zijn. De leerlingen vonden de module toegankelijk en ze vonden het

leuk om aan de opdrachten te werken. Dit is ook terug te zien in de cijfers die de leerlingen aan de module hebben gegeven.

Er zijn ook elementen die nog verbeterd kunnen worden, zo kunnen VWO leerlingen nog meer uitgedaagd worden en zou de maatschappelijke relevantie van programmeren bij HAVO nog beter uitgelegd kunnen worden door er in de instructie aandacht aan te besteden. Ook op het gebied van gepersonaliseerd leren zijn er nog verbeteringen aan te brengen. De uitleg over het gebruik van verschillende bronnen kan beter en de opdrachten kunnen op verschillende wijzen gepresenteerd worden om differentiatie te bevorderen.

Dit ontwerp kan overgedragen worden naar een volgende groep onderzoekers. In de volgende iteratie kunnen veranderingen worden aangebracht gebaseerd op de resultaten van dit onderzoek.

9 Reflectie Anirudh Ekambaranathan

9.1 Wat wilde ik bereiken?

Voor het begin van de module hadden Heleen en ik het bestaande lesmateriaal bekeken dat gebruikt werd op de Grundel en we kwamen tot de conclusie dat leerlingen gebruikt maakten van een verouderde lessenseries over programmeren in Java. We dachten meteen dat we beiden een nieuwe module konden ontwerpen waar de leerlingen meer plezier uit zouden kunnen halen. Na besloten te hebben om daadwerkelijk een module te ontwerpen, waren er ontwerpeisen bij gekomen, zoals de toegankelijkheid voor leerlingen van alle profielen.

Naast het feit dat we een leuke lessenseries wilden neerzetten, wilde ik ook leren hoe het is om een lesmodule te creëren. Het ontwerpen van studiemateriaal is een geheel nieuwe domein dat niet veel in de opleiding is behandeld. Het was daarom aantrekkelijk voor mij om deze ervaring op te doen.

Daarnaast wilde ik leren hoe leerlingen reageren op lesmateriaal, dus wat vinden ze leuk om te doen en wat vinden ze minder interessante activiteiten. Als docent kunnen wij een module erg leuk vinden, maar leerlingen kunnen het heel erg anders ervaren.

9.2 Wat gebeurde er?

Het ontwerpen van de lesmodule ging gepaard met het geven van de lessen. Als gevolg was er niet genoeg tijd om de gehele module van te voren te ontwikkelen en is het dus gaandeweg ontwikkeld. Het ontwerpen van de module heeft veel tijd en energie gekost. Omdat de module van te voren niet af was, was het niet mogelijk om de module als geheel te analyseren voordat deze in de praktijk geïmplementeerd werd.

Het is niet makkelijk om een hele module te ontwerpen. Niet alleen hadden we er niet genoeg tijd voor, maar ook onze onervarenheid speelde hierin een rol. Het was moeilijk voor ons om in te schatten wat de leerlingen nodig hadden en op welk tempo we het materiaal moesten aan bieden. Bovendien hadden we van te voren een vrij ruime planning gemaakt met veel programmeerelementen. Later bleek wel dat we hier niet allemaal aan toe zouden komen.

In het begin probeerde ik me zo veel mogelijk aan de initiële planning te houden. We hadden ook eenzelfde planning gemaakt voor VWO en HAVO. Al snel bleek dat HAVO wat meer tijd nodig had om concepten te doorgronden en dat ze niet hetzelfde tempo kunnen aanhouden als VWO. Dus we hebben de planning voor HAVO aangepast zodat ze meer tijd hadden voor belangrijke concepten zoals if-statements.

De leerlingen zelf waren vrij enthousiast over programmeren en sommige waren erg leergierig. Ze gingen meteen online op zoek naar implementaties en spelletjes die ze konden maken. Verder merkte ik ook bij de opdrachten op dat ze wat meer begeleiding nodig hadden. Ik had de opdrachten dus zo aangepast dat ze nu een stappenplan kregen welk ze konden volgen.

Na dat het onderzoek was uitgevoerd, was er in de enquête naar voren gekomen dat sommige leerlingen meer uitgedaagd wilden worden. Leerlingen hebben diverse behoeften en als je aan alle behoeften wil voldoen, dan moet het lesmateriaal op verschillende manieren aangeboden worden. Dit zou veel tijd kosten en hiervoor moet je je doelgroep ook erg goed kennen. Het is lastig voor een beginnende docent om hierop in te spelen.

Het is me ook persoonlijk opgevallen dat ik niet veel aandacht heb besteed aan de maatschappelijke relevantie van programmeren. Uiteindelijk heb ik toch meer de nadruk gelegd op het vakinhoudelijke aspect.

9.3 Bewustwording van essentiële aspecten

Het ontwerpen van lesmateriaal is geen simpele klus: het vergt veel tijd en energie. Als full-time docent zijnde is het niet makkelijk om het naast je werk te doen. Omdat we alleen maar een paar uur per week les hebben gegeven, is het ons gelukt om het in de lesvoorbereiding te verwerken.

Als je een complete module op een juiste wijze wil ontwikkelen dan moet het van te voren compleet af zijn. Op deze wijze is het mogelijk om de nodige veranderingen door te voeren voordat de lessen beginnen. Als dit niet mogelijk is dan heb je minder tijd om de module aan te passen aan de leerlingen, omdat je daadwerkelijk bezig bent om de module nog te ontwikkelen. Daarentegen geldt wel dat het lesmateriaal gaandeweg ontwikkeld kan worden gebaseerd op de uitkomsten van de voorgaande lessen, maar zoals eerder aangegeven zal de tijdsdruk hoog zijn omdat je full-time aan het lesgeven bent.

Bij het ontwerpen van de module hadden we inhoudelijk verschillende planningen gemaakt tussen HAVO en VWO. Zo lag bij VWO bijvoorbeeld de tempo iets hoger en werd er meer theorie behandeld. Deze planning is te zien in bijlage 10.1. Echter het is ook belangrijk om rekening te houden met de subtiele verschillen, zoals de tempo waarin HAVO leerlingen leren en de begeleiding die ze nodig hebben bij het maken van de opdrachten.

Van mezelf weet ik nu dat mijn interesses vooral liggen bij vakinhoudelijke elementen van lesgeven. Ik vind het leuk om me te verdiepen in het lesmateriaal en de technische aspecten, en houd me minder bezig met de socio-technische aspecten als, bijvoorbeeld maatschappelijke relevantie en consequenties.

9.4 Wat heb ik geleerd en wat ga ik in de toekomst anders doen?

De keuze om een lesmodule te ontwerpen moet ruim van te voren worden gemaakt. Als ik weer de kans krijg om een lesmodule te ontwerpen zal ik ervoor zorgen dat ik hier genoeg tijd voor heb. Ik heb ook geleerd dat het belangrijk is om input te krijgen van ervaren docenten of vakdidactici, vooral omdat ikzelf nog niet veel ervaring heb met lesgeven. Ervaren docenten zullen beter kunnen aangeven wat de behoeften zijn van de leerlingen.

Van mezelf heb ik geleerd dat mijn interesses meer liggen bij het vakinhoudelijke element van informatica. Als ik in de toekomst een les ga ontwerpen, moet ik er mee rekening houden dat ik het niet te technisch maak. Ik zal extra moeite moeten doen om onderdelen als maatschappelijke relevantie te verwerken in het lesmateriaal.

Concluderend kan ik zeggen dat dit ontwerponderzoek een leerzame ervaring is geweest. Ik heb geleerd over het ontwerpproces, maar ik heb ook mezelf beter leren kennen.

10 Reflectie Heleen van der Zaag

Terugkijkend naar de module zoals die gegeven is, voel ik mij erg trots op wat ons gelukt is in de korte tijd die we hadden. We hebben een leuke module neergezet waarbij we veel van onze doelen (deels) behaald hebben. De leerlingen hebben op een creatieve manier kennis gemaakt met programmeren en konden aan het einde van de module op een basisniveau programmeren.

Er zijn natuurlijk een aantal opmerkingen die ik graag bij ons onderzoek en mijn functioneren daarin wil plaatsen. Hieronder zal ik een aantal verbeterpunten noemen, en een aantal punten waarover ik tevreden ben.

Een van de eerste dingen die opvalt, is dat het erg duidelijk is dat Anirudh en ik nieuw waren in het onderwijswereld. Dit is terug te zien in hoe wij zijn begonnen met ontwerpen en hoe ons lesmateriaal eruit ziet. In de theorie samenvatting die ik voor de VWO groep heb geschreven staan bijvoorbeeld teksten als: "het commentaar heb ik weggelaten om het voorbeeld zo kort mogelijk te houden". Dit klinkt heel logisch als je weinig ervaring hebt, maar hierdoor geef je niet een volledig correct voorbeeld. Daarnaast kunnen leerlingen hiernaar terug grijpen, de docent doet het zo, dus wij hoeven ook geen commentaar te geven. Dit zijn, achteraf gezien, onhandige dingen om te doen. Onze onervarenheid is ook terug te zien in onze originele planning, die erg ambitieus is. Wel is het ons gelukt om te leren van de fouten die we maakten. Ook hebben wij snel gereageerd en aanpassingen gemaakt wanneer wij fouten ontdekten.

In de VWO klas heb ik erg geprobeerd om in te spelen op de individuele verschillen tussen de leerlingen. In deze klas waren er soms drie niveaus waarop de leerlingen zaten, een groepje van vier leerlingen liep constant voor op de rest van de klas. Het grootste gedeelte van de klas zat ongeveer op hetzelfde niveau, het niveau van de module, en dan waren er een aantal leerlingen die het programmeren toch wel erg lastig vonden. Vooral aan het begin van deze module hadden deze leerlingen extra begeleiding en instructie nodig. Door de opzet van de module was er veel tijd tijdens de lessen om de leerlingen een-op-een te begeleiden. Hierdoor kon ik extra instructie geven aan de leerlingen die het lastig vonden, en extra opdrachten geven aan de leerlingen die vooruit liepen. Wel was het hierin erg lastig om de tijd goed in de gaten te houden, en alle leerlingen kort te spreken in een les.

Een andere methode die ik uitgeprobeerd heb bij de VWO klas is om de leerlingen uit te dagen om zelf antwoorden te verzinnen op hun eigen vragen. Wanneer een leerling een vraag stelde, probeerde ik hier met de leerling over te sparren en de leerling aan te moedigen om zelf met hypotheses te komen. Hierin ben ik soms een beetje enthousiast geweest. Dit resulteerde in frustratie bij de leerlingen, wat ook terug te zien is in de enquête resultaten van de leerlingen. Een leerling heeft ingevuld dat het erg vervelend was dat elke vraag met een vraag beantwoord werd. Ook maakte deze methode het voor mij als docent nog lastiger om mijn tijd effectief in te delen. Doordat elke vraag leidde tot een vraaggesprek, was het erg lastig om alle vragen van alle leerlingen te behandelen in een les. Het is voor mij erg belangrijk om te leren wanneer deze methode effectief is. Sommige leerlingen en sommige vragen reageren niet goed op deze methode. Zo kan het averechts werken wanneer een korte vraag niet gewoon beantwoord wordt maar leidt tot een vraaggesprek.

Als persoon ben ik erg direct, en ik vind het ook fijn als andere mensen direct zijn tegen mij. Dit heb ik ook toegepast in mijn lessen en hoe ik met de leerlingen omging. Dit werkte erg goed

wanneer het om positieve feedback ging, maar wanneer het om minder positieve feedback ging kan ik erg makkelijk sterker overkomen dan ik het bedoel. Dit kan erg demotiverend werken voor de leerlingen. Daarom is het voor mij belangrijk om mijn feedback positief te verwoorden. Dus dat ik ook wanneer ik aan wil geven dat ik meer verwacht van een leerling, dat ik dit bijvoorbeeld zo verwoord dat ik geloof dat hij/zij beter kan en dat ik er naar uit kijk om dit te zien.

Iets wat mij tijdens deze stage en het maken van deze module duidelijk is geworden, is dat ik veel passie heb voor Informatica en het geven ervan. Dit was ook erg duidelijk te zien in hoe ik voor de klas stond. Voordat ik begon met deze module had ik er erg veel moeite mee, maar toen ik deze module mocht geven, had ik er zin in en was ik erg enthousiast. Deze passie heeft me erg geholpen tijdens de lessen in deze module. Door de energie en passie die ik uitstraalde waren de leerlingen ook positief gestemd tegenover het onderwerp en mij als docent. Dit heb ik ook als feedback van mijn begeleider gekregen.

References

Ala-Mutka, K. (2004). Problems in learning and teaching programming–a literature study for developing visualizations in the codewitz-minerva project. *Codewitz needs analysis*, 1–13.

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, *4*(3), 114–123.

Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, *24*(6), 574–594.

Barendsen, E., Grgurina, N., & Tolboom, J. (2016). A new informatics curriculum for secondary education in the netherlands. In *International conference on informatics in schools: Situation, evolution, and perspectives* (pp. 105–117).

Barendsen, E., & Tolboom, J. (2016). Advies examenprogramma informatica vwo-havo: inhoud en invoering.

Blockly. (n.d.). `https://developers.google.com/blockly/`.

Brooke, J. (2013). Sus: a retrospective. *Journal of usability studies*, *8*(2), 29–40.

Bruning, L., & Michels, B. (2013). *Concept-contextvenster: zicht op de wisselwerking tussen concepten en contexten in het bèta-onderwijs.* slo, nationaal expertisecentrum leerplanontwikkeling.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. In *Acm sigcse bulletin* (Vol. 33, pp. 49–52).

Culwin, F. (1997). Objects first, objects last or objects at all. In *5th annual conference on the teaching of computing* (pp. 56–59).

Davies, S. P. (1993). Models and theories of programming strategy. *International Journal of Man-Machine Studies*, *39*(2), 237–267.

Decker, A. (2007). *How students measure up: An assessment instrument for introductory computer science* (Unpublished doctoral dissertation). State University of New York at Buffalo.

Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, *2*(1), 57–73.

Elenbogen, B. S., Maxim, B. R., & McDonald, C. (2000). Yet, more web exercises for learning c++. *ACM SIGCSE Bulletin*, *32*(1), 290–294.

Frey, C. B., & Osborne, M. A. (2017). The future of employment: how susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, *114*, 254–280.

Gander, W., Petit, A., Berry, G., Demo, B., Vahrenhold, J., McGettrick, A., ... others (2013). Informatics education: Europe cannot afford to miss the boat. *ACM,[online] Available at: http://europe. acm. org/iereport/ie. html*.

Gomes, A., Carmo, L., Bigotte, E., & Mendes, A. (2006). Mathematics and programming problem solving. In *3rd e-learning conference–computer science education* (pp. 1–5).

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. In *International conference on engineering education–icee* (Vol. 2007).

Good, J., Brna, P., & Cox, R. (1997). Novices and program comprehension: Does language make a difference?". In *Proceedings of 19th annual conference of the cognitive science society* (pp. 936–937).

Guindon, R. (1990). Knowledge exploited by experts during software system design. *International Journal of Man-Machine Studies*, *33*(3), 279–304.

Huizinga, J., & Hull, R. F. C. (1949). *Homo ludens. a study of the play-element in culture.[translated by rfc hull.].* Routledge & Kegan Paul.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, *13*(3), 259–290.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd annual conference of the ltsn centre for information and computer sciences* (Vol. 4, pp. 53–58).

Kaczmarczyk, L., & Dopplick, R. (2014). Acm report: preparing students for computing workforce needs in the us. *ACM SIGCSE Bulletin*, *46*(2), 8–8.

Kölling, M., & Rosenberg, J. (1996). Blue—a language for teaching object-oriented programming. In *Acm sigcse bulletin* (Vol. 28, pp. 190–194).

Kulik, J. A. (2001). Student ratings: Validity, utility, and controversy. *New directions for institutional research*, *2001*(109), 9–25.

Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., Clair, C. S., & Thomas, L. (2006). A cognitive approach to identifying measurable milestones for programming skill acquisition. *ACM SIGCSE Bulletin*, *38*(4), 182–194.

Michels, B. (2006). Verschil moet er wezen. *Een werkdocument over verschillen tussen havo-en vwo-leerlingen in de*.

Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, *7*(1), 55–66.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., ... others (2002). Exploring the role of visualization and engagement in computer science education. In *Acm sigcse bulletin* (Vol. 35, pp. 131–152).

Ng, E., & Bereiter, C. (1991). Three levels of goal orientation in learning. *Journal of the Learning Sciences*, *1*(3-4), 243–271.

Orpen, C. (1990). The measurement of student satisfaction: A consumer behavior perspective. *Journal of Human Behavior and Learning*, *7*(1), 34–7.

Pane, J., & Myers, B. (1996). Usability issues in the design of novice programming systems.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. *Journal of Educational Computing Research*, *2*(1), 37–55.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, *13*(2), 137–172.

Robinson, C. F., & Kakela, P. J. (2006). Creating a space to learn: A classroom of fun, interaction, and trust. *College Teaching*, *54*(1), 202–207.

Rowe, G., & Thorburn, G. (2000). Vince—an on-line tutorial tool for teaching introductory

programming. *British Journal of Educational Technology*, *31*(4), 359–369.
Sajaniemi, J. (2002). *Planani-a system for visualizing roles of variables to novice programmes*. University of Joensuu, Department of Computer Science.
Schmidt, V. (2008). Vakdossier 2007 informatica. *Enschede, SLO*.
Scratch. (n.d.). `https://scratch.mit.edu/`.
Shiffman, D. (2009). *Learning processing: a beginner's guide to programming images, animation, and interaction*. Morgan Kaufmann.
Soloway, E. (2013). *Studying the novice programmer*. Psychology Press.
Tew, A. E., & Guzdial, M. (2010). Developing a validated assessment of fundamental cs1 concepts. In *Proceedings of the 41st acm technical symposium on computer science education* (pp. 97–101).
Tews, M. J., Jackson, K., Ramsay, C., & Michel, J. W. (2015). Fun in the college classroom: Examining its nature and relationship with student engagement. *College Teaching*, *63*(1), 16–26.
Tolboom, J., Krüger, J., & Grgurina, N. (2014). Informatica in de bovenbouw havo/vwo. *Naar aantrekkelijk en actueel onderwijs in informatica*.
Van Weert, T. (1998). Informatics for secondary education—the unesco/ifip curriculum as a resource for developed and developing countries. In *Capacity building for it in education in developing countries* (pp. 275–288). Springer.
van Wetenschappen, K. N. A. (2013). *Digitale geletterdheid in het voortgezet onderwijs: vaardigheden en attitudes voor de 21ste eeuw*. Amsterdam: Koninklijke Nederlandse Akademie van Wetenschappen.
Whalley, J. L., & Lister, R. (2009). The bracelet 2009.1 (wellington) specification. In *Proceedings of the eleventh australasian conference on computing education-volume 95* (pp. 9–18).
Wiedenbeck, S., & Ramalingam, V. (1999). Novice comprehension of small programs written in the procedural and object-oriented styles. *International Journal of Human-Computer Studies*, *51*(1), 71–87.
Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. L. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, *11*(3), 255–282.
Wiers-Jenssen, J., Stensaker, B., & Grogaard, J. B. (2002). Student satisfaction: Towards an empirical deconstruction of the concept. *Quality in higher education*, *8*(2), 183–195.
Willis, J. (2006). *based strategies to ignite student learning: Insights from a neurologist and classroom teacher*. ASCD.
Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM Sigcse Bulletin*, *28*(3), 17–22.

11 Bijlagen

11.1 Lesmodule: Programmeren met Processing

Lesmodule: Visueel Programmeren in Processing

Anirudh Ekambaranathan
Heleen van der Zaag

Begeleiders:
Ingrid Breymann, Marieke Huisman, Jan van der Veen

Startdatum: 10-10-2016
Beoogde einddatum: 01-03-2017

October 2, 2017

1 Inleiding

In het onderstaande document wordt een lesmodule uiteengezet die ontwikkeld is door docenten in opleiding van de Universiteit Twente. Deze lesmodule is ontworpen in de hoop de leerlingen een laagdrempelige en leuke kennismaking te geven met programmeren. Daarbij hebben wij gelet op de maatschappelijke context van programmeren, de moeilijkheidsgraad, laagdrempeligheid en het technisch niveau van de module, en de effectiviteit van hetgeen wat behandeld wordt. Daarnaast willen we ook graag dat de leerlingen plezier hebben in het leren van programmeren en enthousiast zijn om er na deze module mee verder te gaan.

Eerst zal onze originele planning gepresenteerd worden, deze is al heel snel een stuk ingekort door vele vrije dagen en uitgevallen lessen. De aangepaste (en dus gegeven) planning wordt hierna gepresenteerd. Bij deze planning worden alle opdrachten gegeven die de leerlingen wekelijks kregen. De eindopdracht duurde twee weken.

2 Originele planning

Een module op Lyceum De Grundel, waar deze module is gegeven, heeft een tijdsperiode van acht weken. De lesmethode zal daarom ook worden opgesplitst in acht onderdelen. Met dat in gedachte is de volgende tijdsplanning bedacht welk te zien is in Tabel 1. Tijdens de ontwikkeling van deze lesmodule is losjes gebruik gemaakt van het boek Learning Processing van Daniel Schiffman, dit boek geeft een toegankelijke handleiding voor de programmeertaal Processing. Het boek is leuk geschreven en werkt heel visueel. Hier is gebruik van gemaakt tijdens het ontwikkelen van de module. Delen van het boek zijn vertaald en omgeschreven om de verschillende lessen te vormen. Ook is er soms gebruik gemaakt van opdrachten die in het boek staan.

3 Aangepaste planning

Door verschillende redenen buiten de invloed van de docenten in opleiding zijn er lessen uitgevallen waardoor er een verschuiving ontstond in de lessen en de behandelde onderwerpen. Hierdoor is er een aangepaste planning gemaakt waarin de twee 'gevorderde' onderwerpen, functies en loops, zijn komen te vervallen. Hier is voor gekozen omdat dit de leerlingen de ruimte gaf om de basis begrippen goed te oefenen. We wilden voorkomen dat de leerlingen veel theoretisch hadden behandeld maar dit niet konden implementeren. Bij de VWO groep is het onderwerp functies wel optioneel aangeboden. De aangepaste planning staat hieronder weergegeven in Tabel 2. De opdrachten die in het volgende hoofdstuk zijn gepresenteerd zijn gebaseerd op deze tijdsplanning.

Table 1: Lesplanning per week

Week	**Doel**	**Onderwerpen**
1	Leerlingen kunnen met pixels en processing een poppetje programmeren.	Hoofdstukken 1 & 2: Pixels en Processing
2	Leerlingen kunnen interactie programmeren en hun poppetje laten reageren op user input.	Hoofdstuk 3: Interactie
3	Leerlingen kunnen variabelen en condities toepassen op interactie met het poppetje.	Hoofdstukken 4 & 5: Variabelen en Condities
4	Leerlingen passen hun kennis toe in een tussentijdse opdracht.	Tussentijdse opdracht.
5	Leerlingen kunnen loops gebruiken om het programmeren van graphics te versnellen en verbeteren.	Hoofdstuk 6: Loops
6	Leerlingen kunnen op een modulaire wijze programmeren met behulp van functies.	Hoofdstuk 7: Functies
7	Leerlingen passen hun kennis toe in een eindopdracht.	Eindopdracht
8	Leerlingen passen hun kennis toe in een eindopdracht.	Eindopdracht

4 Theorie

Elke les (behalve tijdens de tussentijdse opdracht en de eindopdracht) is er aandacht besteed aan een bepaald stuk theorie. Dit verschilde van functie gebruik tot variabelen. Deze theorie is ook altijd aan de leerlingen aangeboden in de vorm van een samenvattend document. Dit document is elke week uitgebreid met de theorie van die week. In bijlage 5.2.6 en 5.2.6 is het einddocument voor beide klassen toegevoegd. Hierin staat een samenvatting van alle theorie die behandeld is in de lessen.

5 Opdrachten

De planning is zo ontworpen dat elke week ongeveer één onderwerp behandeld wordt. Tijdens de lessen passen de leerlingen de concepten toe in een opdracht. Er zijn wekelijkse opdrachten waarin de leerlingen individuele concepten toepassen, en er is een tussentijdse en eindopdracht waarin de leerlingen al hun kennis toepassen om een grotere programma te maken. Voor de tussentijdse opdracht en de eindopdracht hebben de leerlingen meer tijd. De leerlingen in de havo groep kregen andere opdrachten, deze opdrachten waren meer gestruc-

Table 2: Lesplanning per week (zoals uitgevoerd)

Week	**Doel**	**Onderwerpen**
1	Leerlingen kunnen met pixels en processing een poppetje programmeren.	Hoofdstukken 1 & 2: Pixels en Processing
2	Leerlingen kunnen interactie programmeren en hun poppetje laten reageren op user input.	Hoofdstuk 3: Interactie
3 & 4	Leerlingen kunnen variabelen toepassen om hun poppetje te laten bewegen.	Hoofdstuk 4: Variabelen
4 & 5	Leerligen kunnen condities gebruiken om interactie te creëren.	Hoofdstuk 5: Condities.
6	Leerlingen passen wat ze geleerd hebben toe in een tussentijdse opdracht. *vwo leerlingen kregen, optioneel, extra uitleg over zelfgeschreven functies	Tussentijdse opdracht
7	Leerlingen passen hun kennis toe in een eindopdracht.	Eindopdracht
8	Leerlingen passen hun kennis toe in een eindopdracht.	Eindopdracht

tureerd. Hieronder staan de beschrijvingen van de opdrachten voor de vwo en de havo groepen, zoals ze aan de leerlingen werden voorgelegd.

5.1 VWO 4

5.1.1 Week 1

Maak een ontwerp met Processing, gebruik hiervoor simpele 2D vormen - arc(), ellipse(), line(), point(), quad(), rect(), triangle() - en de basis kleurenfuncties - background(), colorMode(), fill(), noFill(), noStroke() en stroke(). Denk eraan dat je size() gebruikt om je schermpje een bepaalde grote te geven, of fullScreen() gebruikt om je sketch de grote van je scherm te geven. Speel de sketch na elke nieuwe regel code af, zo kan je zien of en waar er fouten zijn. Gebruik // tekst of /* tekst */ om je code te becommentariëren.

5.1.2 Week 2

Maak je ontwerp van week 1 interactief door de tekening te beïnvloeden via je muis. Dit zou kunnen zijn doordat bepaalde vormen de muis volgen, de grote veranderd op basis van de muissnelheid, kleuren veranderen op basis van de muis locatie, etc. Gebruik // tekst of /* tekst */ om je code te becommentariëren,

en vergeet niet je programma van week 1 te verbeteren zodat het netjes in de setup() en draw() staat.

5.1.3 Week 3

Gebruik je ontwerp van de afgelopen twee weken en voeg de volgende functionaliteit toe: Zorg dat delen van je ontwerp van kleur veranderen als de muis eroverheen gaat. Beweeg je ontwerp over het scherm. Lukt het je om je ontwerp van alle kanten van het scherm te laten stuiteren? (optioneel) Laat kleuren in en uit vagen. Gebruik // tekst of /* tekst */ om je code te becommentariëren.

5.1.4 Tussentijdse opdracht

Tussentijdse opdracht

1. Maak een nieuw ontwerp/figuur voor deze opdracht, hiervoor kan je inspiratie opdoen uit week 1 van dit kwartiel, gebruik wel variabelen in plaats van cijfers in je functies. Je ontwerp/figuur moet uit minimaal 4 vormen bestaan.

2. Schrijf code die ervoor zorgt dat jouw variabelen veranderen om je ontwerp dynamisch en interactief te maken. Je mag ook systeemvariabelen gebruiken zoals width, height, mouseX en mouseY.

3. Gebruik voorwaardelijke statements om het gedrag van jouw ontwerp te beïnvloeden gebaseerd op bepaalde voorwaarden. Wat gebeurt er als jouw ontwerp de zijkant van het scherm raakt? Of als het groter wordt dan een bepaalde grote? Wat gebeurd er als je muis over bepaalde onderdelen van jouw ontwerp gaat?

Gebruik // tekst of /* tekst */ om je code te becommentariëren. Je wordt niet alleen beoordeeld op je technisch werk, maar ook op je creatieve invulling van de opdracht.

5.1.5 Eindopdracht

Creatief
Schrijf samen met je groepsgenoot een spel. Met de functies die je afgelopen weken hebt geleerd en geoefend moet het je prima lukken om de volgende spellen te kunnen maken. Kies een van deze spellen en maak er je eigen versie van. Je kan er ook voor kiezen om een ander spel te maken, maar overleg dit dan eerst met je docent.

- Bubble pop
- Space invaders
- Een simpele shooter

Technisch
Gebruik van:

- Verschillende vormen
- Variabelen
- Beweging
- Interactie
- Voorwaardelijke statements (if statements)
- Overzichtelijke structuur
- Commentaar
- (optioneel) zelfgeschreven functies

Gebruik // tekst of /* tekst */ om je code te becommentariëren. Je wordt niet alleen beoordeeld op je technisch werk, maar ook op je creatieve invulling van de opdracht.

5.2 HAVO 4

5.2.1 Opdracht 1

Onder de beschrijving staat uitleg over de verschillende instructies die je kan gebruiken tijdens de opdracht (pagina 3). Jullie zullen tijdens deze opdracht een alien figuur natekenen.

Opzetten van het scherm

1. Open de Processing compiler.
2. Lees de uitleg over de verschillende instructies/functies goed door. Pagina 3.
3. Creëer een scherm van 480 bij 270 pixels.
4. Zet de achtergrondkleur van het scherm naar wit. Wit heeft een kleurcode van 255.
5. Zet ellipse modus en rect modus naar CENTER. Zie de uitleg op pagina 3.
6. Leg uit, door commentaar te geven in je code, wat je net hebt gedaan.

Lichaam van de alien.
Je gaat zo het lichaam van de alien tekenen. Dit is een rechthoek met een grijze kleur.

1. Je ziet dat de lijnen en omtrek van de alien zwart is. Zorg ervoor dat de lijnen die je tekent zwart zijn. (Je wilt een zwarte 'pen' selecteren). Kleurcode van zwart is 0.

2. Het lichaam is een rechthoek met een grijze kleur/invulling. Zorg ervoor dat het lichaam grijs gekleurd gaat worden. Het gaat hier om de invulling. De kleur grijs heeft een kleurcode van 150.

3. Je hebt nu een zwarte pen geselecteerd en aangegeven dat je je volgende vorm een grijze kleur wilt geven. Teken nu een rechthoek met de afmetingen 20px breed en 100px hoog. Teken dit ongeveer in het midden van je scherm.

4. Leg uit, door commentaar te geven in je code, wat je net hebt gedaan.

Hoofd van de alien.

1. Je hebt al een zwarte 'pen' geselecteerd in de vorige opgave. Het hoofd heeft een witte invulling. Zorg dus dat je volgende vorm een witte kleur gaat krijgen. Het gaat hier om de invulling en niet om de omtrek. Wit heeft een kleurcode van 255.

2. Teken nu het hoofd van de alien bovenop het lichaam. Het hoofd is 60px breed en 60px hoog. (Tip: het hoofd lijkt het vorm te hebben van een ellips).

3. Leg uit, door commentaar te geven in je code, wat je net hebt gedaan.

Ogen van de alien.

1. De ogen van de alien zijn zwart. Geef aan dat je volgende vorm een zwarte kleur moet hebben. Het gaat hier om de invulling.

2. Teken twee ogen in het hoofd van de alien. Elk oog heeft de afmetingen 16px breed en 32px hoog.

3. Leg uit, door commentaar te geven in je code, wat je net hebt gedaan.

Benen van de alien.

1. De benen van de alien zijn zwart, dus geef aan dat je de volgende lijnen zwart wil tekenen, net zoals in stap 6.

2. Teken de twee benen van de alien. Je kan hiervoor twee lijnen tekenen. Kijk goed naar figuur 1 zodat je weet hoe de benen geplaatst moeten worden.

3. Leg uit, door commentaar te geven in je code, wat je net hebt gedaan.

5.2.2 Opdracht 2

Deze opdracht bestaat uit twee onderdelen. Opdracht A is het tekenen van een vierkant en er voor zorgen dat het vierkant je muis volgt. Met andere woorden, het vierkant gaat waar je je muis heen beweegt. Vervolgens plaatsen we een cirkel op het vierkant, en zorgen we ervoor dat de vierkant en de cirkel als één geheel bewegen.

Opdracht B is het laten verschijnen van vierkanten op het moment dat je met je muis klikt. Het vierkant verschijnt op de plek waar je hebt geklikt. Op het moment dat je een in toets indrukt van het toetsenbord, dan verdwijnen alle vierkanten. In dit document staat stap voor stap uitgelegd hoe je dit gaat doen.

Opdracht A
Een nieuwe programma maken

1. Open de Processing compiler.
2. Zorg ervoor dat je werkt in een nieuwe bestand.

Structureren van code in blokken

1. Je nieuwe programma zal netjes opgesplitst zijn in duidelijke blokken van code. Begin met het maken van een blok code dat je programma opstart.
2. Stop de code voor het opstarten van je scherm hierin.
3. Creëer een blok voor het tekenen van figuren.
4. Zet in dit blok de code voor het tekenen van een vierkant. Zorg er zelf voor dat het een juiste afmeting heeft met een leuke kleur.
5. Je hebt als het goed is ook de achtergrondkleur ingesteld. Heb je dat nog niet gedaan, doe dat dan nu. Denk goed na waar je dit plaatst.

Muis interactie toevoegen

1. Nu gaan we ervoor zorgen dat je vierkant je muis gaat volgen. De eerste twee argumenten voor het tekenen van een vierkant zijn de coördinaten. Vervang deze coördinaten met de coördinaten van de muis.
2. Het kan zijn dat de linker bovenhoek van de vierkant je muis achtervolgt. Dit is natuurlijk niet zo elegant. Zorg ervoor dat het middelpunt van de vierkant op de plek van je muis is. Dit heb je in de vorige les geleerd.

Een cirkel toevoegen

1. Teken een cirkel boven het vierkant.

2. Zorg ervoor dat je cirkel boven het vierkant blijft; ook bij het bewegen van de muis. Je kan als argumenten ook rekenkundige bewerkingen meesturen. Bijvoorbeeld: $rect(10 + 10, 5 + 5, 10, 10)$; Dit tekent een rechthoek met de coördinaten (20, 10). Je kan bijvoorbeeld ook: rect(mouseX + 5, mouseY + 10, 10 ,10); Dit tekent een rechthoek met de coordinaten van waar de muis is, plus een getal: $(x - coordinaatvanmuis + 5, y - coordinaatvanmuis + 10)$.

3. Voeg commentaar toe aan al je code om toe te lichten wat je hebt gedaan.

4. Sla je bestand op en lever het in voor vrijdag 25 november om 23.59 uur.

Opdracht B

Een nieuwe programma maken

1. Open de Processing compiler.

2. Zorg ervoor dat je werkt in een nieuwe bestand.

Structureren van code in blokken

1. Je nieuwe programma zal netjes opgesplitst zijn in duidelijke blokken van code. Begin met het maken van een blok code dat je programma opstart. Gebruik hiervoor de constructie

```
void setup(){
        //Hier komt je code
 }
```

2. Stop de code voor het opstarten van je scherm hierin.

3. Creëer een blok voor het tekenen van figuren. Gebruik hiervoor de constructie:

```
 void draw(){
        //Hier komt je code
}
```

   Denk goed na waar je de code stopt om de achtergrond van je scherm te bepalen.

4. Creëer een blok waarmee je code kan toevoegen op het moment dat er een muisklik is geweest. Gebruik hiervoor de constructie:

```
 void mousePressed(){
        //Hier komt je code
}
```

5. Creëer een blok waarmee je code kan toevoegen op het moment dat er een toets van het toetsenbord is ingedrukt. Gebruik hiervoor de constructie:

```
void keyPressed(){
        //Hier komt je code
}
```

Interactie met de muis toevoegen

1. Voeg de code toe zodat er een vierkant wordt getekend op de plek van de muis wanneer er geklikt wordt.

Interactie met het toetsenbord toevoegen

1. Voeg de code toe zodat alle vierkanten worden weggehaald op het moment dat een toets van het toetsenbord is ingedrukt. Je kan dit doen door het achtergrond weer opnieuw wit te maken. Er wordt dan als het ware een nieuwe witte laag overeen gezet.

2. Voeg commentaar toe aan al je code om toe te lichten wat je hebt gedaan.

3. Sla je bestand op en lever het in voor vrijdag 25 november om 23.59 uur.

5.2.3 Opdracht 3

Opdracht A:
Gebruik wat jullie nu hebben geleerd om een eigen korte interactieve animatie te maken. Je kan gebruik maken van verschillende vormen en je kan de bewegingen van de muis gebruiken. Je mag gebruik maken van de alien van de eerste opdracht, maar je mag ook zelf iets nieuws maken; bijvoorbeeld een dier, een gebouw, een bepaalde object. Zorg ervoor dat wat je tekent bestaat uit meerdere figuren. Vergeet niet commentaar toe te voegen.

Opdracht B:
Licht toe, onder je code, hoe je dit zou willen uitbreiden en wat je nog zou willen leren. Bijvoorbeeld, je zou willen leren hoe je je tekening met de pijltjestoetsen kan besturen.

5.2.4 Opdracht 4

Maak een nieuwe figuur, of gebruik een figuur die jullie eerder in de vorige opdrachten hebben gemaakt, en zorg ervoor dat je de figuur kan besturen met de pijltjestoetsen. Op pagina 3 staat uitgelegd hoe je dit zou kunnen doen. Vergeet niet commentaar toe te voegen.

Optioneel
Ben je klaar, en wil je je programma uitbreiden, zorg er dan voor dat je figuur binnen het scherm blijft. Dus zorg ervoor dat je figuur niet buiten je scherm

kan komen.

Optioneel
Als je nog meer uitgedaagd wil worden, voeg dan nog een figuur toe in je tekening. Zorg ervoor dat je figuur hier niet 'op kan komen'. Dus je kan hier tegen aan botsen, maar je kan er niet doorheen bewegen.

5.2.5 Tussentijdse opdracht

1. Maak een nieuw ontwerp/figuur voor deze opdracht, hiervoor kan je inspiratie opdoen uit week 1 van dit kwartiel. Maak gebruik van variabelen in je ontwerp. Je ontwerp/figuur moet uit minimaal 4 vormen bestaan.

2. Zorg ervoor dat je ontwerp dynamisch en interactief wordt. Maak hiervoor gebruik van je variabelen. Je mag ook systeemvariabelen gebruiken zoals width, height, mouseX en mouseY.

3. Gebruik voorwaardelijke statements (if statements) om het gedrag van jouw ontwerp te beïnvloeden gebaseerd op bepaalde voorwaarden. Wat gebeurt er als jouw ontwerp de zijkant van het scherm raakt? Wat gebeurt er als je muis over bepaalde onderdelen van jouw ontwerp gaat?

   Vergeet niet om je code te becommentariëren.

   Je wordt niet alleen beoordeeld op je technisch werk, maar ook op je creatieve invulling van de opdracht.

5.2.6 Eindopdracht

Creatief
Schrijf samen met je groepsgenoot een spel. Met de functies die je afgelopen weken hebt geleerd en geoefend moet het je prima lukken om de volgende spellen te kunnen maken. Kies een van deze spellen en maak er je eigen versie van. Je kan er ook voor kiezen om een ander spel te maken, maar overleg dit dan eerst met je docent.

- Bubble pop
- Space invaders
- Een simpele shooter

Technisch
Gebruik van:

- Verschillende vormen
- Variabelen
- Beweging

- Interactie
- Voorwaardelijke statements (if statements)
- Overzichtelijke structuur
- Commentaar
- (optioneel) zelfgeschreven functies

Gebruik // tekst of /* tekst */ om je code te becommentariëren. Je wordt niet alleen beoordeeld op je technisch werk, maar ook op je creatieve invulling van de opdracht.

6 Bijlagen

6.1 Theorie samenvatting VWO

Uitleg benodigde functies en instructies

Voor meer uitleg kan je altijd naar https://processing.org/reference/

Week 1: Vormen

`ellipse(x, y, breedte, hoogte)`
Hiermee teken je een ellips. Je moet 4 argumenten meegeven. Een x-coördinaat, een y-coördinaat, de breedte van de ellips en de hoogte van de ellips. De coördinaten die je moet meegeven zijn van de linkerbovenhoek van je ellips. Soms kan dit onhandig zijn, dan kun je de instructie gebruiken zoals die hieronder staat beschreven.

`rect(x, y, breedte, hoogte)`
Hiermee teken je een rechthoek. Je moet 4 argumenten meegeven. Een x-coördinaat, een y-coördinaat, de breedte van de rechthoek en de hoogte van de rechthoek.

`rectMode(CENTER);`
De coördinaten die je moet meegeven met je rechthoek zijn van de linkerbovenhoek. Het is soms makkelijker als de coördinaten het middelpunt zijn van je rechthoek. Met de instructie/functie `rectMode(CENTER);` heb je dat ingesteld. Je hoeft hier verder niks aan te veranderen.

`rectMode(CORNER);`
Als je voorheen de functie `rectMode(CENTER)` hebt gebruikt, dan kun je deze functie gebruiken om het weer terug te zetten naar hoe het eerst was. De coördinaten die je dan meegeeft met je rechthoek, zijn de coördinaten van de linker bovenhoek van je ellips. Je hebt deze functie niet nodig bij je opdracht.

`line(x1,y1,x2,y2);`
Je gebruikt dit om een lijn te tekenen. Het heeft 4 argumenten nodig. Dit is iets anders dan de ellips en rechthoek. De eerste twee argumenten vormen de coördinaten van het beginpunt van je lijn en de laatste twee argumenten vormen de coördinaten van het eindpunt van je lijn.

`//`
Met `//` kun je commentaar geven. Alles wat na de `//` komt wordt genegeerd door de computer. Dit is handig om voor jezelf wat uitleg te geven. Zo snap je later nog wat je hebt gedaan.

Week 2: Algemene functies

`size(breedte, hoogte);`
Dit stelt de grootte van je scherm in. Het eerste argument, `breedte`, geeft aan hoeveel pixels breed je scherm moet zijn, en het tweede argument, `hoogte`, geeft aan hoeveel pixels lang/hoog je scherm moet zijn.

`background(kleur);`
Hiermee stel je een achtergrondkleur in voor je scherm. `kleur` neemt de waarde van een getal. Als het bijvoorbeeld 0 is, dan krijg je een zwarte achtergrond. Is het 255, dan krijg je een witte achtergrond.

`stroke(kleur);`
Hiermee geef je aan met welke kleur je lijnen wilt tekenen. Teken je bijvoorbeeld een rechthoek, waarbij de omtrek zwart is, dan stel je de waarde in op 0: `stroke(0)`.

`fill(kleur);`
Dit gebruik om aan te geven welke kleur de invulling van je volgende vorm moet hebben. Als je een rechthoek tekent nadat je `fill(0)` hebt gedaan, dan krijgt je rechthoek een zwarte kleur.

```
void setup(){
//Hier komt je code
}
```
Dit is een functie welk de compiler als eerste uitvoert. Deze functie wordt maar 1 keer uitgevoerd. Je kunt hierin alle code zetten waarmee je je programma opstart. Bijvoorbeeld, grootte van het scherm.

```
void draw(){
//Hier komt je code
}
```
Dit is een functie welk de compiler herhaaldelijk uitvoert. Op het moment dat een compiler aan het eind van deze functie is gekomen, start het weer van begin af aan. Je kan hierin al je tekeningen zetten.

```
void mousePressed(){
//Hier komt je code
}
```
De code binnen dit blok wordt uitgevoerd op het moment dat de muis wordt ingedrukt. De code zal éénmaal worden uitgevoerd.

```
void keyPressed(){
//Hier komt je code
}
```
De code binnen dit blok wordt uitgevoerd op het moment dat een toets van het toetsenbord wordt ingedrukt. De code zal éénmaal worden uitgevoerd.

Week 3 & 4: Variabelen

Variabelen kan je gebruiken om informatie in op te slaan, deze informatie kan je tijdens het runnen van je script weer veranderen, mocht dat nodig zijn. Elke variabele bestaat uit 3 elementen, het soort, de naam en de waarde. Hieronder staat een voorbeeld:
int aantalPaginas = 4;
Soorten variabelen geven aan wat voor informatie er in de variabele staat, dit heeft de computer nodig om met deze informatie te werken. Veel voorkomende soorten staan hieronder beschreven.

Code	Naam	Uitleg	Voorbeeld
int	Integer	Een 'heel' getal	1 of 2 of 3124
float	Float	Een decimaal getal	3.1415
boolean	Boolean	Een waarde die waar of niet waar kan zijn	mousePressed, of het regent of niet
char	Character	Een letter	"a", "y"
string	String	Een letter reeks	"boe", "een aantal woorden"

De naam van een variabele moet uniek zijn, je kan dus niet een naam meerdere keren gebruiken. Je begint een variabele altijd met een kleine letter (woorden die beginnen met een hoofdletter worden voor classes gebruikt). Als je variabele naam langer is dan een woord, wordt het tweedeWoord er met een hoofdletter achter gezet. Dit vergtWatWennen, maarJeWentErSnelGenoegAan.

De variabele krijgt een waarde doordat je die eraan toewijst, dit doe je met een = teken. Je kan hier ook rekenkundige berekeningen mee doen. Voorbeelden:
a = 5;
b = 6 + 4;
c = a + b;
d = b / a;

Je kan ook een aantal shortcuts gebruiken.
a++ is een shortcut voor a = a + 1
b-- is een shortcut voor b = b – 1

Er zijn ook een aantal variabelen die vast zijn, deze zitten in Processing. Deze namen kun je dus ook niet zelf aan een eigen variabele geven.

`mouseX`
De is de x-coördinaat van de muis. Dit is gewoon een getal. Je kan dit als argument gebruiken in plaats van een ander getal. Bijvoorbeeld: `rect(mouseX,2,5,5);`

`mouseY`
De is de x-coördinaat van de muis. Dit is gewoon een getal. Je kan dit als argument gebruiken in plaats van een ander getal. Bijvoorbeeld: `rect(2,mouseY,5,5);`

`width`
De breedte van het schermpje (in pixels) van de sketch.

`height`
De hoogte van het schermpje (in pixels) van de sketch.

`frameRate`
De frequentie waarmee de `draw()` functie wordt uitgevoerd.

`displayWidth`
De breedte van het scherm (in pixels) van de computer.

`displayHeight`
De hoogte van het scherm (in pixels) van de computer.

`Key`

De laatst ingedrukte toets van het toetsenbord.

`keycode`
Numerieke code voor een toets die ingedrukt wordt op het toetsenbord.

`keyPressed`
Boolean, is een toets ingedrukt?

`mousePressed`
Boolean, is de muis ingedrukt?

`mouseButton`
Welke muis toets is ingedrukt? Rechts, links of midden?

Week 4 & 5: Voorwaardelijke Statements

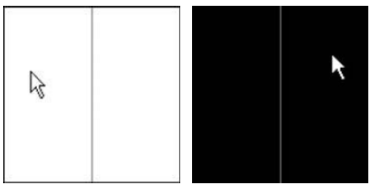
```
if (voorwaardelijke statement) {
   // code voor actie
} else {
   // code voor actie
}

while (voorwaardelijke statement) {
   //code voor actie
}
```

Met voorwaardelijke statements kan je iets laten gebeuren op een bepaald moment. Hiervoor moet iets waar of niet waar zijn. Dit kan je bereiken door een vergelijking te gebruiken. Hiervoor kan je de volgende operators gebruiken

Code	Uitleg	Voorbeeld
>	Groter dan	`if (temperature > 20) { //actie }`
<	Kleiner dan	`if (temperature < 20) { //actie }`
>=	Groter dan of gelijk aan	`if (temperature >= 20) { //actie }`
<=	Kleiner dan of gelijk aan	`if (temperature <= 20) { //actie }`
==	Is gelijk aan	`if (temperature == 20) { //actie }`
!=	Niet gelijk aan	`if (temperature != 20) { //actie }`

If the mouse is on the left side of the screen, draw a white background, otherwise draw a black background.

```
if (mouseX < width/2) {
  background(255);
} else {
  background(0);
}
```

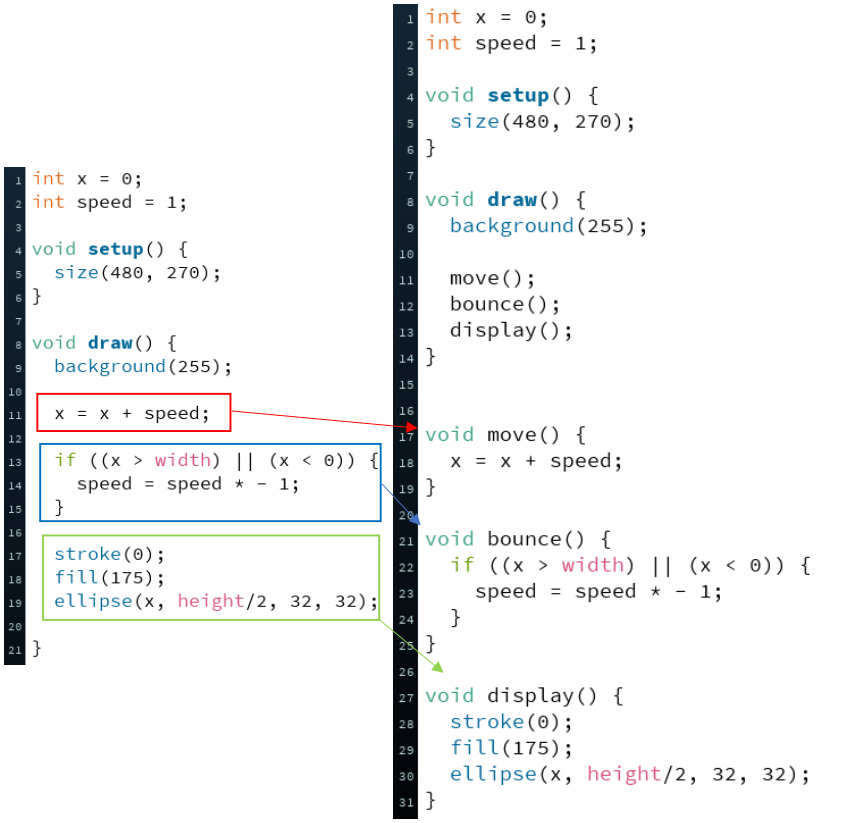


Je kan ook meerdere dingen in je voorwaardelijke statement zetten. Hiervoor gebruik je && voor en en || voor of. Dus als je iets wil doen wanneer de temperatuur tussen 18 en 20 graden zit kan je bijvoorbeeld dit gebruiken:

```
if ( temperature > 18 && temperature < 20) {
   //actie
}
```

Optioneel: Functies

Functies kunnen gebruikt worden om code te structureren en simpeler te maken. Door je code in kleinere brokken op te splitsen is het beter te achterhalen waar fouten zouden kunnen zitten. Ook is het makkelijker om extra functies toe te voegen. Zo is het ook mogelijk om de draw functie overzichtelijk en opgeruimd te houden. Hieronder zie je een klein voorbeeld van hoe functies gebruikt kunnen worden. (het commentaar heb ik weggelaten om het voorbeeld zo kort mogelijk te houden)



```
int x = 0;
int speed = 1;

void setup() {
  size(480, 270);
}

void draw() {
  background(255);

  x = x + speed;

  if ((x > width) || (x < 0)) {
    speed = speed * - 1;
  }

  stroke(0);
  fill(175);
  ellipse(x, height/2, 32, 32);

}
```

```
int x = 0;
int speed = 1;

void setup() {
  size(480, 270);
}

void draw() {
  background(255);

  move();
  bounce();
  display();
}


void move() {
  x = x + speed;
}

void bounce() {
  if ((x > width) || (x < 0)) {
    speed = speed * - 1;
  }
}

void display() {
  stroke(0);
  fill(175);
  ellipse(x, height/2, 32, 32);
}
```

Dit ziet er nu een beetje onhandig uit, maar als je bijvoorbeeld met een ingewikkeld figuur werkt is het al overzichtelijker.

Zoals je kan zien lijken de setup() en draw() functies erg op de zelfgemaakte functies. Hieronder leg ik uit hoe je functies zelf kan bouwen.

```
returnType    functionName (arguments ) {
   // Code body of function
}
```

returnType is gerelateerd aan wat de functie terugstuurt. In de meeste gevallen stuurt het niks terug (zoals in het voorbeeld hierboven), dan is de returnType een void. Hieronder is een voorbeeld met een int als returnfunctie. De functie naam kan je zelf instellen, vergelijkbaar met variabel namen. Wel moet je opletten dat je geen systeem namen gebruikt (zoals setup, draw etc.).

```
void setup() {
  size(480, 270);
}

void draw() {
  background(255);
  stroke(0);

  float d = distance(width/2, height/2, mouseX, mouseY);
  fill(d*3, d*2, d);
  ellipseMode(CENTER);
  ellipse(width/2, height/2, 100, 100);
}

float distance(float x1, float y1, float x2, float y2) {
  float dx = x1 - x2;
  float dy = y1 - y2;
  float d = sqrt(dx*dx + dy*dy);
  return d;
}
```

Hier wordt de ABC-formule gebruikt om de afstand tussen twee punten te meten.

6.2 Theorie samenvatting HAVO

Uitleg benodigde functies/instructies

Hieronder staat kort uitgelegd wat jullie hebben geleerd.

Week 1

Vormen

`ellipse(x, y, breedte, hoogte)`

Hiermee teken je een ellips. Je moet 4 argumenten meegeven. Een x-coördinaat, een y-coördinaat, de breedte van de ellips en de hoogte van de ellips. De coördinaten die je moet meegeven zijn van de linkerbovenhoek van je ellips. Soms kan dit onhandig zijn, dan kun je de instructie gebruiken zoals die hieronder staat beschreven.

`ellipseMode(CENTER);`

De coördinaten die je moet meegeven met je ellips zijn van de linkerbovenhoek. Het is makkelijker als de coördinaten het middelpunt zijn van je ellips. Met de instructie/functie `ellipseMode(CENTER);` heb je dat ingesteld. Je hoeft hier verder niks aan te veranderen.

`ellipseMode(CORNER);`

Als je voorheen de functie `ellipseMode(CENTER)` hebt gebruikt, dan kun je deze functie gebruiken om het weer terug te zetten naar hoe het eerst was. De coördinaten die je dan meegeeft met je ellips, zijn de coördinaten van de linker bovenhoek van je ellips. Je hebt deze functie niet nodig bij je opdracht.

`rect(x, y, breedte, hoogte)`

Hiermee teken je een rechthoek. Je moet 4 argumenten meegeven. Een x-coördinaat, een y-coördinaat, de breedte van de rechthoek en de hoogte van de rechthoek.

Week 2 en 3

Interactie

Hieronder staan functies die in de lessen zijn behandeld.

```
size(breedte, hoogte);
```

Dit stelt de grootte van je scherm in. Het eerste argument, `breedte`, geeft aan hoeveel pixels breed je scherm moet zijn, en het tweede argument, `hoogte`, geeft aan hoeveel pixels lang/hoog je scherm moet zijn.

```
background(kleur);
```

Hiermee stel je een achtergrondkleur in voor je scherm. `kleur` neemt de waarde van een getal. Als het bijvoorbeeld 0 is, dan krijg je een zwarte achtergrond. Is het 255, dan krijg je een witte achtergrond.

```
stroke(kleur);
```

Hiermee geef je aan met welke kleur je lijnen wilt tekenen. Teken je bijvoorbeeld een rechthoek, waarbij de omtrek zwart is, dan stel je de waarde in op 0: `stroke(0).`

```
fill(kleur);
```

Dit gebruik om aan te geven welke kleur de invulling van je volgende vorm moet hebben. Als je een rechthoek tekent nadat je `fill(0)` hebt gedaan, dan krijgt je rechthoek een zwarte kleur.

```
void setup(){
	//Hier komt je code
}
```

Dit is een functie welk de compiler als eerste uitvoert. Deze functie wordt maar 1 keer uitgevoerd. Je kunt hierin alle code zetten waarmee je je programma opstart. Bijvoorbeeld, grootte van het scherm.

```
void draw(){
	//Hier komt je code
}
```

Dit is een functie welk de compiler herhaaldelijk uitvoert. Op het moment dat een compiler aan het eind van deze functie is gekomen, start het weer van begin af aan. Je kan hierin al je tekeningen zetten.

```
void mousePressed(){
	//Hier komt je code
}
```

De code binnen dit blok wordt uitgevoerd op het moment dat de muis wordt ingedrukt. De code zal éénmaal worden uitgevoerd.

```
void keyPressed(){
	//Hier komt je code
```

```
}
```
De code binnen dit blok wordt uitgevoerd op het moment dat een toets van het toetsenbord wordt ingedrukt. De code zal éénmaal worden uitgevoerd.

`mouseX`
De is de x-coördinaat van de muis. Dit is gewoon een getal. Je kan dit als argument gebruiken in plaats van een ander getal. Bijvoorbeeld: `rect(mouseX,2,5,5);`

`mouseY`
De is de x-coördinaat van de muis. Dit is gewoon een getal. Je kan dit als argument gebruiken in plaats van een ander getal. Bijvoorbeeld: `rect(2,mouseY,5,5);`

`rectMode(CENTER);`

De coördinaten die je moet meegeven met je rechthoek zijn van de linkerbovenhoek. Het is soms makkelijker als de coördinaten het middelpunt zijn van je rechthoek. Met de instructie/functie `rectMode(CENTER);` heb je dat ingesteld. Je hoeft hier verder niks aan te veranderen.

`rectMode(CORNER);`

Als je voorheen de functie `rectMode(CENTER)` hebt gebruikt, dan kun je deze functie gebruiken om het weer terug te zetten naar hoe het eerst was. De coördinaten die je dan meegeeft met je rechthoek, zijn de coördinaten van de linker bovenhoek van je ellips. Je hebt deze functie niet nodig bij je opdracht.

`line(x1,y1,x2,y2);`

Je gebruikt dit om een lijn te tekenen. Het heeft 4 argumenten nodig. Dit is iets anders dan de ellips en rechthoek. De eerste twee argumenten vormen de coördinaten van het beginpunt van je lijn en de laatste twee argumenten vormen de coördinaten van het eindpunt van je lijn.

`//`

Met `//` kun je commentaar geven. Alles wat na de `//` komt wordt genegeerd door de computer. Dit is handig om voor jezelf wat uitleg te geven. Zo snap je later nog wat je hebt gedaan.

Week 4

Variabelen

- Een variabele is een container met een naam, waarin je waardes kan opslaan.
- Een variabele heeft een:
  - **Type** (bijv. Integer, decimal, character,)
  - **Naam** (bijv. mouseX, mouseY)
  - **Inhoud** (bijv. 12, 14, 3.14, ‘a’)

Een variabele moet je declareren. Daarin geef je de type, naam en eventueel de inhoud mee. Bijvoorbeeld:

```
int x = 5;

int y;

y = 10;

char mijn_karakter = ‘a’;
```

Naam

Een variabele heeft een naam. Hiervoor gelden de volgende regels:

- De naam mag niet met een getal beginnen.
- Er mag geen interpunctie in voorkomen, (behalve lage streepje).
- Meestal begint het met een kleine letter.

Inhoud

Een variabele heeft een bepaalde waarde of inhoud. Een variabele kan op verschillende manieren een waarde worden toegekend. De inhoud:

- Kan een nieuwe waarde zijn
- Kan de waarde van een andere variabele zijn
- Kan een rekenkundige operatie zijn
- Kan een combinatie van alle drie zijn

Bijvoorbeeld:

```
int variabele_1 = 5;

int variabele_2 = variabele_1;

int variabele_3 = 5 + 13 * 5;

int variabele_4 = variabele_1 * 2 + 7;

variabele_1 = 5;

variabele_1 = 10;

variabele_1 = variabele_1 + 20;
```

Type

Een variabele kan van verschillende types zijn.

- Hele getallen
  - Engels: Integer
  - int
  - Voorbeelden: 5 ,12 ,6, 20000 etc.
- Komma getallen (in het Engels worden er punten gebruikt ipv komma's)
  - Engels: Floating point number
  - float
  - Voorbeelden: 3.14, -10.5, 21.245, -0.05
- Karakters
  - Engels: characters
  - char
  - Voorbeelden: 'a', 'b', 'z'

Voorbeeld:

```
char mijn_karakter = 'a';
char mijnKarakter = 'b';
float pi = 3.1415;
//Maar niet:
int pi_getal = 3.1415;
```

Week 5

If-Statements

Tijdens het programmeren zal je problemen tegenkomen waar je voorwaardelijke condities wilt stellen. Deze voorwaardelijke uitdrukkingen worden ook wel in het Engels ‘if-statements’ genoemd.

Je wilt bijvoorbeeld zeggen: “Als x kleiner is dan 5, doe dan dit”. Of, “Als y gelijk is aan 100, doe dan dit”. Weer een ander voorbeeld, “Als z niet gelijk is aan 20, doe dan dit”.

Dit soort condities en uitdrukkingen kun je ook in code uit drukken. Dit ziet er dan als volgt uit.

```
if(conditie){
      statement
}
```

Als de conditie tussen de haken waar is, dan wordt de statement uitgevoerd. Laten eerst kijken naar een simpele voorbeeld. Dit zal ik daarna stap voor stap uitleggen.

```
int x = 3;            //Stap 1
          if(x < 5){            //Stap 2 en 3
                rect(50,50,100,100); //Stap 4
          }               //Stap 5
```

Er gebeuren hier een aantal dingen. In totaal is dit kleine stuk code opgedeeld in 5 stappen.

1. Als eerste maken we een variabele aan. We noemen onze variabele x, en we geven het een waarde van 3. Met andere woorden, x is gelijk aan 3.
2. De tweede regel code bestaat uit 2 stappen. De eerste stap is de conditie. Er staat namelijk “`if(x < 5)`”. Vertaald naar het Nederlands staat er, “Als x kleiner is dan 5.” Het symbool “<” betekent “kleiner dan”. Als deze conditie, “`if(x < 5)`”, inderdaad waar is, dan wordt de code tussen de haken uitgevoerd. In onze geval is dit ook waar. Want, x is inderdaad kleiner dan 5. Vergeet niet dat x gelijk is aan 3, en 3 is kleiner dan 5.
3. Stap 3 is het openende haakje: “`{`”. Alle code dat hierna komt, wordt uitgevoerd als de conditie “`if(x < 5)`” waar is.
4. Met deze code wordt er een rechthoek getekend. Dit wordt alleen gedaan als de conditie “`if(x < 5)`” waar is. Dit is in ons geval zo. Dus er wordt een rechthoek getekend.
5. Dit is het sluitende haakje. Alles wat tussen het beginnende en sluitende haakje staan, wordt uitgevoerd als de conditie in de “if” waar is.

Hiermee wordt duidelijk hoe een if-statement gebruikt kan worden. Laten we kijken naar een andere voorbeeld.

```
int mijn_getal = 10;
if(mijn_getal > 50){
      mijn_getal = mijn_getal + 1;
}
```

In dit geval wordt er gezegd “Als mijn_getal groter is dan 50, verhoog dan de waarde van mijn_getal met 1”. In dit geval wordt de code tussen de haken niet uitgevoerd, omdat de conditie “`if(mijn_getal > 50)`” niet waar is. De variabele, “`mijn_getal`” is niet groter dan 50. Dus de code “`mijn_getal = mijn_getal + 1`” wordt niet uitgevoerd.

In de tabel hieronder staan de verschillende manieren waarop je condities kan uitdrukken.

Symbool	Betekenis
<	Kleiner dan
>	Groter dan
==	Gelijk aan
!=	Niet gelijk aan

Pijltjes toetsen gebruiken

Je kan de functie “keyPressed” gebruiken om de toetsenbord te gebruiken. Kijk naar dit voorbeeld.

```
void keyPressed(){
      //Code wordt alleen uitgevoerd op het moment dat er een toets //is
      ingedrukt.
}
```

De code binnen deze functie wordt alleen uitgevoerd op het moment dat er een toets wordt ingedrukt. Dus bijvoorbeeld, de spatieknop, enter, ‘v’, pijltje omhoog etc.

Op het moment dat je een knop indrukt, wordt er in de computer opgeslagen welke knop je hebt ingedrukt. Dit wordt opgeslagen in een variabele. De naam van de variabele is:

```
keyCode
```

Deze variabele hoef je niet aan het begin van je programma te maken, maar zit al standaard in het systeem. Men noemt het dan ook wel een ‘systeemvariabele’. Een andere voorbeeld van een systeemvariabele is, ‘mouseX’ en ‘mouseY’.

In de variabele worden geen getallen opgeslagen, maar letters (en de pijltjestoetsen). Een aantal voorbeelden van waarden zijn:

- `UP`
- `DOWN`
- `LEFT`
- `RIGHT`

Laten we kijken naar een voorbeeld. Op het moment dat je de pijltjestoets naar rechts indrukt, dan is de waarde van `keyCode` gelijk aan `RIGHT`.

Stel we willen hier nu gebruik van maken in onze programma. We maken een variabele aan waarin we x-coördinaat opslaan. Vervolgens maken we een rechthoek. We willen dan zeggen: “Als de pijltjestoets naar rechts wordt ingedrukt, verhoog dan de x-coördinaat”. Dan verandert de x-coördinaat van de rechthoek, en dan beweegt het naar rechts.

We hebben zoiets eerder gemaakt in onze animatie, maar nu gaan we het doen met onze toetsenbord.

Het ziet er dan als volgt uit.

```
int x_coordinaat = 50;

void setup(){
  size(500,500);          //Schermgrootte instellen
}

void draw(){
  background(255);           //Achtergrondkleur
  rectMode(CENTER);
  fill(0);                //Kleur van de rechthoek

  rect(x_coordinaat,100,50,50);  //Rechthoek
}

void keyPressed(){
  if(keyCode == RIGHT){      //Als de keyCode pijltje omhoog is
    x_coordinaat = x_coordinaat + 1;   /verhoog dan de x-coördinaat
  }

}
```

In de keyPressed functie passen we toe dat we pijltjestoetsen kunnen gebruiken. Als de keyCode gelijk is aan pijltje omhoog, dan wordt de code “`x_coordinaat = x_coordinaat + 1;`” uitgevoerd. Dit zorgt ervoor dat de rechthoek naar rechts beweegt. Op dezelfde wijze kunnen we ook de vierkant naar boven, beneden en naar links laten bewegen.

Meerdere condities

Soms wil je meerdere condities meegeven. Bijvoorbeeld, “Als de x-coördinaat kleiner is dan 300 **en** als de pijltjestoets omhoog wordt ingedrukt, voer dan dit uit”. Dit doen we als volgt.

```
if(keyCode == RIGHT && x_coordinaat < 300){
    x_coordinaat = x_coordinaat + 1;
}
```

De “&&” is de ‘en’ teken in code taal. Zo kunnen we meerdere condities meegeven. Je kunt ook een ‘of’ gebruiken. Dit staat in de tabel hieronder.

Symbool	Betekenis
&&	En
||	Of

Meerdere if-statements

Ja kan meerdere if statements binnen elkaar gebruiken. Bijvoorbeeld:

```
if(keyCode == RIGHT){
   if(x_coordinaat < 300){
      x_coordinaat = x_coordinaat + 1;
   }
}
```

Voor meer informatie kun je altijd naar:

https://processing.org/reference/

11.2 Enqueête: Evaluatie Module Programmeren

Evaluatie Module Programmeren

Fijn dat jullie ons willen helpen deze module programmeren te evalueren! Zo kunnen wij er van leren en de module nog beter maken!

*Required

1. **Wat is jouw leeftijd?** *

2. **In welke klas zit je?** *
   Mark only one oval.
   - ◯ HAVO 4
   - ◯ VWO 4

3. **Ik ben een** *
   Mark only one oval.
   - ◯ Man
   - ◯ Vrouw

Doelen van de cursus

4. **De leerdoelen van de cursus waren voor mij duidelijk.** *
   Mark only one oval.

	1	2	3	4	5	
Helemaal mee oneens	◯	◯	◯	◯	◯	Helemaal mee eens

5. **Ik heb voldoende kennis opgedaan in deze module.** *
   Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

6. **Ik heb voldoende inzicht gekregen in programmeren.** *
   Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

7. **Ik heb voldoende vaardigheden ontwikkeld.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

8. **Het niveau van de module was** *
Mark only one oval.

	1	2	3	4	5	
Veel te laag	◯	◯	◯	◯	◯	Veel te hoog

9. **Ik had voldoende voorkennis om deze module te volgen.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

10. **Deze module is goed afgestemd op voorgaande modules.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

11. **Deze module helpt mij te begrijpen hoe programmeren terugkomt in de samenleving.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

12. **Ik vond het leuk om te leren programmeren.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

13. **Ik heb een idee van wat ik met programmeren zou kunnen doen.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

14. **Iedereen zou moeten leren programmeren.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

Onderwijsproces en Studiemateriaal

15. **De module was goed georganiseerd.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

16. **Het studiemateriaal sloot goed aan bij de module.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

17. **Het niveau van het studiemateriaal was** *
Mark only one oval.

	1	2	3	4	5	
Veel te laag	◯	◯	◯	◯	◯	Veel te hoog

18. **De hoeveelheid studiemateriaal was** *
Mark only one oval.

	1	2	3	4	5	
Veel te veel	◯	◯	◯	◯	◯	Veel te weinig

Lessen en opdrachten

19. **De lessen waren goed** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

20. **Het niveau van de lessen was** *
Mark only one oval.

	1	2	3	4	5	
Veel te laag	◯	◯	◯	◯	◯	Veel te hoog

21. **De lessen hadden een toegevoegde waarde.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

22. **De docent bracht de stof helder over.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

23. **De opdrachten waren duidelijk.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

24. **De docent(en) begeleidde(n) de opdrachten goed.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

25. **De opdrachten sloten goed aan bij de inhoud van de module.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

26. **Ik had genoeg tijd voor de opdrachten.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

27. **Ik vond het leuk om aan de opdrachten te werken.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

28. **De feedback van de docent(en) maakte de plus- en minpunten van mijn werk duidelijk.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

Toepassingen en Bruikbaarheid

29. **Ik denk dat ik vaak zal blijven programmeren.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

30. **Ik vind programmeren onnodig complex.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

31. **Ik vind programmeren makkelijk.**
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

32. **Ik denk dat ik de hulp van iemand anders nodig zal hebben bij het programmeren.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

33. **De verschillende onderdelen van de module waren goed geïntegreerd.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

34. **Ik vond dat er teveel inconsistenties waren in de module.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

35. **Ik denk dat de meeste mensen snel kunnen leren programmeren.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

36. **Ik vond programmeren lastig om te leren.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

37. **Tijdens het programmeren ben ik zelfverzekerd.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

38. **Ik moest veel leren voordat ik aan de slag kon met programmeren.** *
Mark only one oval.

	1	2	3	4	5	
Helemaal mee eens	◯	◯	◯	◯	◯	Helemaal mee oneens

Rapportcijfer

39. **Wat voor een rapportcijfer zou je de module geven?** *
Mark only one oval.

- ◯ 10
- ◯ 9
- ◯ 8
- ◯ 7
- ◯ 6
- ◯ 5
- ◯ 4
- ◯ 3
- ◯ 2
- ◯ 1

40. **Wat zijn de goede punten van de module?** *

41. **Wat zijn verbeterpunten van de module?** *

42. **Hoe zou je de module omschrijven aan bijvoorbeeld vrienden die geen informatica volgen?** *

43. **Heb je nog opmerkingen?**

11.3 Expert Review - Erik Barendsen

Je module ziet er interessant uit! Ik heb een paar opmerkingen erover:

- Jullie hebben op een kansrijke manier geprobeerd te differentiëren tussen havo en vwo. Ik ben benieuwd hoe dat in de klas is uitgekomen.
- Functies met parameters zijn lastig. Jullie vwo-lln zijn er niet allemaal aan toe gekomen, las ik. Het is goed dat de havo-versie er niet van afhankelijk is.
- Ik mis expliciete leerdoelen. Wat moeten volgens jullie de lln aan het einde kunnen?
- De aanpak met stapsgewijze opdrachten (doe eerst dit, dan dat) werkt voor veel leerlingen, maar niet voor allemaal. Het verdient aanbexveling om expliciet reflectie in te bouwen: bijvoorbeeld voorspellen wat er gebeurt, verklaren wat je ziet, een meta-vraag over een aanpak stellen, etc.
- Het is jammer dat algoritmische aspecten niet aan bod komen. Algoritmen bedenken en formuleren kan bovendien voorkomen dat lln zonder na te denken achter het toetsenbord springen.

11.4 Expert Review - Wim Nijhuis

1. *Wat was je algemene indruk van de module?*

   Positief, leerlingen hebben voldoende idee gekregen van de basisprincipes programmeren. De combinatie programmeren en creativiteit was mooi

2. *Hoe vergelijkt de module met hoe je het zelf afgelopen jaren hebt gedaan?*

   Module is niet vergelijkbaar met afgelopen jaren. Dit was meer 'traditioneler', redelijk vanuit de 'harde' = beta-kant benaderd. Vandaar dat jullie opzet verfrissend was.

3. *Zou je het zelf gebruiken? En waarom?*

   Ik zou het graag willen gebruiken. Maar het behoeft wel een vervolg.

4. *Wat zijn verbeterpunten van de module?*

   Nu zijn de lessen in elkaar gezet op grond van de ervaring tijdens de lessen. Daarmee was het uiteindelijke doel (wat wil je bereiken na een serie lessen) niet helder. Hadden jullie dit zelf wel?

5. *Hoe vond je dat de klassen erop reageerde? (Bijv. vonden ze het wel of niet leuk?)*

   Leerlingen hebben het leuk gevonden, het waren ongedwongen lessen.

6. *Wat vond je van het niveau van de module?*

   In leerjaar 4 als introductie programmeren prima, maar nogmaals, er hoort een vervolg op te komen. Qua diepgang is er nog wel wat op te merken, leerlingen uit leerjaar 2 zouden het zonder probleem kunnen volgen.

7. *Wat vond je van de differentiatie tussen HAVO en VWO?*

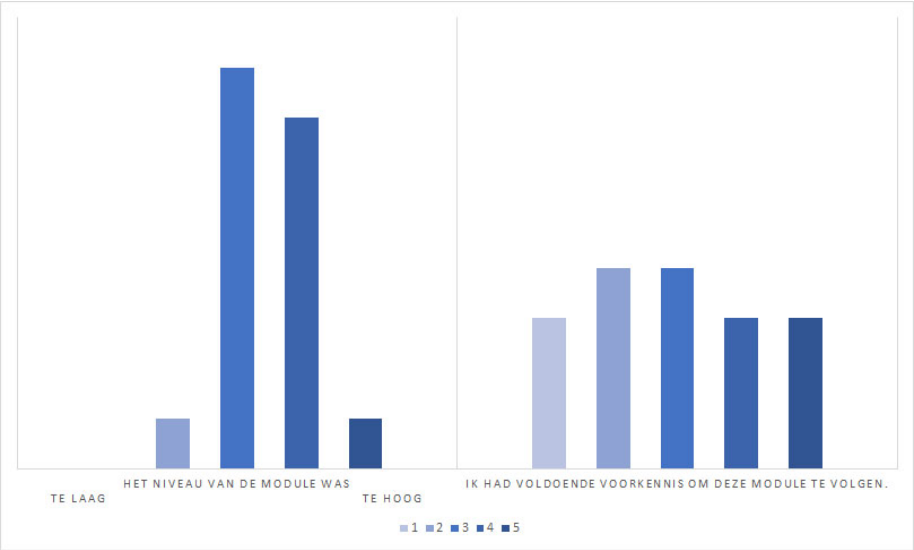
   Hier ligt wel een verbeterpuntje, vwo had mi wel wat meer uitdaging aangekund.

11.5 Resultaten van de Enquête

11.5.1 Toegankelijkheid

Enquete vragen

(a) HAVO



(b) VWO

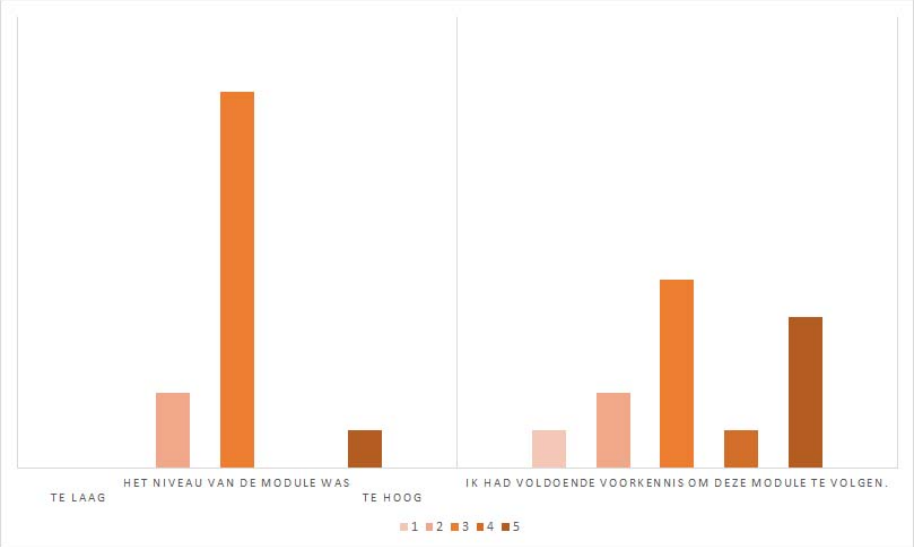
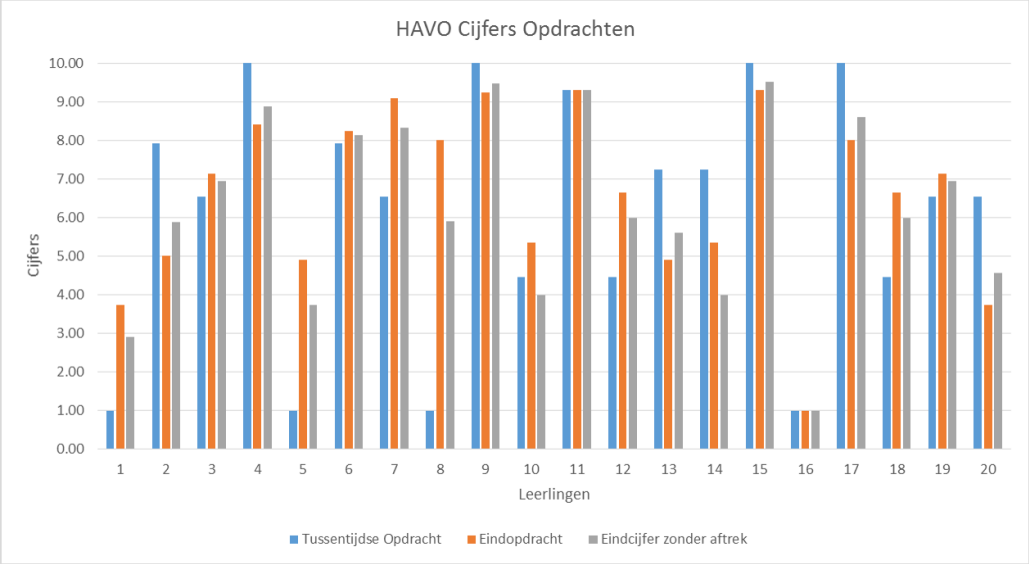


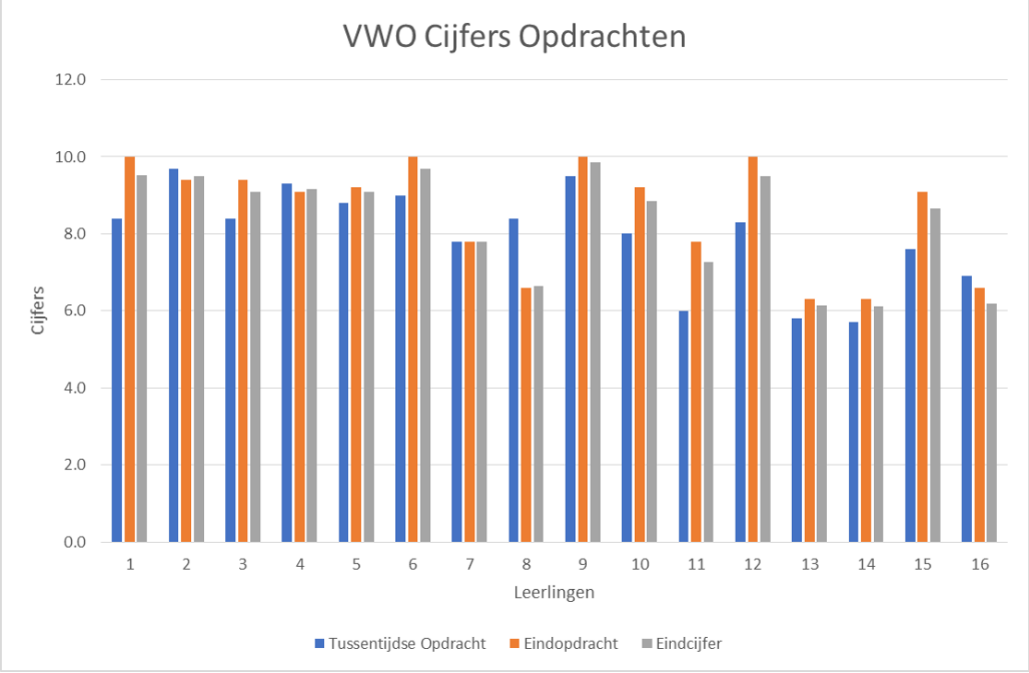
Figure 3: Resultaten voor "Het niveau van de module was te laag/te hoog" en "Ik had voldoende voorkennis om deze module te volgen" voor havo (3a) en vwo (3b)

Cijfers Leerlingen

(a) HAVO Cijfers

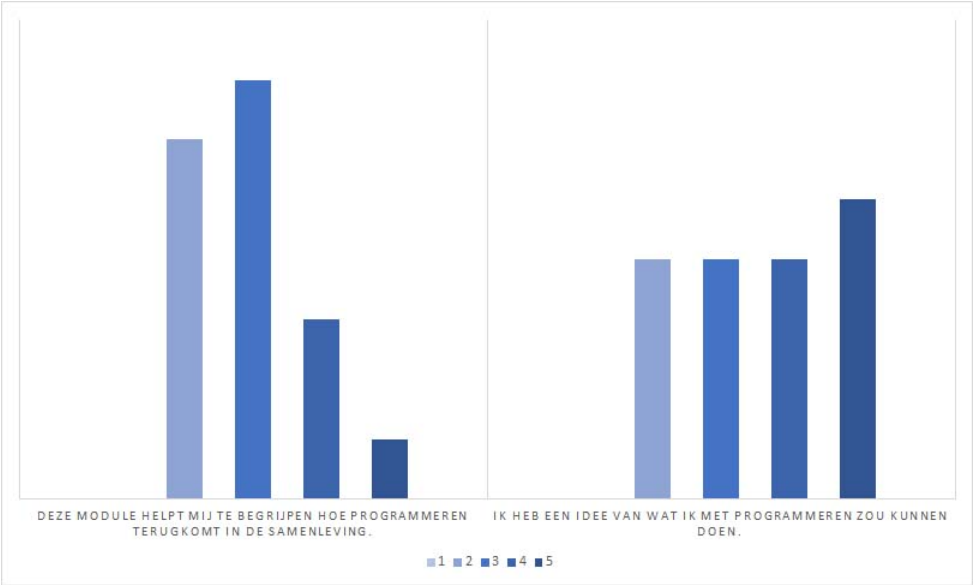


(b) VWO Cijfers



11.5.2 Maatschappelijke Relevantie

(a) HAVO



(b) VWO

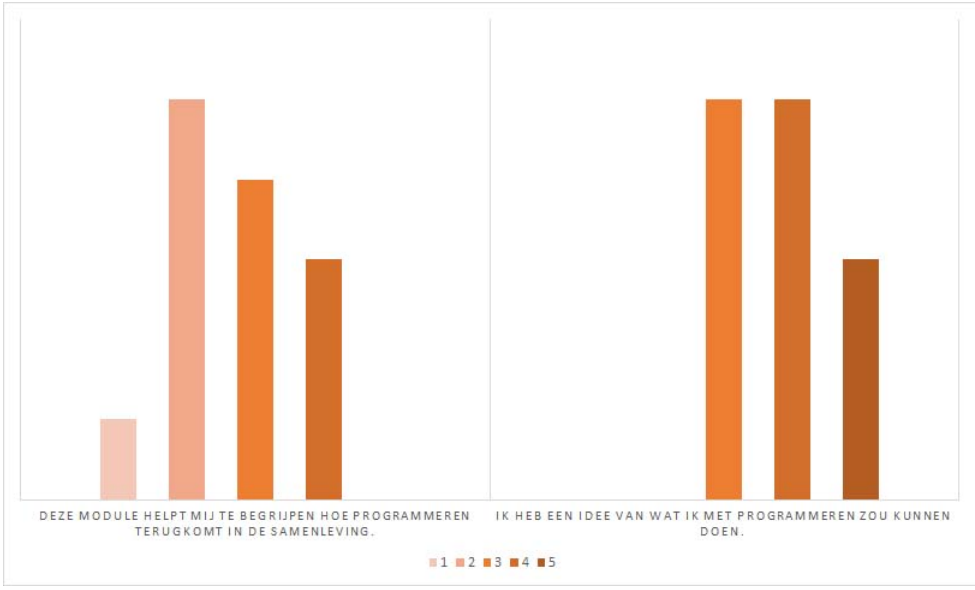
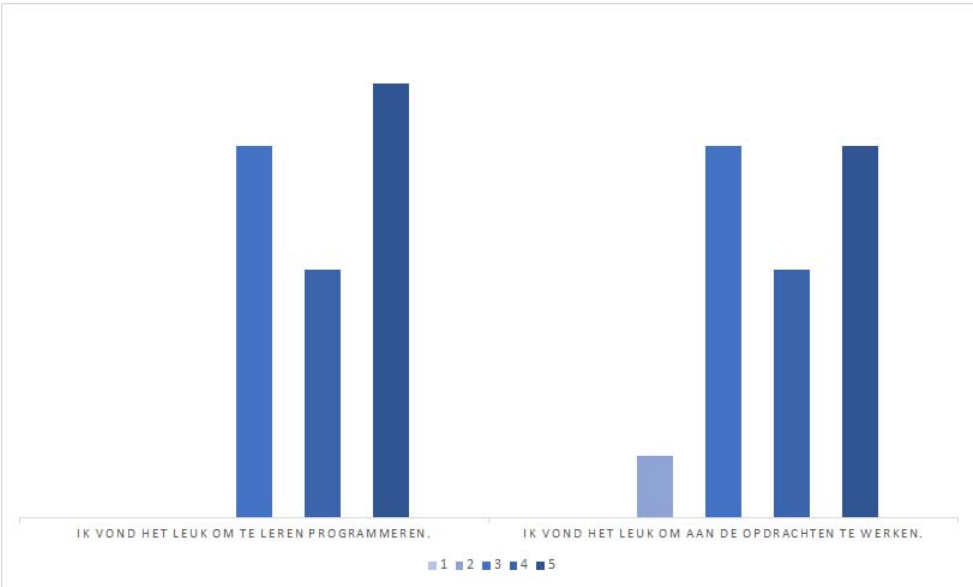


Figure 5: Resultaten van "Deze module helpt mij te begrijpen hoe programmeren terugkomt in de samenleving" en "Ik heb een idee van wat ik met programmeren zou kunnen doen" voor maatschappelijke relevantie voor havo (5a) en vwo (5b)

11.5.3 Plezier

(a) HAVO



(b) VWO

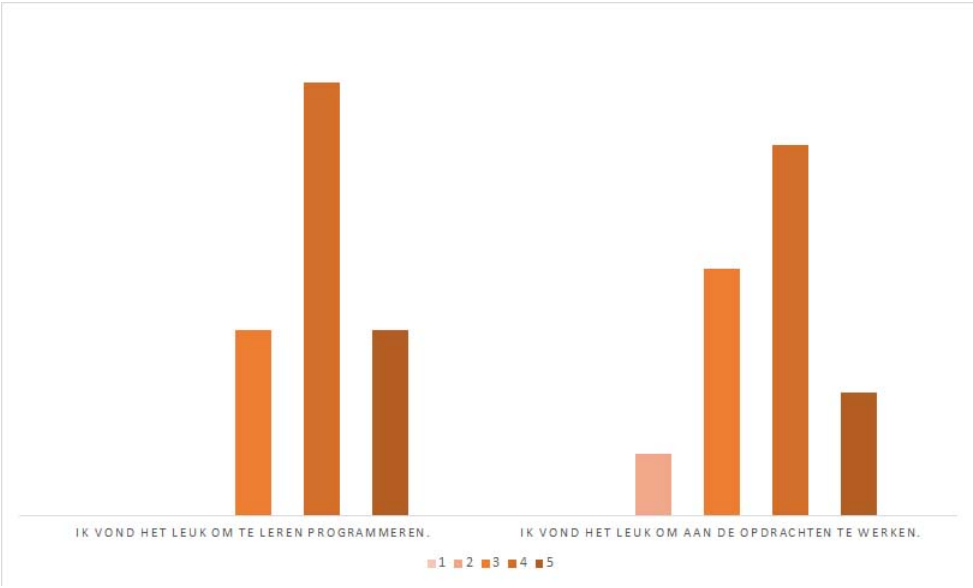


Figure 6: Resultaten voor het ervaren plezier van de havo leerlingen (6a) en de VWO-leerlingen (6b)

11.5.4 Tevredenheid

Enquête vraag: De lessen waren goed

(a) HAVO (b) VWO

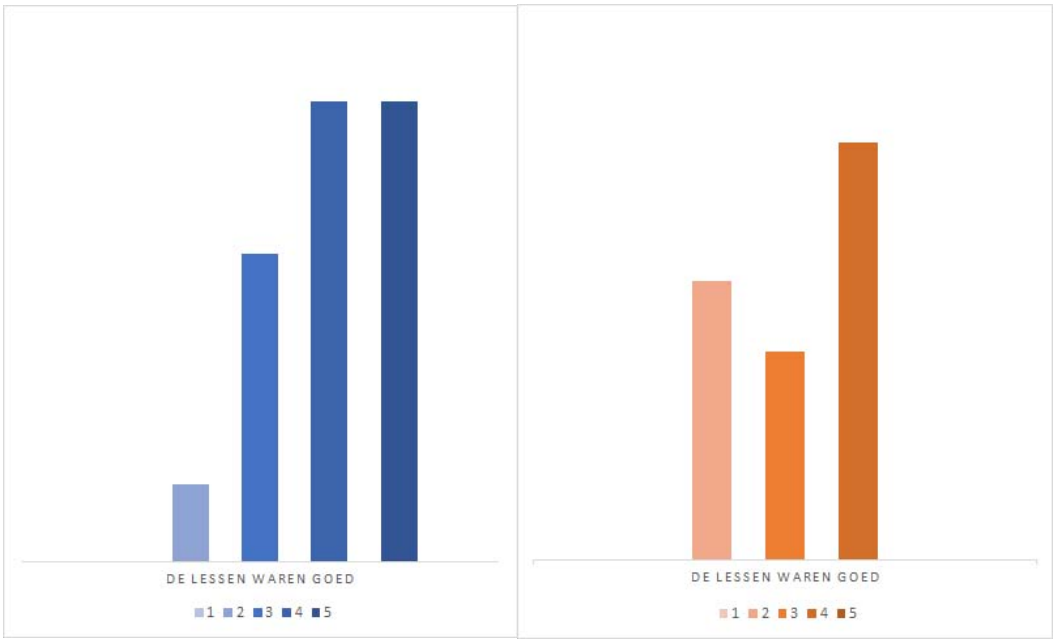


Figure 7: Resultaten voor "De lessen waren goed" voor havo (7a) en vwo (7b)

Rapportcijfer

SUS Score

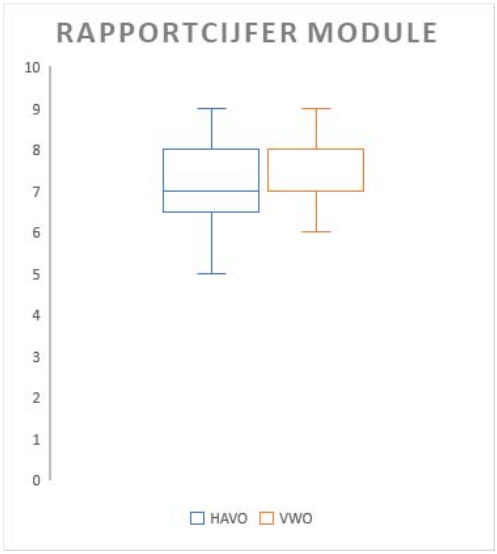


Figure 8: Het rapportcijfer die de leerlingen uit havo (blauw) en vwo (oranje) aan de module hebben gegeven

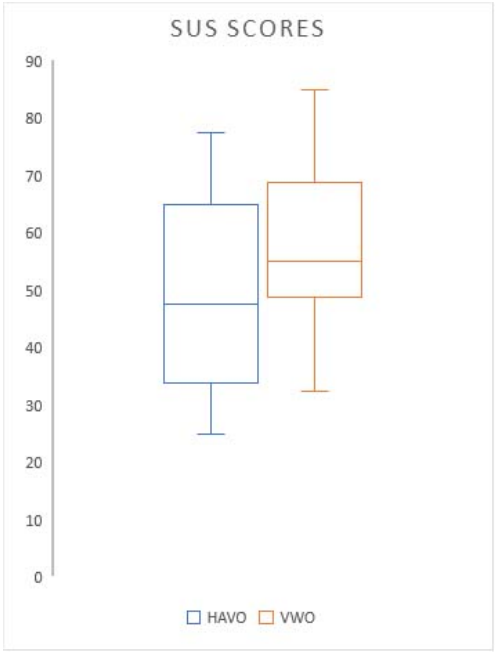
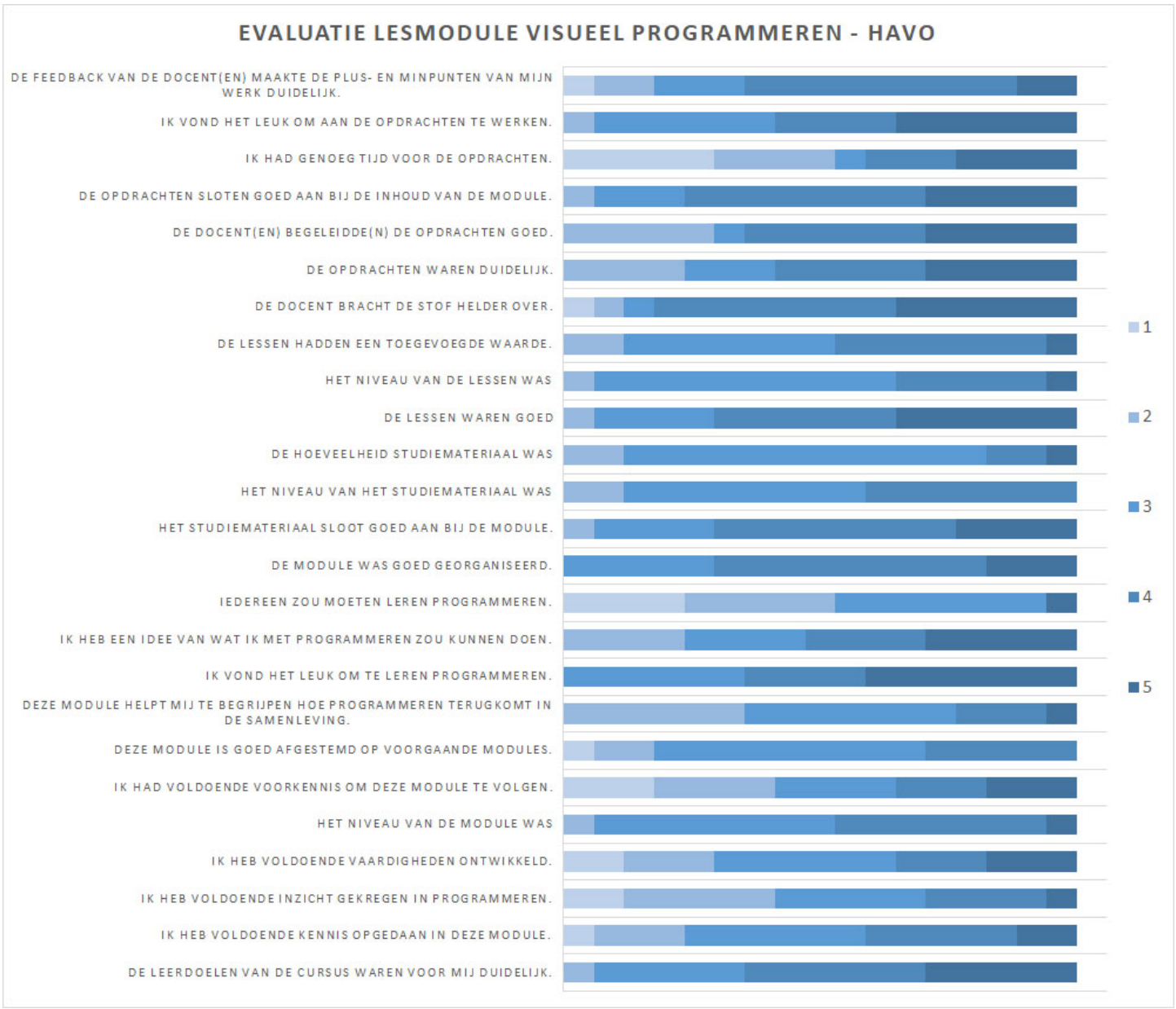


Figure 9: De SUS Scores voor HAVO (blauw) en VWO (oranje)

11.5.5 Alle resultaten HAVO

Figure 10: Resultaten HAVO groep gesloten vragen



11.5.6 Alle resultaten VWO

Figure 11: Resultaten VWO groep gesloten vragen