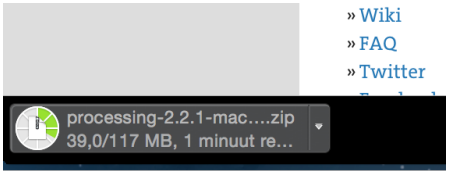
WELKOM BIJ UNICODING

Bedankt voor het kiezen van Unicoding – Coding for Kids. Unicoding is een workshop coderen voor basisscholieren van groep 8. In de cursus leren de leerlingen stap voor stap hoe ze een spelletje kunnen maken, via de programmeer taal van Processing. Via deze handleidingen kunt u als leraar zelf een workshop voorbereiden en uitvoeren in uw klas.

Na de eerste les weten de leerlingen al hoe ze vormpjes kunnen tekenen, wat een x- en y-stelsel is en hoe je kunt kleuren met de zogenaamde RGB-kleuren in Processing.

PROCESSING

We beginnen met de software. Processing is een gratis programma, met een eigen en relatief simpele programmeertaal. Processing is de perfecte tool voor de eerste kennismaking met coderen.



Processing kan gedownload worden vanaf deze website. Klik vervolgens op (Donate &) Download en kies daarna voor welke computer u Processing wilt downloaden. Na gekozen te hebben uit Windows, Linux of Mac OSX start de download automatisch.

Na Processing te hebben gedownload, opent het in een zip-map. Pak Processing uit (voor Windows en Linux. Mac doet dit automatisch) en daarna kunt u Processing opstarten. Het scherm ziet als volgt uit:

Bovenin staan elke knoppen. Deze betekenen (van links naar rechts):

- Run (voert de code uit)
- Stop (stopt het uitvoeren)
- New (opent nieuw bestand)
- Open (opent een bestand)
- Save (slaat het bestand op)
- Export (exporteer spel)

In het witte vlak komt de code te staan. In het zwarte vlak komen eventuele foutmeldingen te staan.

Nu dat we Processing succesvol geïnstalleerd hebben, kunnen we beginnen met het tekenen van vormen met behulp van code.

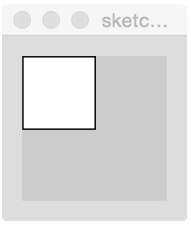
VORMEN

In Processing is het mogelijk om vormen te tekenen. In deze les leggen we eerst uit hoe u rechthoeken en ellipsen kunt coderen.

Een rechthoek (*rectangle* in Engels) heeft de afkorting rect. We beginnen met een rechthoek van 50 pixels breed en 50 pixels hoog. Neem de volgende regel over:

```
rect(0, 0, 50, 50);
```

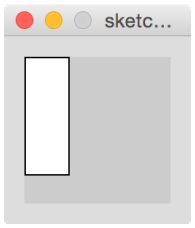
Rect betekent dat we een rechthoek gaan tekenen. De haakjes geven aan dat die rechthoek uitgedrukt gaat worden in waardes (cijfers). De eerste en tweede waarde bepalen de positie. Deze wordt later deze les uitgelegd. De derde waarde geeft aan hoe breed de rechthoek moet worden. In dit geval is dat 50 pixels. De laatste waarde bepaalt de hoogte. Hier is dat ook 50 pixels.



Wanneer we deze cijfers veranderen, verandert ook de vorm en de grootte van de rechthoek. Neem bijvoorbeeld de volgende regel over:

```
rect(0, 0, 30, 80);
```

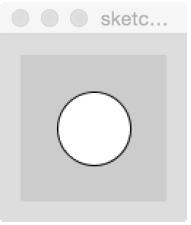
Nu zien we dat het vierkantje van 50 bij 50 een rechthoek is geworden van 30 pixels breed en 80 pixels lang. Laat de leerlingen vooral met deze cijfertjes spelen, waardoor ze zien dat de grootte steeds anders wordt. Wanneer ze een waarde boven de 100 invullen, past het rechthoekje zelfs al niet meer in het venster, maar dat is geen enkel probleem.



Nu gaan we verder met een ellips (*ellipse* in het Engels). Een ellips is een ronde of ovale vorm. We gaan nu een cirkel tekenen met hulp van een ellips. Om de de cirkel mooi in het midden te krijgen, veranderen we de eerste en tweede waarde alvast:

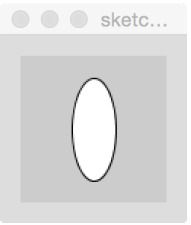
```
ellipse(50, 50, 50, 50);
```

En klaar is kees! In het midden van het venster staat nu een cirkel van 50 pixels hoog en 50 pixels breed. Wederom kunnen we leerlingen nu kijken wat er gebeurt als ze de derde en vierde waardes aanpassen. Als we verschillende getallen gebruiken, zal de cirkel een ovaal worden. Bijvoorbeeld:




```
ellipse(50, 50, 30, 70);
```

Zoals u ziet is de cirkel nu een ovaal geworden. Wanneer de kinderen genoeg geoefend hebben met vormpjes kunnen we door met een iets moeilijkere stap: coördinaten.



COÖRDINATEN

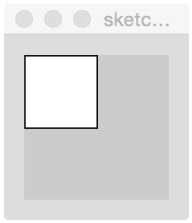
Een beeldscherm is opgebouwd uit pixels. Een pixel is een stipje in het scherm die een kleur kan aannemen. Meer over kleuren volgt later in deze les. We beginnen met het zetten van een stip in Processing:

```
point(10, 10);
```

Het is haast niet zichtbaar, maar linksboven in het scherm staat nu een klein puntje. Dit punt is dus (10, 10) en dat houdt in dat de punt 10 pixels naar rechts staat, gezien vanaf de rand en 10 pixels naar onder, gezien vanaf boven.

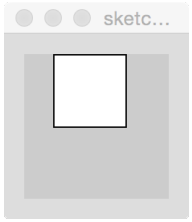
De eerste waarde is de zogenaamde x-waarde en de tweede waarde is de y-waarde. De x-as loopt van links naar rechts. Wanneer je het eerste getal verandert, zal de vorm dus opzij bewegen. De y-as loopt van boven naar onder. Wanneer de het tweede getal verandert zal de vorm dus in hoogte verschuiven. We nemen de *rectangle* uit de vorige paragraaf als voorbeeld:

```
rect(0, 0, 50, 50);
```



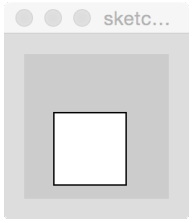
De eerste waarde is dus de x-waarde. Wanneer we deze verhogen zal de rechthoek naar links of naar rechts bewegen. In dit geval kunnen we alleen naar rechts. We tellen 20 op bij de waarde:

```
rect(20, 0, 50, 50);
```



Het vierkantje is 20 pixels naar rechts opgeschoven. We gaan nu ook nog de tweede waarde verhogen, zodat het vierkantje omlaag beweegt. Voor dit voorbeeld laten we het vierkantje 40 pixels naar onder bewegen. De nieuwe code wordt:

```
rect(20, 40, 50, 50);
```

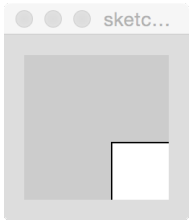


Er zit nu een ruimte van 40 pixels tussen de bovenkant van het venster en het vierkantje.

Nu weten we hoe we een vierkantje kunnen verplaatsen, maar waarom doet hij dat eigenlijk? De eerste en tweede waarde, zijn het punt vanaf waar het vierkantje getekend moet worden. Het punt (20, 40) is de linkerbovenhoek van het vierkantje.

De grootte van het venster is standaard 100 bij 100 pixels. Als we het vierkantje het startpunt (60, 60) geven zal het niet meer helemaal in het venster passen:

```
rect(60, 60, 50, 50);
```



Om het vierkantje wel volledig te laten passen, zullen de x-waarde en y-waarde onder 50 moeten blijven (want 50 opzij + 50 breed of hoog = 100) of moeten we het venster groter maken:

```
size(200, 200);
```

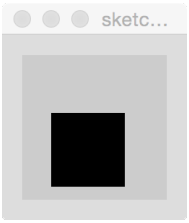
Door deze regel helemaal boven te zetten, wordt het venster vier keer zo groot dan het was. En nu past het vierkantje wel! Laat de leerlingen zelf eens met de startwaarden en *size* spelen, om door te krijgen hoe coördinaten werken.

RGB-KLEUREN

Op een digitaal scherm, zoals het beeldscherm van een laptop, computer, televisie, maar ook op telefoons en tablets, wordt een kleur uitgedrukt in een RGB-waarde. RGB staat voor de kleuren waaruit deze waarden zijn opgebouwd: rood, groen en blauw.

We kunnen de kleuren toepassen op de opvulling van vormen. De vormen die we al hebben geleerd zijn rechthoeken en ellipsen. Belangrijk is dat we eerst opschrijven welke kleur de vorm moet krijgen, voordat ze de vorm zelf opschrijven. Omdat de computer van boven naar beneden leest, moet hij eerst weten hoe alles uit moet komen te zien, voordat hij kan beginnen met tekenen:

```
fill(0, 0, 0);
rect(20, 20, 50, 50);
```

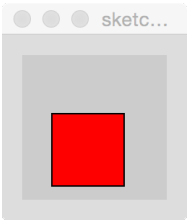


We hebben nu een vierkantje getekend met een zwarte opvulling. Door de rgb-waarde allemaal 0 te maken, hebben we gekozen voor een opvulling zonder kleur. Hoe lager de rgb-waarden, hoe doffer de kleur. Alleen maar nullen zorgt voor zwart.

RGB-waarden gaan van 0 tot 255. Wanneer we alle waardes in 255 veranderen, krijgen we een witte opvulling. Hoe dichter de waarden bij 255 liggen, hoe feller de kleur. Een hogere waarde betekent ook dat die kleur meer gebruikt wordt.

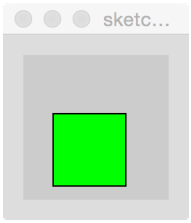
We gaan nu een rood vierkantje maken. We verandere de waarden van de *rectangle* niet, alleen die van de *fill*:

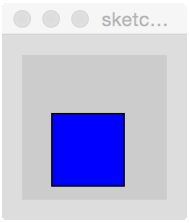
```
fill(255, 0, 0);
```



We gebruiken de rode kleur nu volledig en de groene en blauwe kleuren staan uit (0), dus zien we alleen een pure rode kleur.

Dit kunnen we ook doen met alleen groen en alleen blauw. Verander de andere waarden naar 255 en zorg dat de rest dan op 0 staat:






```
fill(0, 255, 0);
```

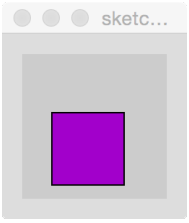
Nu hebben we een groen vierkantje. Tenslotte vullen we voor blauw de volgende regel in:

```
fill(0, 0, 255);
```

Alle 3 de basiskleuren hebben we nu gehad. Maar de waarden kunnen ook allemaal verschillend zijn.

Om deze kleur na te maken gebruiken we de volgende waarden:

```
fill(160, 0, 200);
```



Laat je leerlingen zelf experimenteren met de waardes en ervaren welke kleuren ze allemaal kunnen maken. Als ze alle drie de waardes gelijk maken, zien ze dat ze grijswaarden creëren, maar daar komen we in een andere les op terug.

TERUGBLIK OP LES 1 DEEL A

In deze eerste les deel A hebben we gekeken naar twee van de vormen die we kunnen maken in Processing, hoe we deze vormen kunnen plaatsen en van grootte laten veranderen en hoe ze deze vormen kunnen kleuren.

Er kwamen veel cijfers voorbij. Cijfers zijn cruciaal in coderen, want ze worden voor alles gebruikt. Het is geen probleem om te experimenteren met cijfers. Op deze manier zie je wat cijfers betekenen en wat ze doen.

WAT GAAN WE DOEN IN LES 1 DEEL B?

Les 1 deel B is alvast een uitstapje van wat we al hebben geleerd en wat we nog gaan doen. In deze les hebben we al uitgelegd wat enkele functies zijn en wat we ermee kunnen. In les 1 deel B gaan we al een spelletje maken. Zonder veel te hoeven coderen, kunnen de leerlingen de volgende keer alvast ervaren wat er mogelijk is na een cursus van enkele lessen. Alle leerlingen kunnen een spel van onze website downloaden, zonder opmaak en figuurtjes waarna ze die zelf gaan implementeren.

De basis van het spelletje is die van ons, na 10 lessen Processing te hebben gehad. Het spelletje wordt een zogenaamde "*dodger / catcher*" waarbij de speler voorwerpen die omlaag vallen moet ontwijken of vangen door opzij te bewegen. De basis van het spelletje en de handleiding voor les 1 deel B, vindt u op onze website onder het kopje "Downloads."

Alle handleidingen, downloads en meer vindt u op unicoding.nl