Kris Merckx

SMART

We leven in een hoogtechnologische omgeving. Dit maakt sommige mensen bang en onzeker.
Alles wordt smart: onze telefoons, computers, televisies, auto's… Maar vaak hebben we het gevoel niet meer mee te kunnen, dom te zijn in een smart omgeving.

Dit boek is jouw wegwijzer door de smart wereld. Het is eveneens een kookboek dat je leert koken met alle smart ingrediënten…

Een kok moet zijn ingrediënten kennen om beter te leren koken.

Leren koken

INTRO

- **Wat zijn microcontrollers en embedded systemen?**
- **Wat is een huisstijl?**
- **Wat is het verschil tussen vectorafbeeldingen en pixelafbeeldingen?**
- **Wat kan AI en wat kan het niet?**
- **...**

Vergelijk dit boek met een opleiding voedingsleer. We leren je proeven van diverse ingrediënten. We leren welke ingrediënten goed samen passen, hoe je ze bewerkt en samenstelt. We leren ook al een klein beetje koken, je maakt kleine multimedia-recepten met je eigen favoriete ingrediënten.

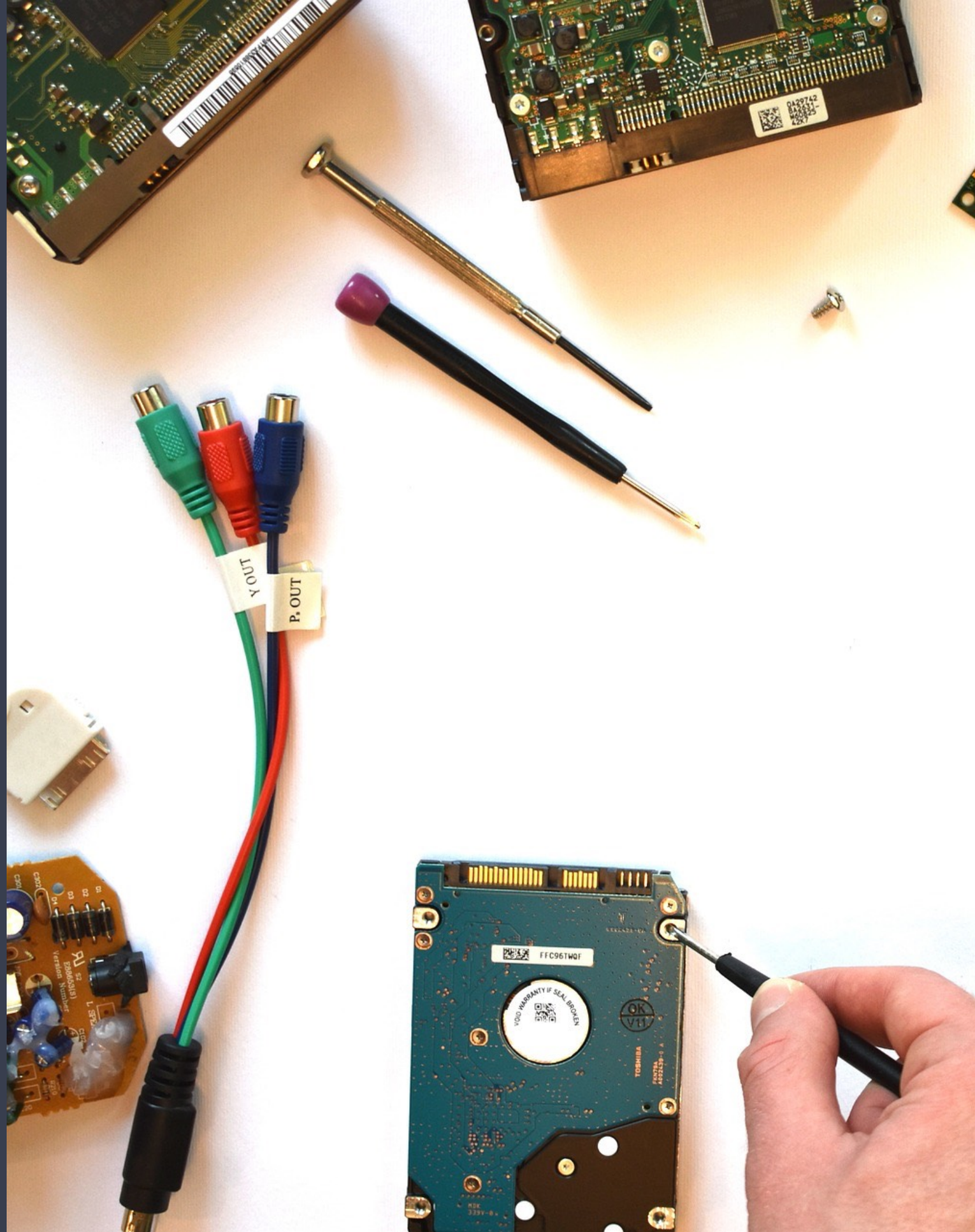
Je leert de smaak te pakken krijgen, je selecteert, proeft, kookt. Wie goed wil koken, moet niet alleen de ingrediënten kennen, maar de gepaste bereidingswijzen. Je moet je kookgerei leren kennen: hoe werkt het fornuis, de bakoven, messen, snijgereedschap enz.

Natuurlijk leren we niet echt koken. Je leert om te gaan met digitale systemen, je leert wat je ermee kan doen. Uiteraard gaan we daarin verder dan je normale computerkennis en -ervaring. Iedereen kan in een tekstverwerker een tekst invoeren, net zoals iedereen wel een ei kan bakken.

Als je dit boek leest, of delen ervan, heb je als startende multimediale kok de nodige kennis en ervaring om aan de slag te gaan met video, audio, animatie, grafische vormgeving, webdesign...

DEEL 1
HARDWARE

- Wat zijn microcontrollers?
- Wat zijn sensoren en actuatoren?
- Wat zijn DNA- en quantumcomputers?
- …



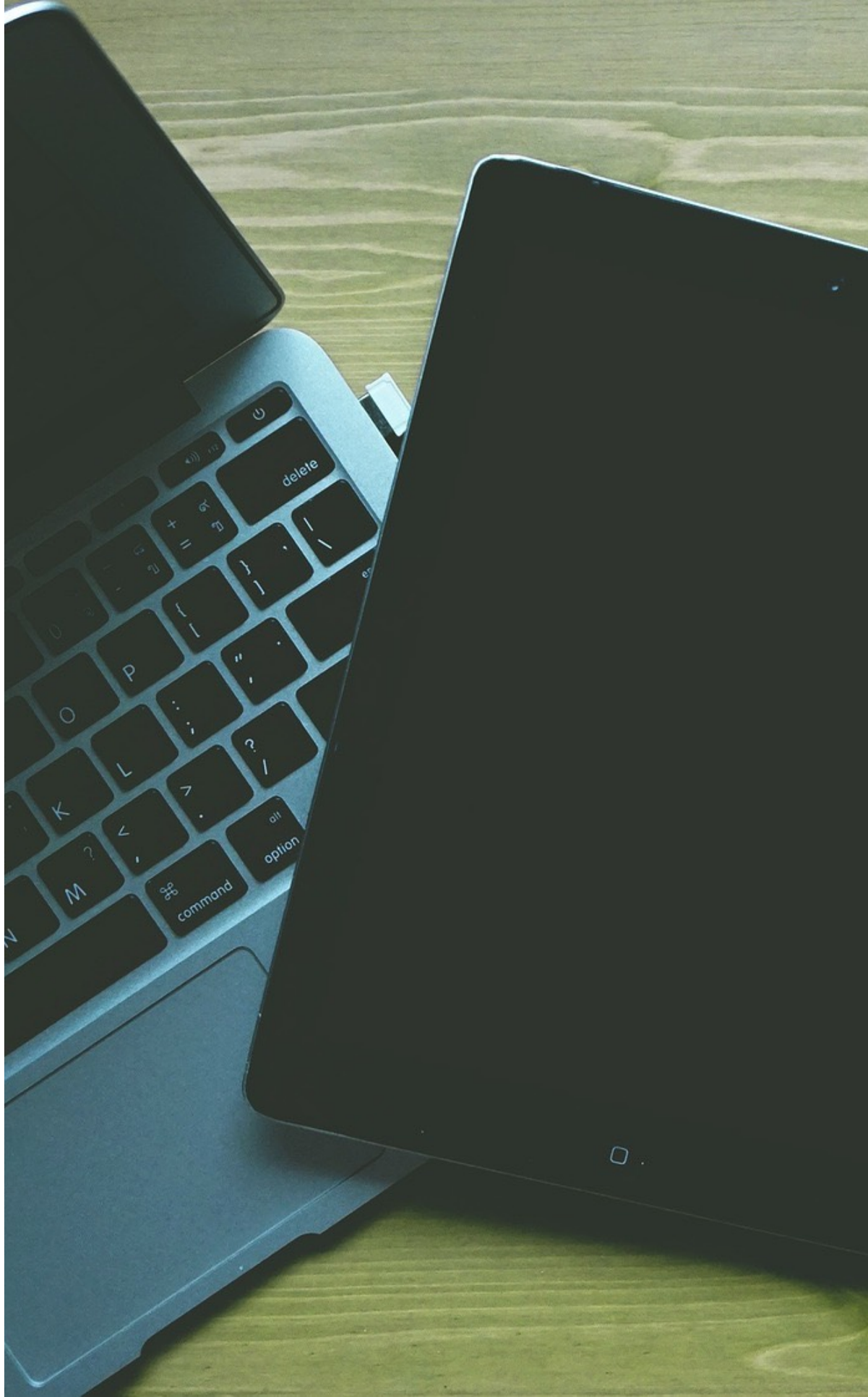
COMPUTERS IN MATEN EN GEWICHTEN

Computers hebben niet noodzakelijk een scherm, computers bedien je niet sowieso met een muis en toetsenbord. Bovendien zijn computers niet nieuw.

Een computer is een systeem, over het algemeen een elektronische en/of digitaal systeem, *dat invoer verwerkt tot uitvoer*. Zo hoor je het toch vaak. Maar zo'n definitie is vooral heel erg vaag.

Een eenvoudig voorbeeld: ik tik deze tekst in op een toetsenbord (invoer). De computer verwerkt mijn invoer. Het toestel registreert elke toetsaanslag en plakt die achter elkaar in een tekstdocument. Ik kreeg het eindresultaat (uitvoer, output) meteen te zien op mijn computerscherm. Ik kan er ook voor kiezen om het document te bewaren, af te drukken of te verspreiden als bijvoorbeeld een PDF-bestand.

Heel eenvoudig dus: invoer, verwerking, uitvoer…



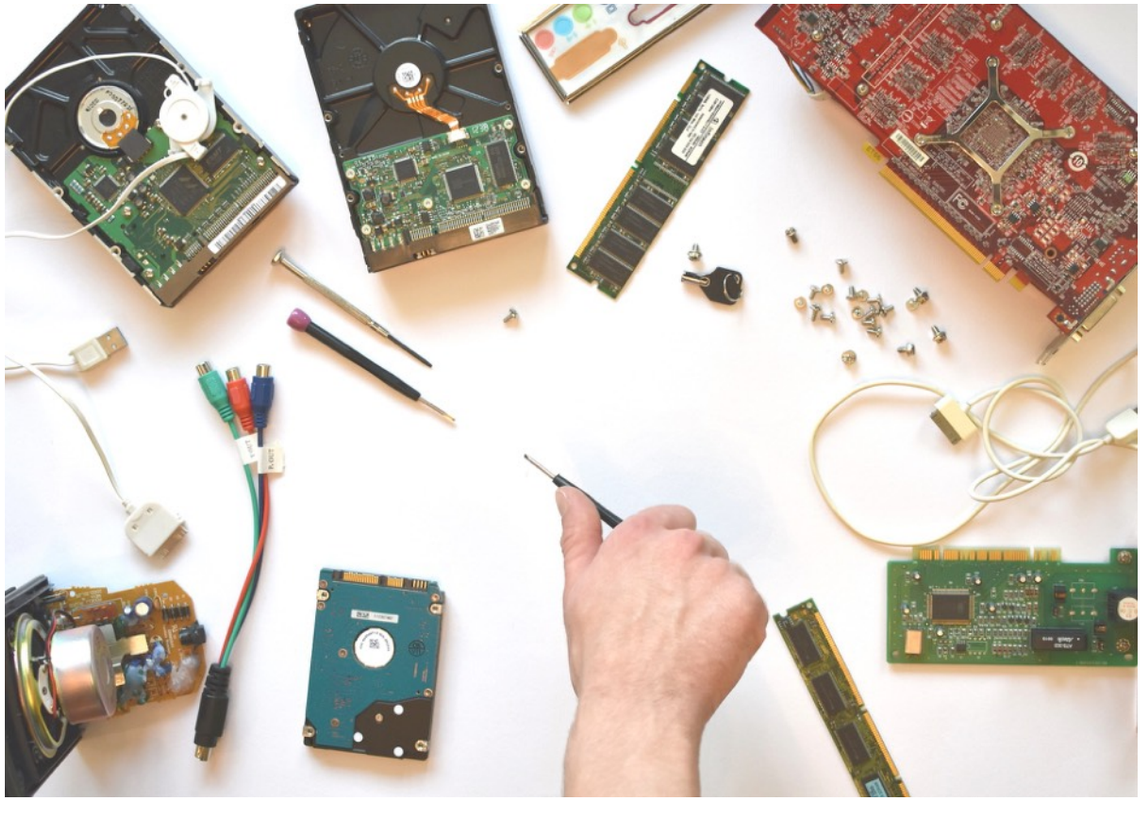
IBM-mainframecomputer

Niet alleen je **laptop** of **desktop** is een computer. Je **smartphone** is dat al evenzeer. Computers kom je zo wat overal tegen. Je tablet bijvoorbeeld. Misschien zit er wel eentje in je auto. Als je naar de **bankterminal** loopt, werk je op een computer die vermoedelijk via een netwerk verbonden is met een **mainframesysteem** van je bank.

Universiteiten gebruiken **supercomputers**, dat zijn toestellen met een enorme rekenkracht. Zulke toestellen nemen niet alleen heel veel ruimte in beslag, maar ze kosten bovendien gigantisch veel geld. Niet meteen iets om in je keuken te zetten.

Misschien heb je thuis wel een **gameconsole** zoals een Microsoft Xbox, een Sony Playstation of Nintendo Wii… ook dat is een computer. Zulke toestellen hebben vaak een zeer krachtige grafische processor het snel genereren van interactieve animaties mogelijk te maken.

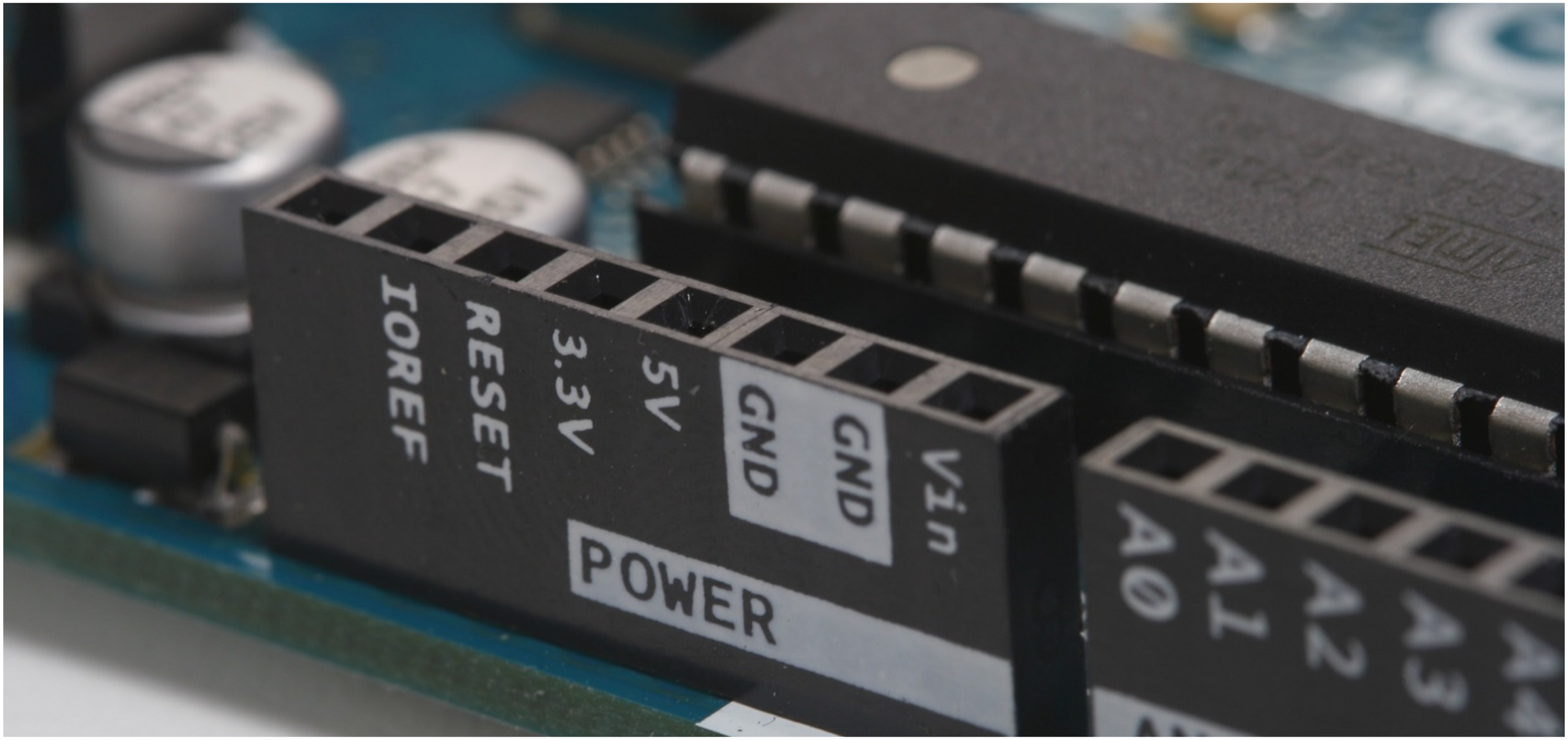
Misschien bevat ook jouw **televisie** wel een kleine computer. Je **DVD- of Bluray-speler** heeft zonder enige twijfel een klein computersysteem aan boord, want het moet de digitale informatie van de DVD- of BD-schijven verwerken tot visuele output op het televisiescherm.



Een computer (de **hardware**), of het nu gaat om een klassieke desktop, een tablet, smartphone of een laptop, bestaat uit een hoop onderdelen. Je vindt er een processor in, een tijdelijk werkgeheugen (RAM), een opslagmedium (harde schijf, SSD, SD...), invoerapparaten (muis, touchscreen, toetsenbord...), een outputsysteem (bestanden, scherm, webserver...). Al die diverse toestellen doen in wezen niet zo'n aardig verschillende dingen. Ze verwerken invoer en tonen en/of bewaren de uitvoer. Ze kunnen niet zonder "**software**", de ongrijpbare programma's die ergens virtueel zijn "geïnstalleerd" als digitale gegevens op een opslagmedium. De software geeft onder de vorm van "binaire" code instructies aan de toestellen, verwerkt de invoer en presenteert de uitvoer.

Software bestaat op diverse niveaus:

1. een basissysteem (bios, uefi) dat de contacten tussen alle onderdelen regelt,
2. een besturingssysteem (zoals Android, iOS, Linux, Mac OS X, UNIX, Windows...)
3. of gewoon software met een heel specifieke taak (zoals GIMP, MS Word of... Processing).



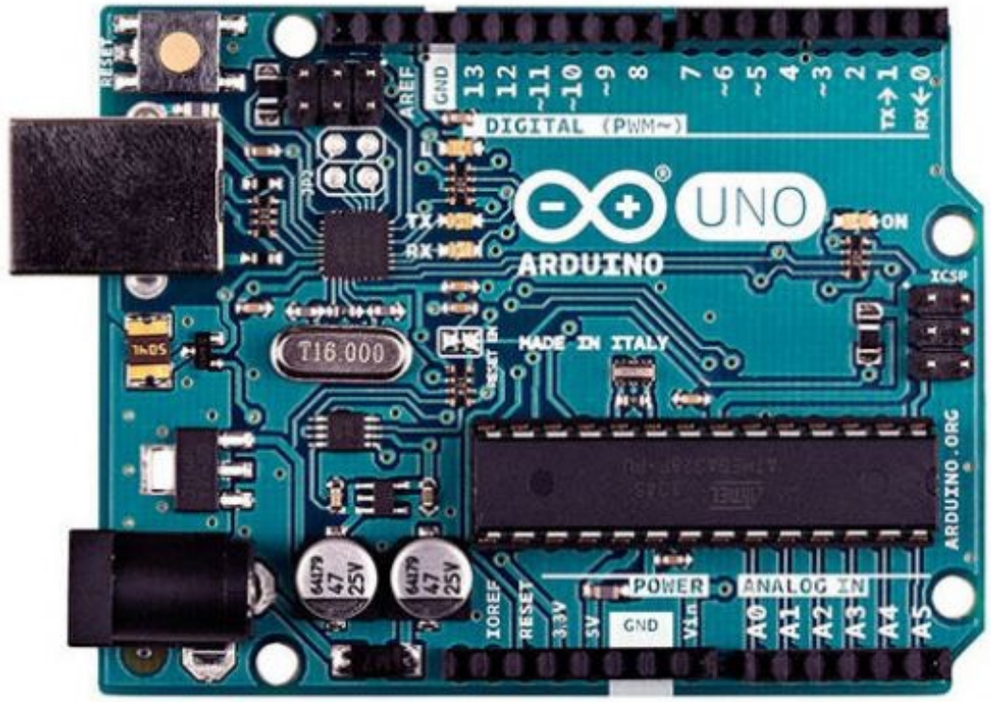
De Arduino is een open source ontwikkelplatform met microcontroller. Je kan er snel prototypes van digitale/elektronische toestellen mee bouwen.

Maar er bestaan ook andere verschijningsvormen die men gemakshalve eveneens aanduidt met de naam “computer”, maar in wezen niet helemaal voldoen aan de definitie hier boven. Een microcontroller is zo’n “computer”. Men gebruikt een **microcontroller** of **microprocessor** om elektronische apparatuur te besturen.

Heel wat moderne apparaten bevatten zo'n microcontroller: een magnetron, een auto, een wasmachine, sommige telefoons...

Waarom zou je met zulke toestellen aan de slag gaan of er tijd in steken, want in wezen zijn ze veel trager dan een standaard “computer” of zelfs een goedkope smartphone?

Embedded systemen en microcontrollers zijn aan een ware veroveringstocht bezig. Wellicht hoorde je de term “internet of things” al eens vallen. Ruwweg houdt dit in dat stilaan elk huishoudelijk toestel, maar ook auto's, bewakingscamera's en domoticasystemen zulke embedded systemen aan boord hebben. Wanneer je al die dingen (things) aansluit op het internet, krijg je een allesomvattend netwerk van digitaal verbonden toestellen: het **internet of things**.



Een Arduino-microcontroller.

We stellen ons hier even geen vragen over de gevolgen en gevaren voor de privacy en de ethiek.

Kortom, embedded systemen en microcontrollers zijn niet meer weg te denken uit de wereld van vandaag en morgen.

Embedded systemen bieden tal van **voordelen** in vergelijking met een “computer”.

1. De verwerking gebeurt real time (beeld je maar eens in dat je auto zou werken op Windows),
2. snel
3. en verbruikt zeer weinig energie.
4. Een embedded systeem is ook veel goedkoper.

Nadeel is dat de functionaliteit heel beperkt is. Je kan bijvoorbeeld geen tekstverwerker installeren in jekoelkast of videomontagesoftware in je auto.

Afbeelding:

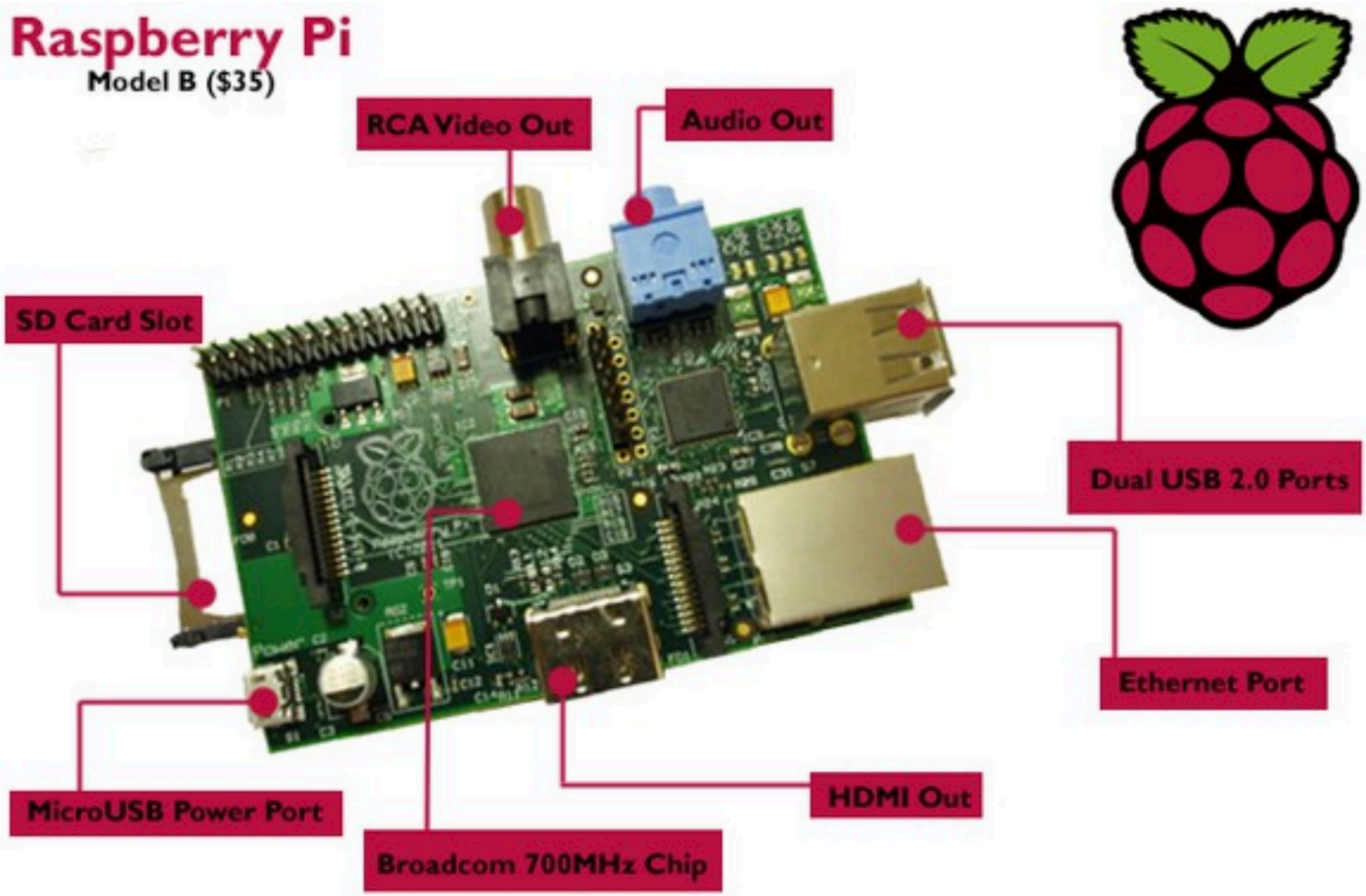
Het Leuvense IT-bedrijf EASICS ontwikkelde de beeldherkenningshardware waarmee de sorteermachines van de Belgische firma BEST zijn uitgerust. De hardware van EASICS herkent tegen een onwaarschijnlijk hoog tempo "ongewenste" elementen tussen bijvoorbeeld razendsnel voorbij rollende frieten, krenten, garnalen, spijkers of wat dan ook en activeert een luchtdrukstraal die de ongewenste element wegspuit. Een "multifunctionele" computer, hoe krachtig en snel ook, zou er niet in slagen om die taak zo snel af te handelen. Omdat de programmacode in de chip is gecodeerd, verloopt de verwerking hier onwaarschijnlijk snel.

Naast de term "microcontroller" en "embedded system" duikt ook wel eens de term "**system on a chip" (SoC)** op. Een SoC (met een C wel te begrijpen), integreert alle componenten van een computer en/of elektronisch systeem op één enkele chip.

SoC's worden veelvuldig toegepast in de markt van mobiele consumentenelektronica. Een "embedded systeem" bevat vaak een SoC.

Afbeelding:

Diverse SoC's in een smartphone.

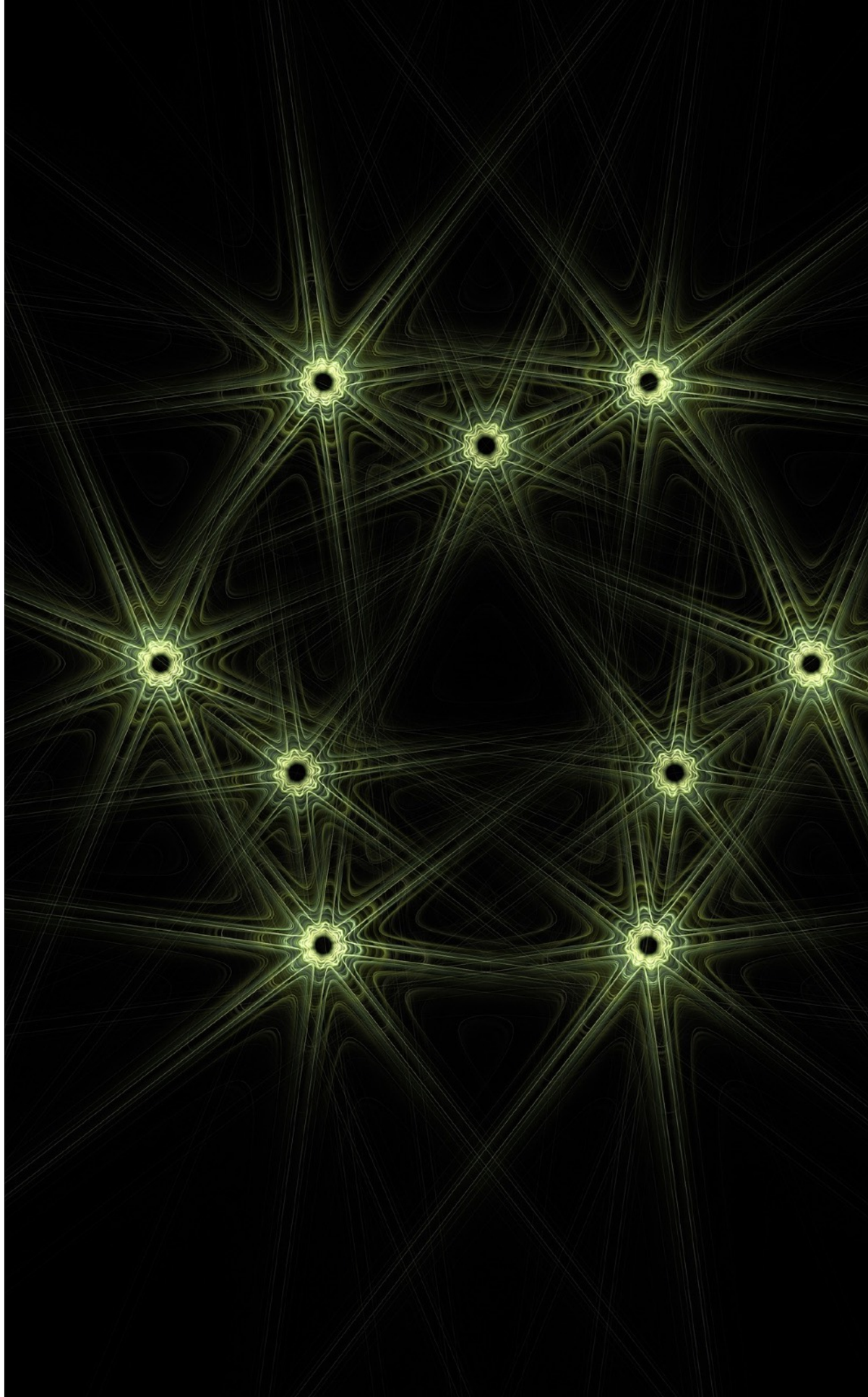


Een Raspberry Pi is een kleine computer waarop je ook een echt besturingssysteem als Linux of zelfs Windows kan installeren. Daarnaast kan je het eveneens gebruiken als ontwikkelbord.

QUANTUMCOMPUTERS & DNA-COMPUTERS

Kan een computer nog sneller eens de grenzen van Moores wet zijn bereikt? Op een bepaald moment kunnen transistors niet meer kleiner worden gebouwd. Er zijn grenzen aan de miniaturisering. Volgens sommigen zullen quantumcomputers de ultieme uitkomst bieden. Je hoort de meest waanzinnige verhalen over de rekenkracht van quantumcomputers. Berekeningen waar een conventionele supercomputer ongeveer de leeftijd van het heelal zou voor nodig hebben, kunnen met een quantumcomputer binnen een paar weken worden uitgevoerd.

Slik, dat is snel! Meer nog: waarom gebruiken we de klassieke transistorcomputers dan nog?



Quantumcomputers

Het antwoord is eenvoudig en ingewikkeld tegelijk. In de eerste plaats kan je **quantumcomputers** niet simpel in het lokale warenhuis kopen, ze zijn immers nog volop in ontwikkeling. Bovendien zullen ze de klassieke computers niet van de kaart vegen. Ze zijn onvoorstelbaar snel (eens men een werkend model gebouwd heeft), maar toch wel bijzonder traag voor conventionele computertaken zoals bijvoorbeeld tekstverwerking. Een quantumcomputer zou het ganse weersysteem van de aarde kunnen simuleren in een fractie van tijd, maar het moeilijk hebben met fotobewerking zoals bijvoorbeeld in Adobe Photoshop.

Wat zijn quantumcomputers dan wel en hoe werken ze? Waarom hebben ze zoveel meer rekenkracht? Het antwoord is niet eenvoudig. Ze maken gebruik van de allesbehalve intuïtieve wetten van de quantummechanica. Bij klassieke computers is een bit ofwel 0 ofwel 1. Quantumcomputers werken met **Qubits** die tegelijk 1 of 0 kunnen zijn. Dat is, (inderdaad ik hoor het u al denken) niet logisch, maar het kan wel.

Begrijpt u het niet helemaal? Niet ongewoon, want de quantumschaal van elementaire deeltjes waarmee quantumcomputing werkt, gedraagt zich niet zoals we van objecten uit onze dagdagelijkse werkelijkheid gewend zijn. Qubits zijn in feite ook niet 0 en 1 tegelijk, maar ze geven een zekere "kans". Als je ze meet, geven ze bijvoorbeeld de waarden 0.34 en 0.66 terug, waardes die schommelen tussen 0 en 1. Ze zijn dus niet meteen geschikt voor langdurige opslag van informatie, maar door het feit dat ze twee waarden retourneren i.p.v. één, groeit de rekenkracht exponentieel.

DNA-computers

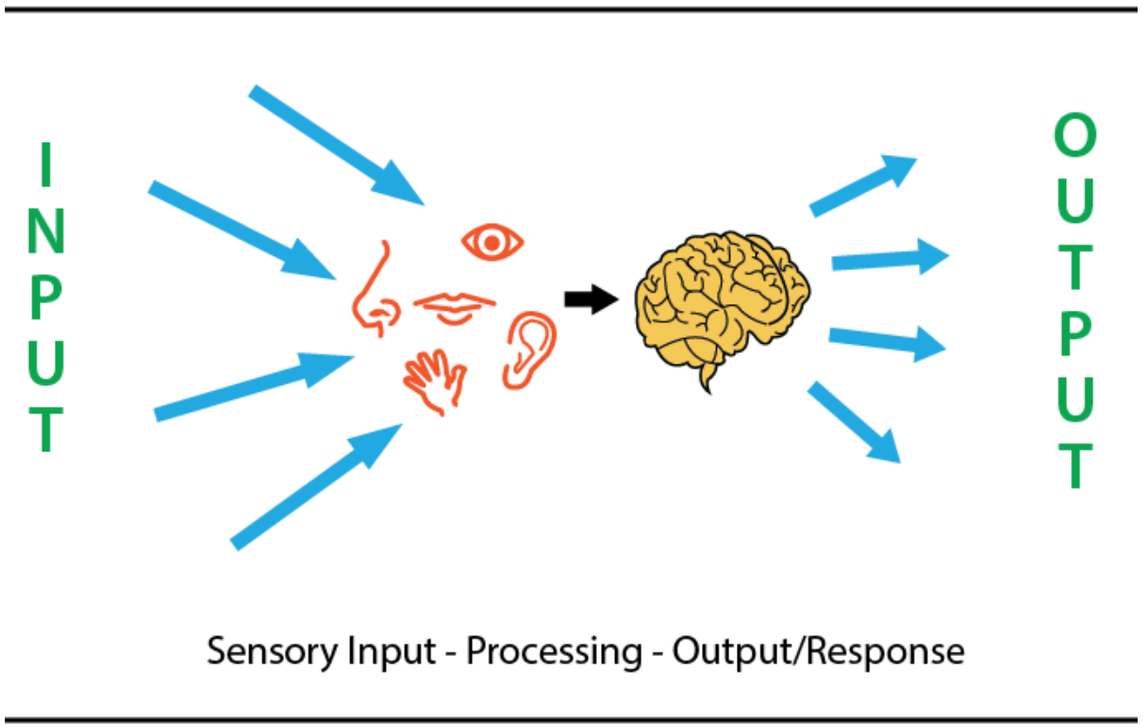
De huidige microprocessors zijn gebaseerd op tansistors/chips samengesteld uit silicium (silicon in het Engels). Zoals gezegd stoten we daar op grenzen. Ze kunnen niet nog kleiner gemaakt worden. Daarom zoeken wetenschappers naar nieuwe materialen. DNA-moleculen waaruit alle levende wezens zijn opgebouwd, zijn in feite ook kleine computers die onwaarschijnlijk snel berekeningen kunnen uitvoeren. **Biochips** gebouwd met DNA zullen in de toekomst mogelijk een onderdeel van jouw computer worden. De eerste proeven tonen aan dat **DNA-computers** miljarden keer meer informatie kunnen opslaan dan de huidige opslagmedia.

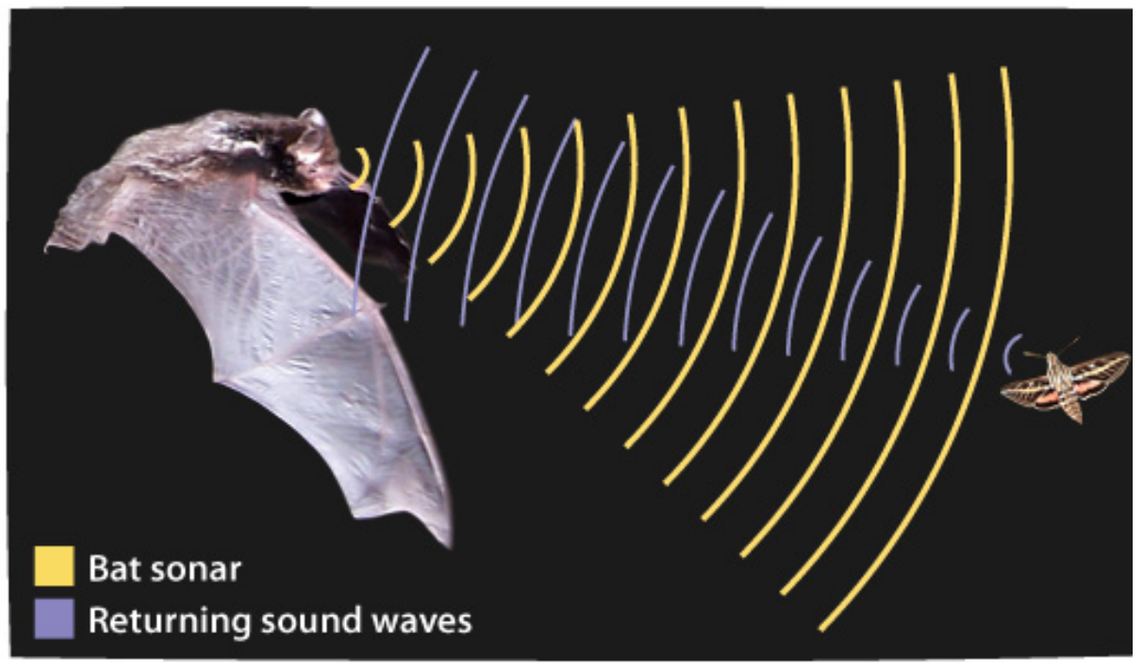
ANALOOG, ELEKTRONISCH OF DIGITAAL

De fysieke wereld waarin wij leven gedraagt zich **analoog**. Natuurkundige fenomenen zoals licht, geluid, temperatuur en zelfs het alcoholgehalte in je bloed... kunnen voortdurend wijzigen.

Een elektronisch apparaat zet signalen uit de omgeving (geluid, licht, warmte ...) om in elektrische signalen (spanningen, stromen ...). Om informatie uit de werkelijkheid 'waar te nemen', beschikt een elektronisch toestel over **sensoren**.

Bijvoorbeeld: een foto-elektrische cel zet licht om in een elektrisch signaal, een microfoon zet drukgolven in de lucht om in een veranderlijke elektrische spanning.





Biologische sensoren

Wij mensen beschikken over 5 zintuigen: zicht (ogen), gehoor (oren), smaakzin, geurzin, tastzin/voelzin… Daarnaast zijn we ook gevoelig voor temperatuur en bijvoorbeeld elektrische sensaties (bijvoorbeeld als je je vinger in het stopcontact steekt). Dieren en planten beschikken vaak ook over andere sensoren of hun zintuigen hebben een ander bereik.

Zo kunnen vleermuizen ultrasone geluiden waarnemen. Ultrageluid of ultrasoon geluid is geluid waarvan de frequentie te hoog is om gehoord te worden door het menselijk oor. Slangen kunnen infrarood licht waarnemen als grijstinten. Hierdoor kunnen ze warmbloedige dieren vinden in het duister. M.a.w. ook al heb je vijf zintuigen, toch kan je nog niet alles frequenties waarnemen.

Planten zijn in staat om licht, chemische stoffen, magnetische en mechanische (aanrakings) sensaties, temperatuur, elektriciteit en geluid waar te nemen.

Kortom, sensorische waarneming is niet uniek voor de mens. Bovendien zijn onze “sensoren” beperkt.

Kunstmatige sensoren

Zonder dat we het zelf goed en wel beseffen worden we omringd door sensoren. Je vindt ze in je mobiele telefoons (camera, microfoon, accelerometer ...), het toetsenbord en de muis van je computer, de thermostaat van de verwarming, in automatische lampen, het scherm van een tabletcomputer ... Als je na een nachtje stappen aan de kant wordt gezet door de politie, vind je ze zelfs in de alcoholtester (een chemische sensor).

Omdat de opgewekte elektrische signalen vaak te zwak zijn om bruikbaar te zijn, bevat veel elektronica ingebouwde of aangesloten **versterkers**. Analoog houdt in dat de informatie als een continue golf of 'stroom' wordt opgeslagen. *Zoals je weet, bestaat geluid in werkelijkheid uit een reeks voortdurende trillingen in de lucht, een golf met andere woorden. Bij een microfoon brengen de trillingen van de lucht een membraan (een soort vliesje) in beweging. Deze beweging wordt omgezet in een veranderlijk elektrisch signaal, dat opgeslagen kan worden op magneetbanden*. **Analoog betekent dus 'naar analogie met de werkelijkheid'.** Analoge signalen zijn erg onderhevig aan storingen of interferenties van bijvoorbeeld andere apparaten.

Digitaliseren

Elektronisch is nog wat anders dan digitaal, ook al hoor je de termen weleens door elkaar gebruiken.

Als een systeem een signaal (geluid, beeld, temperatuur...) digitaliseert, gaat het de elektrische signalen omzetten in een reeks getallen. Als je elk uur de temperatuur meet met een temperatuursensor, kan je elk uur het elektrisch signaal meten en bewaren als een cijfer (=een digit). Zo krijg je een reeks cijfers/digits die de wijzigende temperatuur doorheen de tijd opslaan.

Immers, een analoog signaal wijzigt voortdurend. De temperatuur bijvoorbeeld wijzigt doorheen de dag, maar normaal gezien niet om de seconde.

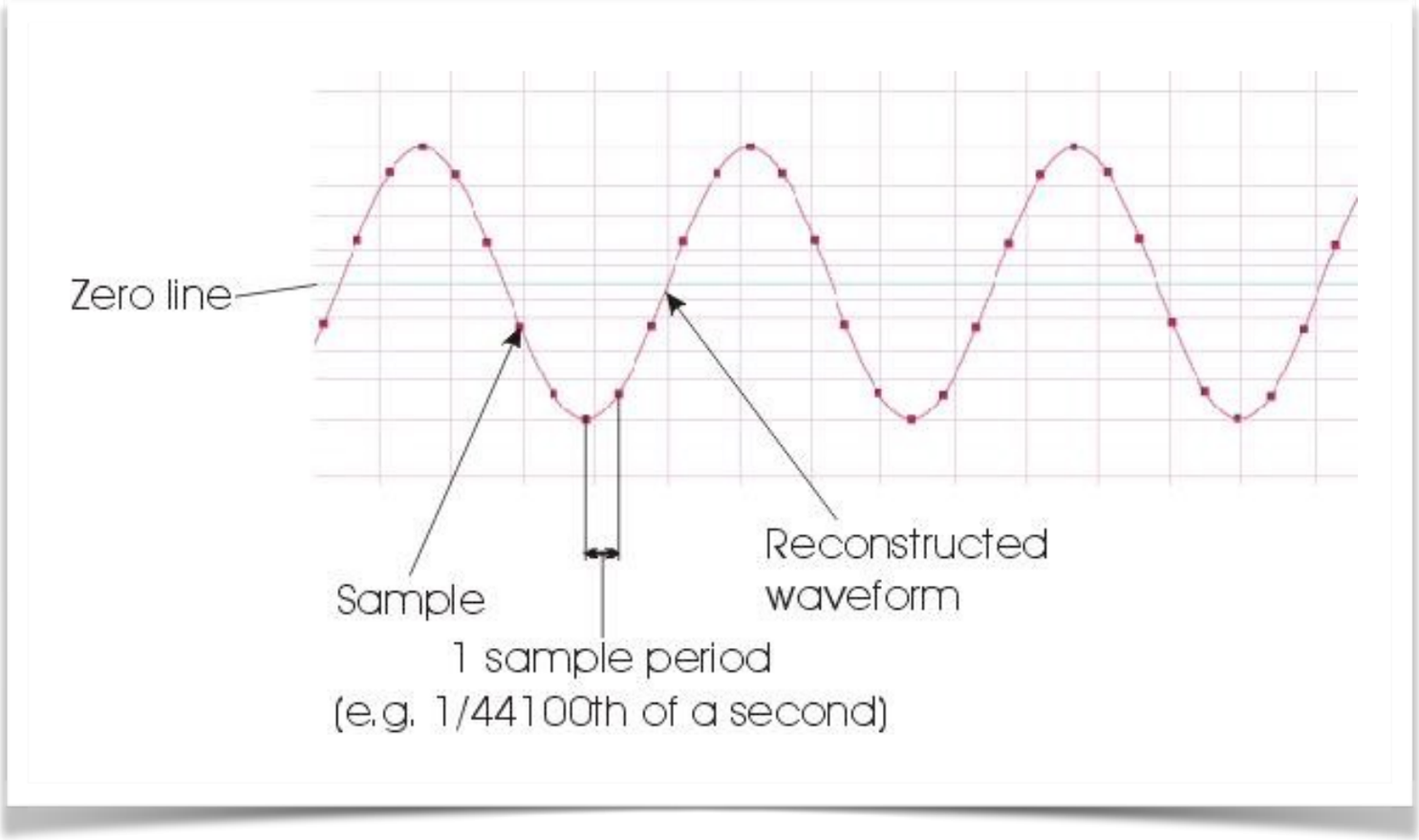
Bij geluid daarentegen moet je meer metingen doen. Om iemands stem op te nemen, volstaat het niet om elk uur even de geluidsgolf te meten. Dat doe je best meerdere keren per seconde, om zo nauwkeurig mogelijk de uitgesproken tekst te kunnen opnemen en eventueel weer af te spelen achteraf. Wil je geluid registreren met Audio-CD-kwaliteit, dan moet je het geluid 44 100 keer per seconde meten (44,100Hz).

Bij een beeldsensor van een digitale camera is de tijd van minder belang, maar moeten er meerdere signalen tegelijkertijd worden gemeten. Een foto of beeld is immers "rechthoekig". Een beeldsensor verdeelt een rechthoekig kader in kleine vakjes (rijen en kolommen) en meet bij het fotograferen van elk van deze vakjes de kleurwaarde. Elk vakje noemen we in dit geval een picture element (pixel).

Elke meting, of het nu om geluid of beeld gaat, noemen we een SAMPLE. In het Nederlands noemen we dit proces DISCRETISATIE.

Samengevat:

Een elektronisch toestel meet met behulp van sensoren analoge informatie uit de werkelijkheid. Vervolgens zet het die informatie om in een elektrisch signaal. Bij het digitaliseren wordt dit elektrisch signaal omgezet in cijfers (digits).



Een geluidsgolf bemonsteren (samplen): het geluidsniveau wordt 44100 keer per seconde gemeten. Elke meting krijgt een getalwaarde boven of onder nul.

Afspraken maken

Bij het bouwen van zo'n systeem moeten er vooraf afspraken gemaakt worden. Bij een afbeelding kan dit gaan om de hoeveelheid pixels die men horizontaal of verticaal wil meten. Bij geluid over het aantal metingen per seconde. Een audio-cd bevat bijvoorbeeld 44100 samples of metingen per seconde (sample rate van 44.100 Hertz = 44,1 kHz).

Elke sample of meting wordt uitgedrukt als een getal tussen 0 en een vooraf bepaalde maximale waarde. Hoe hoger die waarde, hoe groter het bereik en hoe nauwkeuriger het signaal kan worden gemeten en weergegeven. Wanneer we licht meten met slechts twee lichtintensiteitswaarden, bevat ons eindresultaat enkel wit of zwart.

Binair coderen

De meeste moderne computers zijn digitaal. Ze zetten analoge informatie om in reeksen van getallen, zoals we hierboven reeds uitgelegd hebben.

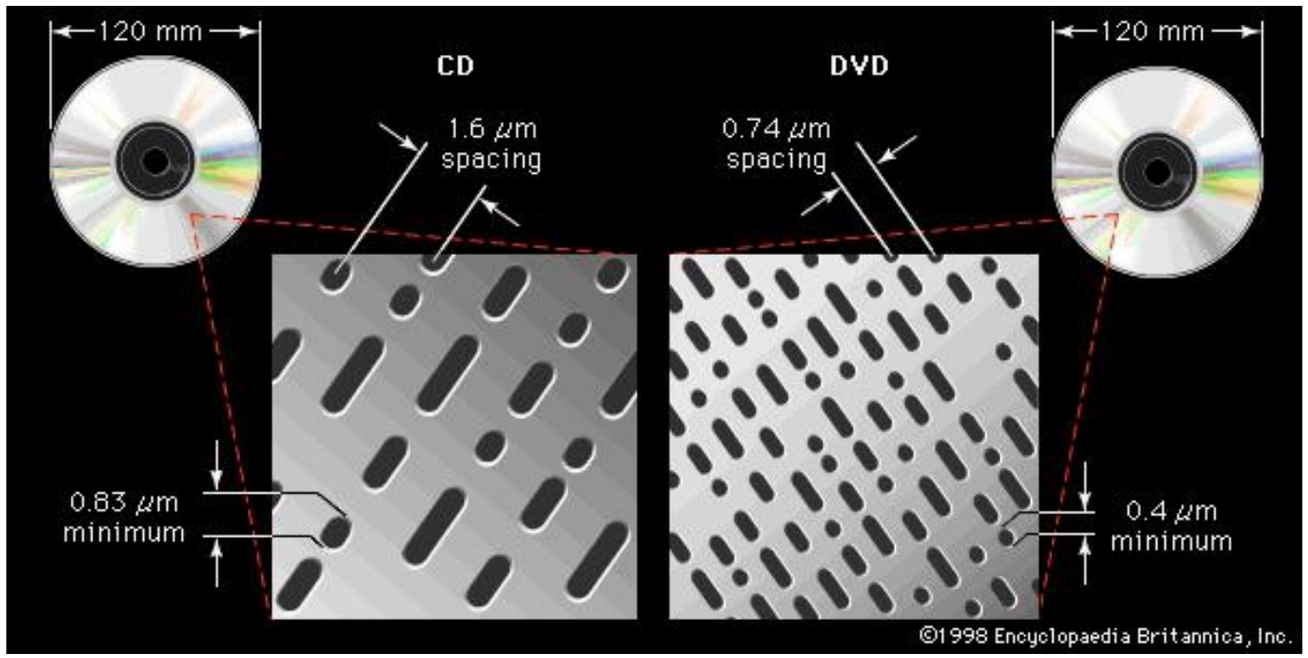
Over het algemeen gebruiken computers hiervoor het **binaire talstelsel**. Dit betekent eenvoudigweg dat ze slechts twee karakters gebruiken om informatie op te slaan. In het decimale talstelsel gebruiken we hiervoor 10 verschillende karakters nl. 0,1,2,3,4,5,6,7,8 en 9. Het binaire talstelsel werkt enkel met 0 en 1. Dit lijkt handiger omdat een computer in wezen niet meer is dan een enorme schakelkast met miljoenen of miljarden schakelaars die aan of uit kunnen staan.

Dit betekent dus dat elk getal (elke gemeten waarde) binair wordt opgeslagen. Een binair cijfer of bit (binary digit) kan dus slechts twee vormen aannemen: een 0 of een 1. Een binair getal noemen we anders naargelang het bereik. Zo noemen we een getal van 8 bits een byte. Wij leren op school letterlijk tellen op onze vingers (digitus in het Latijn) en we gebruiken daarom niet voor niets een tiendelig of decimaal talstelsel. Daarom is het ook even wennen als we met binaire getallen aan het tellen gaan:

In het hoofdstuk "De taal van de computer" leggen we uit hoe eenvoudig het binaire talstelsel eigenlijk wel is. Je leert ook werken met het hexadecimale talstelsel dat in de computerwereld eveneens veelvuldig gebruikt wordt.

0 = 0, 1 = 1, 2 = 10, 3 = 11, 4 = 100, 5 = 101, 6 = 110, 7 = 111, 8 = 1000 ...

Een bereik van 8 bits geeft 2 tot macht 8 (256) mogelijke waarden. Een audio-cd heeft een bereik van 16 bits of 32.768 mogelijke waarden. Deze vorm van discretisatie beïnvloedt in sterke mate de kwaliteit. Hoe lager het aantal beschikbare bits of hoe trager het bemonsteren gebeurt, hoe lager de kwaliteit. Een te hoog bereik is echter evenmin zinvol. Het menselijk oor kan de extra kwaliteit toch niet altijd waarnemen. Daarom probeert men vaak een gulden middenweg te vinden tussen kwaliteit en 'beperkingen', zoals opslagcapaciteit en doorvoersnelheid (bijvoorbeeld de snelheid van de internetverbinding). In onze tijd worden digitale gegevens ook opgeslagen en voor langere tijd bewaard.



Bewaren van binaire gegevens

Hoe gaat dit fysisch precies in zijn werk en welke voordelen biedt deze manier van opslaan?

In een digitaal systeem bestaat elke waarde uit een reeks bitjes, die elk afzonderlijk slechts 2 geldige waarden kennen, namelijk 0 of 1. In een elektronische schakeling kunnen we dit zien als een verschil tussen bijvoorbeeld 0 en 1,8 Volt, op een cd als een minuscuul putje of net geen, op een harde schijf als een klein gebiedje dat al dan niet gemagnetiseerd is, op een flash-geheugenkaartje als een pakketje elektronen (elektrische lading) dat wordt vastgehouden in een transistor ...

Samengevat betekent 'digitaal' dus dat we de informatie bewaren als een reeks binaire getallen. Een geluidsgolf kunnen we bijvoorbeeld weergeven als een reeks getallen met een bepaalde grootte. Als we die getallen op een grafiek zetten, zien we de oorspronkelijke golf weer verschijnen. Digitale informatie wordt bijna altijd met binaire en niet met decimale getallen weergegeven. Binaire getallen zijn uitermate geschikt om door elektrische schakelingen of bedrading te sturen omdat die enkel maar een aan-toestand (1) en uit-toestand (0) herkennen. In een analoog systeem kan elke gewenste waarde tussen een bepaalde ondergrens en bovengrens worden voorgesteld.

Voordelen van digitalisering: foutonderdrukking en bewerkingsmogelijkheden

Als er ruis optreedt door bijvoorbeeld slijtage op een magneetband of een elektromagnetische storing, kan het oorspronkelijk signaal niet meer onderscheiden worden van de ruis. De ruis die er achteraf bijgekomen is, heeft de oorspronkelijke signaalniveaus immers aangetast.

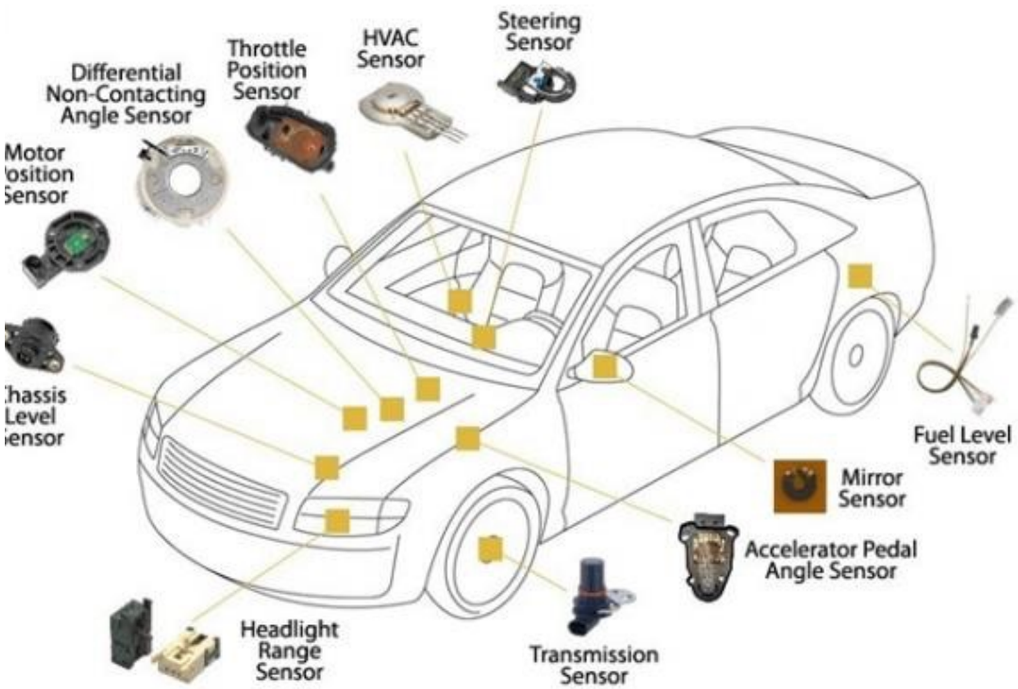
Ook in een digitaal systeem kan ruis optreden. Een kras in een optische schijf zoals een cd kan putjes minder diep maken of elektromagnetische storingen kunnen bepaalde zones op een harde schijf demagnetiseren. Toch kan over het algemeen het oorspronkelijke niveau hersteld worden (als de storing niet te groot is) omdat elke bit slechts twee geldige waarden heeft: *het is ofwel 1 ofwel 0 en daaruit moet het systeem zelf kiezen*.

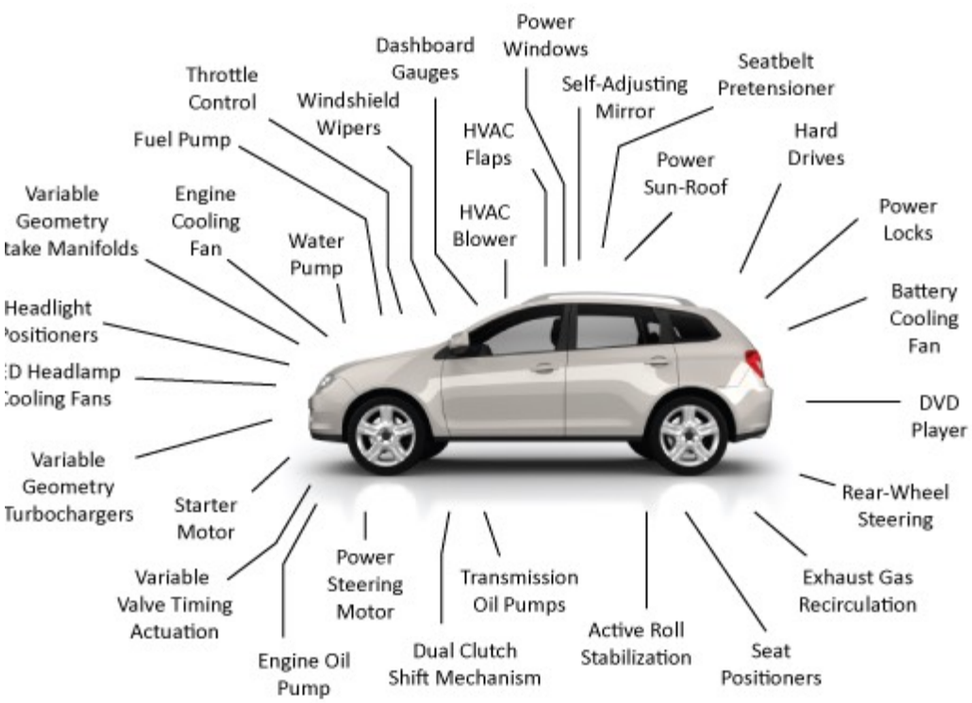
Een kras kan bij wijze van spreken van 0 een 0,3 maken, maar die waarde ligt dichter bij de 0 dan bij de 1. Enkel een 0,5 zou een twijfelgeval kunnen worden. Als de schade door slijtage, ruis of storingen dus niet te groot is, kan een digitaal systeem de oorspronkelijke informatie of kwaliteit volledig herstellen. Maar zelfs in het geval van grote schade kan men door **codeertechnieken** (het uitrekenen van bepaalde controlegetallen over groepen van bits) de fouten toch nog herstellen (uiteraard binnen bepaalde grenzen).

Sommige systemen maken gebruik van symbolen die meer dan twee waarden kunnen aannemen, waarbij één geheugencel 2 of 3 bit aan informatie kan bevatten. Dit soort transistors houden pakketjes elektronen van 4 of 8 verschillende (nominale) groottes vast. Zulke systemen zullen echter sneller falen bij kleine storingen of ruis. Toepassingen die een hoge betrouwbaarheid vereisen (zoals medische implantaten of ruimtevaart) zullen daarom 1 bit-cellen gebruiken.

Digitalisering biedt nog tal van andere voordelen. Zodra een signaal is gedigitaliseerd, kan het makkelijk bewerkt en aangepast worden.

Voor het toepassen van een **filter** op een afbeelding hebben we bijvoorbeeld niet langer een speciaal onderdeel of toestel nodig, dit kan met een computerprogramma. Dezelfde geheugendragers kunnen gebruikt worden om verschillende vormen van informatie te bewaren: foto's, tekst, muziek, beeld, programma's .





Sensoren en actuatoren in een auto.

Actuatoren: tijd voor actie

Actuatoren zijn het tegengestelde van sensoren. Actuatoren vormen een ander onderdeel van een elektronisch systeem: zij zetten elektrische signalen om in andere signalen zoals mechanische kracht, geluid ... Een luidspreker is een bekend voorbeeld van een actuator. Hij laat de lucht trillen door eerst zelf te trillen. Zo zet een elektrische motor elektriciteit om in een mechanische kracht.

Wikipedia omschrijft het op de volgende manier:

Een volledig systeem om iets te controleren en te reageren op deze waarnemingen bestaat uit de volgende drie onderdelen:

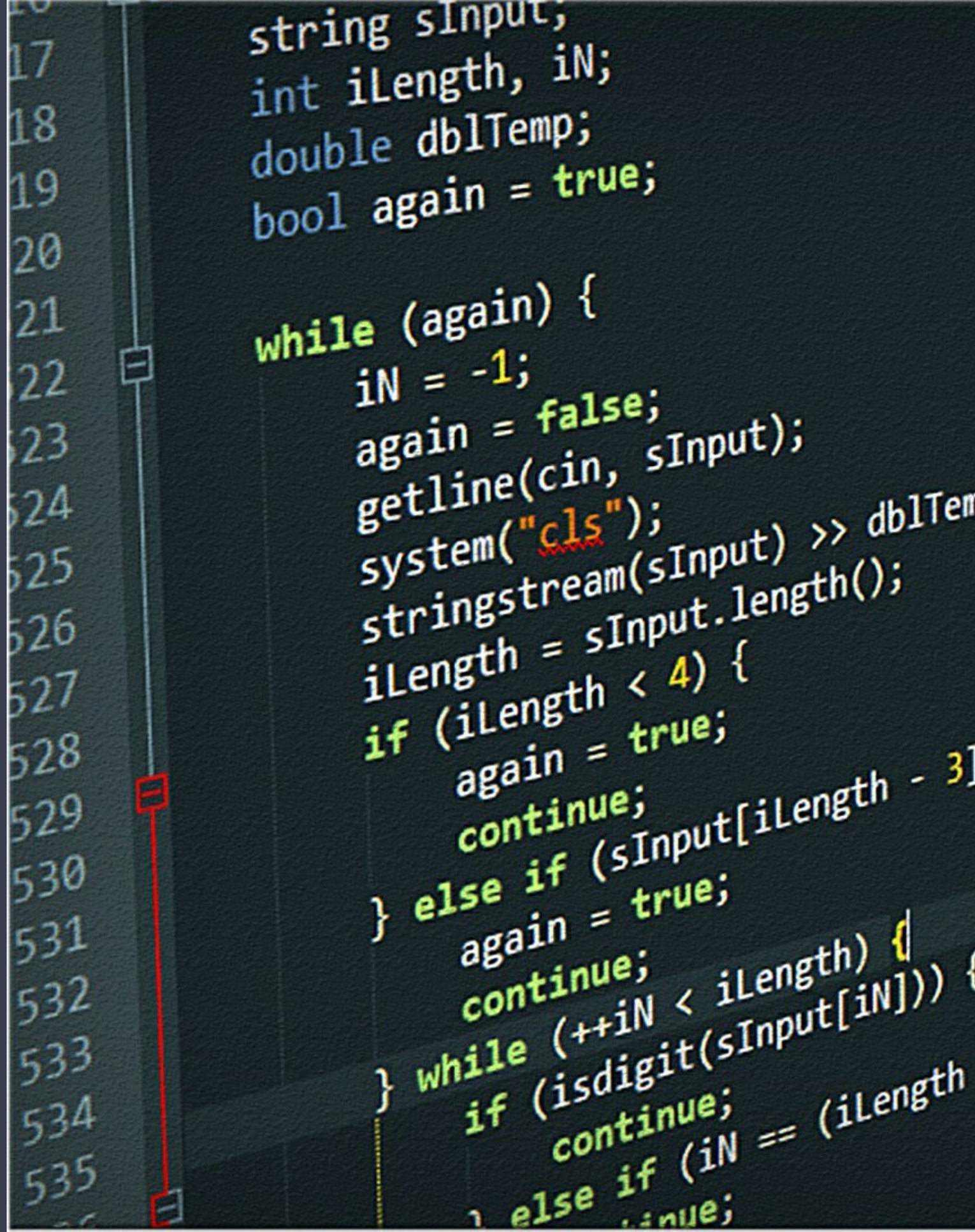
1. Sensor:
   dit is de ingang van het systeem, er wordt hier door middel van een mechanisme een waarneming gedaan. Deze waarneming kan bijvoorbeeld de omgevingstemperatuur zijn. De uitgang van dit onderdeel kan een analoog of een digitaal signaal zijn.

2. Regelaar:
   veelal wordt dit gedaan door microprocessoren of digitale signaalprocessoren, die met digitale waarden werken.

3. Actuator:
   hier wordt de uiteindelijke beslissing analoog of digitaal uitgevoerd.

DEEL 2 SOFTWARE

- Hoe werken programmeertalen?
- Wat zijn ‘compilers’?
- …

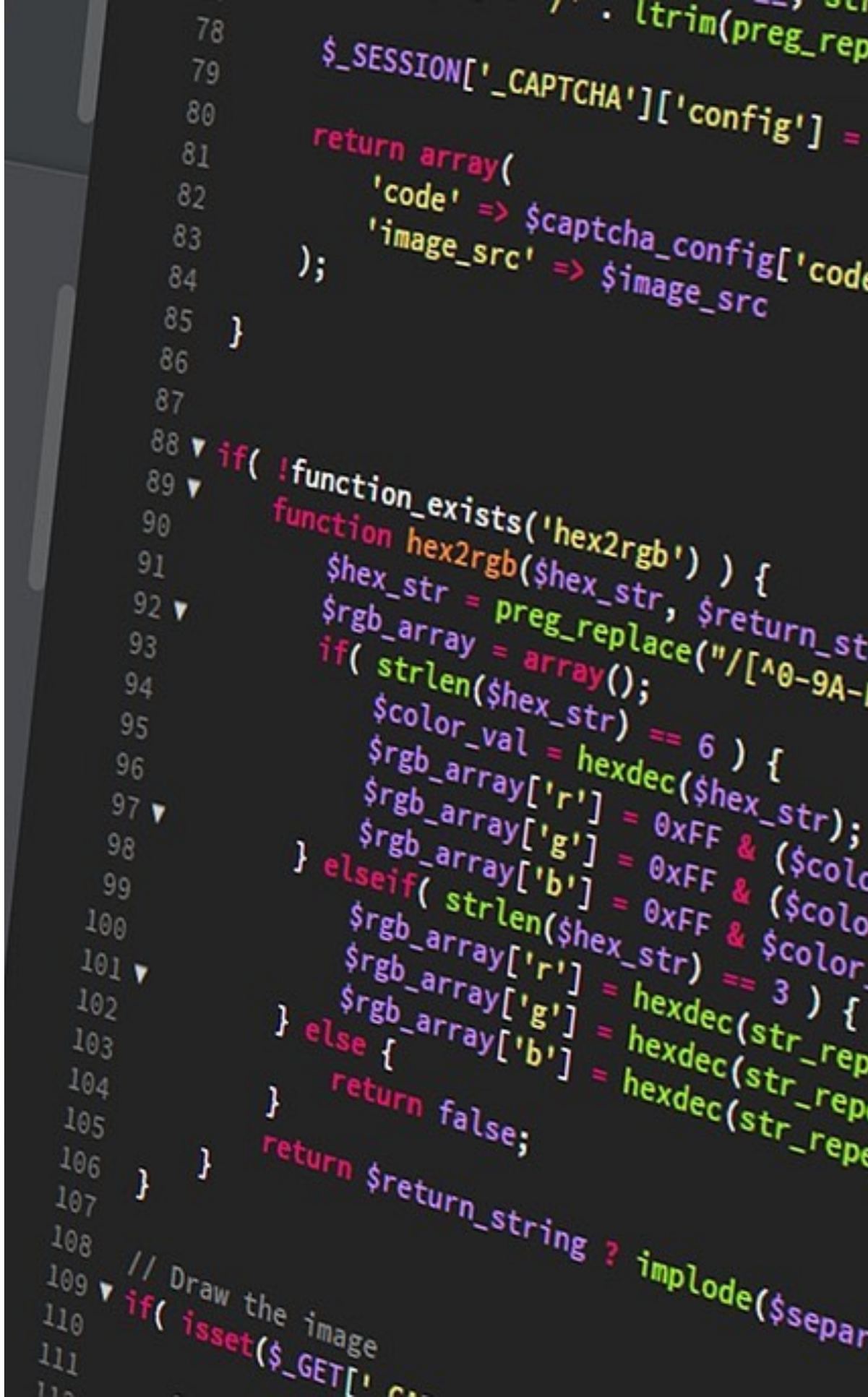


DE TAAL VAN DE COMPUTER

Als je leert hoe je iets moet doen, dan krijg je als mens doorgaans een stappenplan van instructies voorgeschoteld. Eerst moet je dit doen, daarna dat... Vergelijk het met een kok die een recept nauwgezet moet volgen.

Ook een computer volgt een reeks instructies om diverse taken te kunnen uitvoeren. Die reeksen instructies kunnen vastliggen in de hardware. Maar in veel gevallen kan je ze ook toevoegen aan een computersysteem in de vorm van software.

Software wordt gemaakt door menselijke programmeurs. Ze stellen een lijst van instructies op die een computer moet uitvoeren om een bepaald probleem op te lossen. Software helpt ons op die manier om teksten vlotter te kunnen schrijven en aanpassen (tekstverwerker), om berekeningen te doen (rekenprogramma), foto's te bewerken… *Maar in welke taal liggen die instructies vast?*



Computers spreken getallen, maar niet (altijd) decimaal

Zoals we eerder al hebben gezien, vertalen elektronische systemen zoals computers analoge informatie naar reeksen getallen. Omdat de basis van elektronische systemen “elektrische schakelingen” zijn, kennen ze in feite maar twee toestanden nl AAN of UIT.

Dit betekent dat computers best niet werken met decimale getallen zoals wij mensen doen. Ze noteren elk getal in binaire vorm.

Vermoedelijk heb je op school ooit geleerd over binaire getallen. De kans is groot dat je dit niet zo leuk vond. Nochtans werkt het vrij eenvoudig.

Ook al is wiskunde niet meteen je ding of baal je van cijfers, toch kan je allemaal in zekere zin wel met getallen en cijfers overweg. Doorgaans rekenen we in het tiendelig talstelsel en dat is niet ongewoon. Het heeft te maken met het feit dat we tien vingers en tien tenen hebben. In het decimaal stelsel gebruik je 10 verschillende “tekens” om getallen mee uit te drukken: 0, 1, 2, 3, 4, 5, 6, 7, 8 en 9.

Al onze eenheden zijn machten van 10. Een "eeuw" telt bijvoorbeeld 100 jaar of meer bepaald 10 tot de macht 2 (10*10=100). 1000 komt overeen met 10 tot de derde macht of 10*10*10.

Bij getallen bepaalt de plaats van het cijfer de macht die je er aan toewijst. Het getal 92, is op de onderstaande manier samengesteld/geteld:

9	2
9 maal 10 tot de eerste macht	2 maal 10 tot de 0de macht.
9*10	2 * 1

Getallen lezen we op die manier van rechts naar links. Helemaal rechts zien we als positie 0 (de 0de macht = altijd gelijk aan 1). Elke stap die we naar LINKS zetten, verhoogt de macht met 1:

...	3	2	1	0
...	Macht 3	Macht 2	Macht 1	Macht 0

Binair en hexadecimaal rekenen

In het binaire talstelsel gebruiken we maar 2 tekens om daarmee alle getallen te schrijven nl. 0 en 1. In het hexadecimale (of zestiendelige) talstelsel, dat eveneens vaak gebruikt wordt in de computerwereld, gebruiken we 16 tekens om getallen mee weer te geven. Omdat er maar 10 verschillende karakters bestaan voor het weergeven van cijfers, voegt men hier een aantal letters uit het alfabet toe:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F…

Waarom zou je dit gebruiken? Wel, het laat toe om lange getallen op die manier korter te noteren. Het getal 255 kan je op die manier noteren als FF. En zoals je ziet: een decimale 10 noteer je hexadecimaal gewoon als een A.

Vooral voor het noteren van kleurwaardes van pixels gebruikt men zeer vaak hexadecimale code. Immers elke pixel kent een kleurwaarde voor **rood, groen en blauw**. Die waardes noteert men eenvoudig naast elkaar. Je zou kunnen schrijven rgb(255, 255, 255) om een pixel wit te maken. Maar hexadecimaal noteer je het als volgt #FFFFFF.

Positie van cijfer in getal (van rechts naar links)	…	3	2	1	0
macht of exponent "n"	…	Macht 3	Macht 2	Macht 1	Macht 0
factor decimaal	…	10^3	10^2	10^1	10^0
factor binair	…	2^3	2^2	2^1	2^0
factor hexadecimaal	…	16^3	16^2	16^1	16^0

Het getal 92 noteer je binair als: 01011100 Hoe komen we daar aan? We beginnen aan de rechterkant van het getal:

Binaire getal	0	1	0	1	1	1	0	0
macht of exponent "n"	Macht 7	Macht 6	Macht 5	Macht 4	Macht 3	Macht 2	Macht 1	Macht 0
factor binair	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Omrekening	$0 * 2^7$	$1 * 2^6$	$0 * 2^5$	$1 * 2^4$	$1 * 2^3$	$1 * 2^2$	$0 * 2^1$	$0 * 2^0$
Decimaal	64	0	0	16	8	4	0	0

Nu maak je de som van alle bekomen resultaten: 64 + 16 + 8 + 4 = 92

Het getal 92 noteer je hexadecimaal als: 5C (C = decimale 12)

Hexadecimale getal	5	C
macht of exponent "n"	Macht 1	Macht 0
Factor hexadecimaal	16^1	16^0
Omrekening	$5 * 16^1$	$C * 16^0$
Decimaal	80	12 * 1
Resultaat omrekening naar decimaal	80	12

Nu maak je de som van alle bekomen resultaten: 80 + 12 = 92

Programmeertalen

Omdat het voor een mens behoorlijk moeilijk is om pakweg tien miljoen schakelaars te bedienen enkel door een reeks nullen en enen in te tikken, zijn er oplossingen bedacht in de vorm van programmeertalen.

E e n programmeertaal bevat wel woorden uit menselijke talen (meestal uit het Engels), waardoor ze bevattelijk wordt. Meestal gaat het om korte instructies, mnemotechnische hulpmiddelen, symbolen, operatoren of logische schakelingen zoals *and, or, if, else, end, print* ... Omdat die taal niet uit binaire code (nullen en enen) bestaat, is ze voor computers niet begrijpelijk.

Compilers

Een compiler is een speciaal stukje software (weer een programma) dat de in een programmeertaal geschreven instructies omzet in binaire code of machinetaal. Zo blijft de code voor de programmeur leesbaar en kunnen de instructies nadien ook weer aangepast worden. Na elke aanpassing aan de code moet het programma weliswaar opnieuw worden gecompileerd.

Omdat het programmeren in dit soort assembleertaal toch nog heel wat inzicht vroeg, zijn later de zogenaamde 'hogere programmeertalen' en 'scriptingtalen' ontwikkeld. Ze staan een stuk dichter bij de menselijke taal en zijn hierdoor eenvoudiger te leren.

Heel veel talen en systemen

In de loop der jaren (en nog steeds) zijn er honderden verschillende programmeertalen bedacht, vaak met een specifiek doel voor ogen. Elke programmeertaal moet echter op één of andere manier "verwerkt" worden vooraleer het een bruikbaar programma is.

Er bestaan bijzonder veel programmeertalen, elk met hun eigen compiler of interpreter (dit leggen we nog uit). Programmeurs die software schrijven die heel dicht bij de hardware zit, zoals besturingssystemen, gebruiken al een eeuwigheid de taal C of C++. Mac OS X, Linux en Windows zijn geschreven in C. Nadeel van zo'n talen is de code voor de ene hardware niet zo maar zonder aanpassingen op een ander hardwaresysteem werkt. Daarom kan je niet zo maar Android installeren op een iPhone, om een voorbeeld te geven. Adobe Photoshop voor Mac, kan je niet installeren op een Windowscomputer. Voor Windows heb je een aangepaste versie nodig.

Het uitvoeren van programmacode

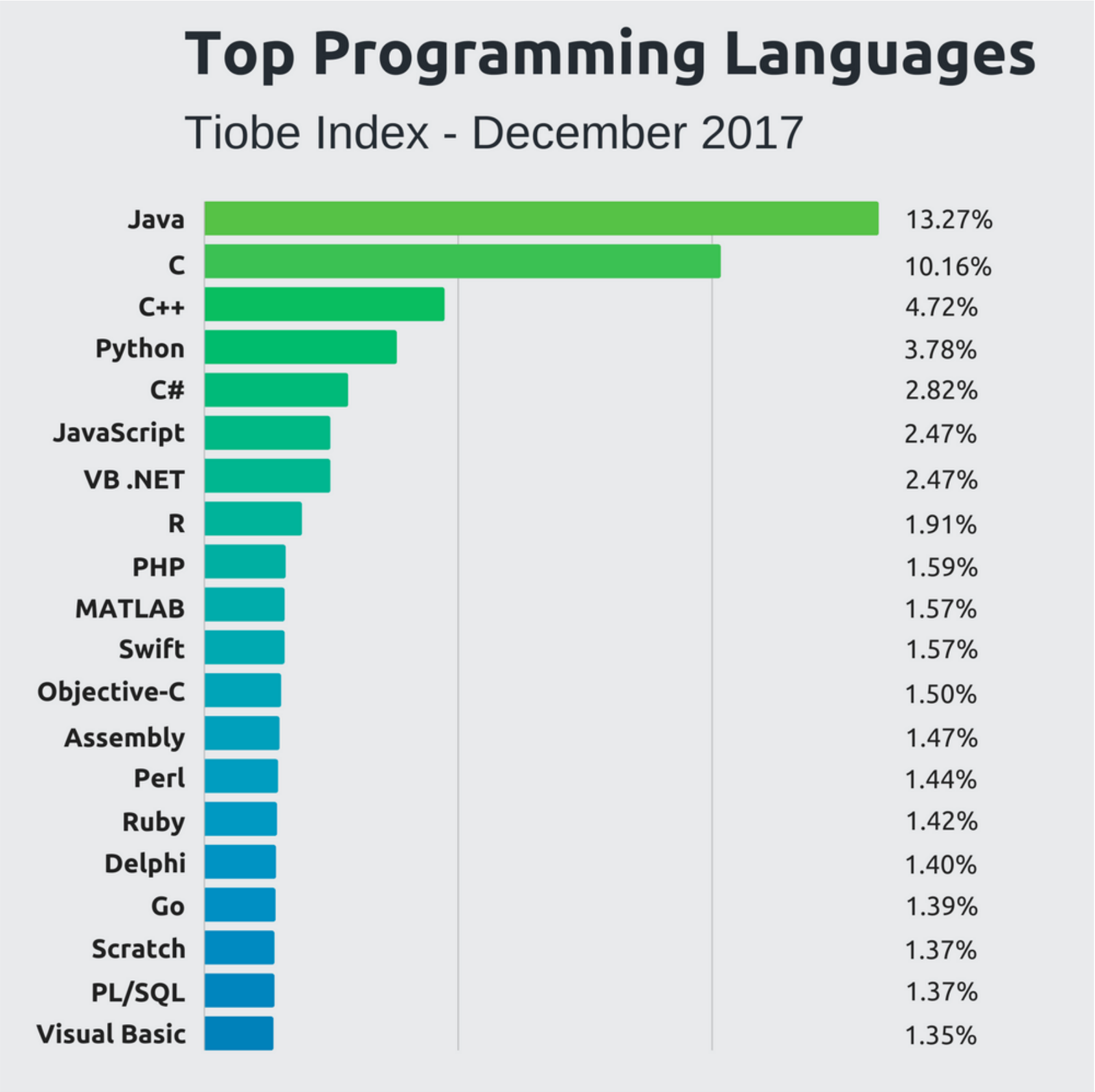
De manier waarop computers programmeercode verwerken kan verschillen:

1. **COMPILERS**
   Zoals gezegd moet programmeercode gecompileerd worden naar nullen en enen die de hardware begrijpt. Dit betekent echter (zie ook vorige pagina) dat code en ook gecompileerde code… voor het ene systeem niet zo maar werkt op het andere systeem. Een programmeur moet zijn code telkens aanpassen en opnieuw compileren als hij zijn code wil beschikbaar maken voor andere hardwareplatformen.

2. **INTERPRETERS**
   Soms is de code heel erg beperkt. In webpagina's bijvoorbeeld kan een webontwikkelaar eveneens code opnemen in de taal "javascript". Meestal is het aantal regels code hier beperkt. Hij heeft die code niet te compileren. Bovendien blijven zijn programma-instructies beperkt tot het browservenster. In dit geval zal de browser de code "interpreteren" en zelf "compileren" op het moment dat iemand de webpagina bezoekt. Javascript is de taal die in webpagina's wordt gebruikt.

3. **BYTECODE & CONTAINERS**
   Dankzij een programmeertaal zoals JAVA moet de programmeur maar één keer code schrijven die voor alle hardwareplatformen werkt. De gebruiker moet op zijn systeem (Mac OS, Linux, Windows…) vooraf een soort container installeren die ervoor zorgt dat de code bij gebruik meteen vertaald wordt naar instructies voor het platform waarop ze uitgevoerd wordt. Zo'n container heet vaak een "runtime environment". De programmeur dient zijn code vooraf niet te compileren voor elk afzonderlijk platform, maar kan ze geautomatiseerd "vertalen" naar een soort tussenlagen code (bytecode) die de container begrijpt.

4. **SOFTWARECONTAINERS**
   Tegenwoordig lost men de verschillen tussen diverse platformen op nog andere manieren op. Programmeurs verpakken een softwarepakket in een soort "doos" waarin ze eveneens alle specifieke platformgebonden elementen verpakken. Hierdoor kunnen ze software, soms zelfs volledige besturingssystemen (Linux, Windows…), in een soort "fake systeem" opstarten.


Top Programming Languages
Tiobe Index - December 2017
Java 13.27%
C 10.16%
C++ 4.72%
Python 3.78%
C# 2.82%
JavaScript 2.47%
VB .NET 2.47%
R 1.91%
PHP 1.59%
MATLAB 1.57%
Swift 1.57%
Objective-C 1.50%
Assembly 1.47%
Perl 1.44%
Ruby 1.42%
Delphi 1.40%
Go 1.39%
Scratch 1.37%
PL/SQL 1.37%
Visual Basic 1.35%

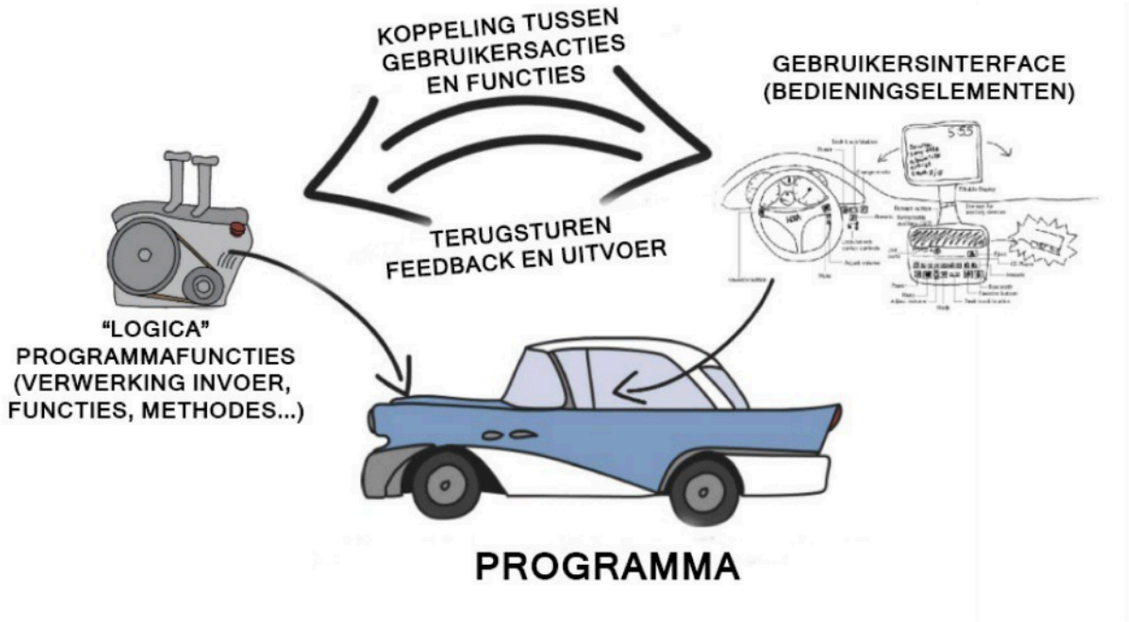
Programmeerconcepten

Vergelijk een programma of applicatie met een auto. De buitenkant van de auto en vooral het dashboard, stuur, pedalen, handrem, versnellingspook vormen de **interface**, waarmee de chauffeur de auto bestuurt.

Onder de motorkap zit de motor die reageert op de acties van de chauffeur. Een programmeertaal zelf is nog geen motor en al zeker geen auto. Met een programmeertaal beschik je wel over een manier en de onderdelen om een auto met motor te bouwen. De programmeur is tegelijk de ingenieur en de monteur die de motor ontwikkelt en bouwt. Niet alle programmeurs zijn daarin even bedreven. Daarom heb je soms een uitzonderlijk goed programma op het vlak van **functionaliteit**, maar een vreselijke bedieningsinterface. Ook het omgekeerde kan het geval zijn. Uiteraard werken programmeurs niet altijd in hun eentje.

Een programma komt pas tot leven als het reageert op de gebruiker. Een programma zonder **gebruikersinteractie** lijkt op een auto zonder stuur. Je kan een programma dingen laten doen met of zonder interactie met de gebruiker. Maar natuurlijk wordt het pas echt leuk wanneer de gebruiker merkt dat het programma op zijn acties reageert. Je zou als gebruiker immers al snel denken dat er iets fout loopt als het programma zo maar wat dingen doet (of niet doet).

De analogie met een auto zal het hopelijk allemaal wat duidelijker maken:



Programmeren is (geen) kinderspel

Programmeertalen zijn niet eenvoudig om te leren. Meer dan in menselijke taal zijn ze heel strikt gebonden aan regeltjes. Een fout stukje spelling of spraakkunst en ... oeps, het werkt niet zoals je wil of zelfs helemaal niet. Je bent trouwens niet de enige die programmeren moeilijk vindt. Jef Raskin die in een grijs verleden de interface van de Apple Macintoshcomputer ontwierp, schrijft:

> *"There is no question that modern systems are becoming increasingly complex and that programming tools need to accommodate this increasing complexity. Simple things have been made unnecessarily difficult, and we have failed to provide sufficient and sufficiently well-designed software tools needed to ease the difficulties of working in today's computer environment".*

Als je nooit eerder hebt geprogrammeerd, kan je je vermoedelijk moeilijk voorstellen wat programmeren nu precies allemaal inhoudt. Hoe begin je eraan? Moet je die codes en instructies werkelijk allemaal uit je hoofd kennen? Stel dat je een tekenprogramma wil schrijven, moet je dan ook eerst een "venster" programmeren waarin jouw stuk software wordt weergegeven?

Het antwoord is daarop niet eenduidig: *soms wel en soms niet.*

Hulpmiddelen voor de programmeur

Programmacode schrijf je niet in een tekstverwerker, maar in een **editor**. Dat is een eenvoudig tekstprogramma dat in veel gevallen automatisch tips geeft aan de programmeur, zodat hij de code niet helemaal uit zijn hoofd moet kennen en ook in kleur aanduidt of de code correct is en/of fouten bevat.

Daarnaast bestaan er ook heel wat **IDE's (integrated development environments)**, waar de programmeur ook eenvoudig interfaces voor zijn programma mee kan bouwen. Hij sleept bijvoorbeeld een knop op een scherm en programmeert dan wat er moet gebeuren als de gebruiker op zo'n knop klikt.

Om kinderen te leren programmeren zijn er **visuele programmeeromgevingen** ontwikkeld zoals Scratch. Op de volgende pagina gaan we aan de slag met Processing.

Een eenvoudig voorbeeld in Processing

Processing is een op de programmeertaal JAVA gebaseerde programmeertaal en ontwikkelomgeving, bedoelt voor artiesten, grafische vormgevers en iedereen die net echt thuis is in programmeren. Je bouwt er in een oogwenk blitse interactieve animaties en tekeningen mee.

Je begint met het schrijven van de functie `void setup() {}` waarmee je het programma start. Bekijk het als de "sleutel" waarmee je de automotor start. Tussen de akkolades schrijf je alle instructies die het programma moet uitvoeren als het start.

```
void setup(){

     background(255,255,255);
     size(800,600);

}
```

De opdracht `background(255,255,255);` ontvangt tussen de haakjes drie getallen: een waarde voor rood, groen en blauw. 255 is de maximumwaarde. Als we de drie basiskleuren hun maximumwaarde geven, dan verandert de achtergrond in .... wit. De instructie `size(800,600)` bepaalt dat het programma gestart moet worden in een venster van 800 bij 600 pixels. Je merkt dat je hier decimale waarden kan ingeven. Je voert twee getallen in gescheiden door een komma (opgelet: het gaat niet om een kommagetal). De compiler zal deze instructies nadien vertalen naar bytecode die in een "container" wordt gedraaid. Je moet dus niet zelf nullen en enen intikken.

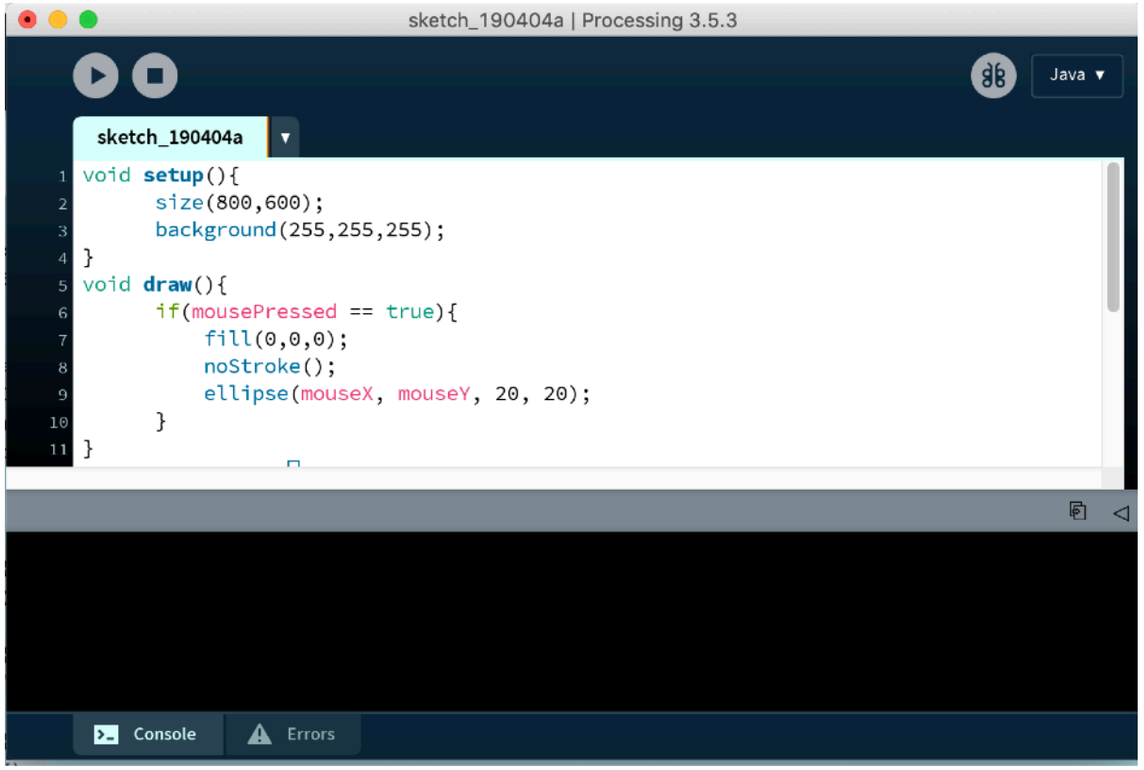
Vervolgens kan je programmeren wat er moet gebeuren en/of wat de gebruiker allemaal kan doen. Hiervoor gebruik je een tweede functie `void draw(){}`.

```
void draw(){

     fill(0,0,0);
     ellipse(width/2, height/2, 40, 40);

}
```

We tekenen hier een ellips op het scherm. Die ellips krijgt vier parameters. De eerste waarde betreft de positie van het middelpunt op het scherm. Die wordt bepaald aan de hand van de x- en de y-waarde. Door die in te stellen op width/2 en height/2, staat onze ellips precies midden in het programmavenster. De twee volgende waardes 40 en 40 bepalen de horizontale en verticale straal. Door de waardes gelijk te maken krijgen we geen ellips, maar een cirkel. De instructie `fill()` werkt net als background(). Maar we vullen die nu met zwart: de waardes voor rood, groen en blauw zetten we daarom op 0, het ontbreken van elke kleur.



Wijzigen we de code nu als volgt, dan tekent het programma enkel een ellips als we de muis indrukken. Bovendien tekent het programma de ellips dan op de plaats waar de muiscursor zich dan op het scherm bevindt. Het resultaat is een eenvoudig penseel waarmee je kan tekenen op je scherm.

```
void draw(){

      if(mousePressed == true){

          fill(0,0,0);

          noStroke();

          ellipse(mouseX, mouseY, 20, 20);

      }

}
```

Je kan het programma nu compileren door op de startknop te drukken. Maar het is ook mogelijk om het meteen te bewaren als een echte app voor Mac OS, Linux of Windows.

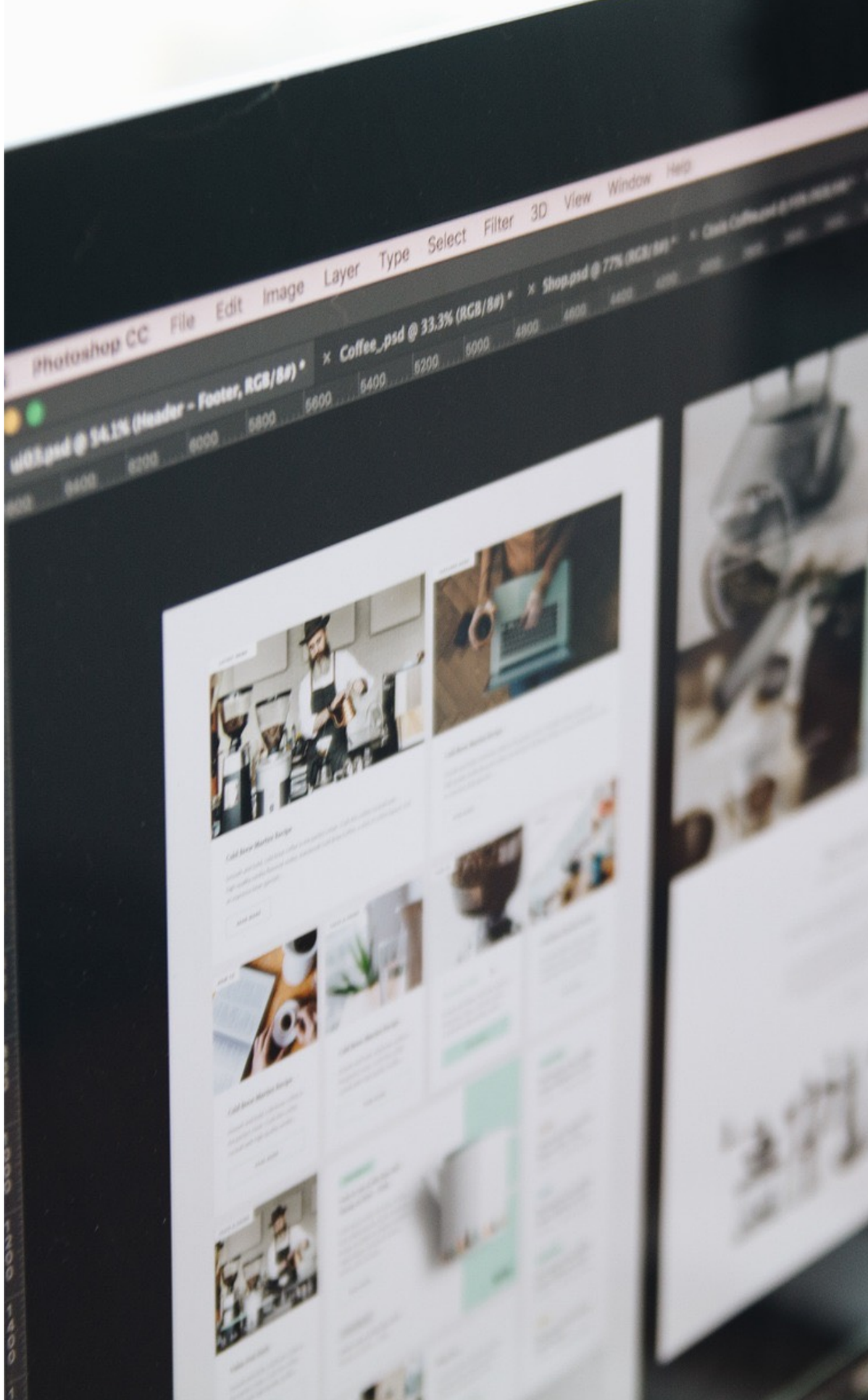
Je kan Processing gratis downloaden vanaf www.processing.org. Voorbeelden en lessen vind je op www.schoolvoorbeeld.be/multimedia/processing

SOORTEN SOFTWARE: OPERATING SYSTEMS, API'S, APPLICATIES, SAAS...

Vaak stelt men het nogal simplistisch voor: een computer bestaat uit hardware en software. Maar zoals je wellicht al begrijpt als je het eerste deel hebt gelezen, zo eenvoudig is het niet…

Software bestaat op diverse niveaus:

1. een basissysteem (bios, uefi) dat de contacten tussen alle onderdelen regelt,
2. een besturingssysteem (zoals Android, iOS, Linux, Mac OS X, UNIX, Windows...)
3. of gewoon software met een heel specifieke taak (zoals GIMP, MS Word of... Processing) of functionaliteit.



Operating systems

Een computer is eigenlijk niet veel meer dan een krachtige rekenmachine. Het Engelse "to compute" betekent niet meer of minder dan "berekenen". Het rekent met nullen en enen en sluist die door een reeks processors (verwerkers) en circuits. In vergelijking met een microcontroller of SoC is de laptop, tablet, smartphone of desktopcomputer waarmee je werkt, vooral bedoeld als een zeer veelzijdig toestel. Je moet het voor meerdere taken kunnen inzetten. Het is geen ijverige werkmier of honingbij die voor één specifieke taak wordt ingezet.

Een **OS of operating system** maakt het mogelijk om complexe programma's uit te voeren die gebruik maken van de aanwezige hardware. De mogelijkheden van de hardware vormen vaak de grens waar software aan gebonden is. Een operating software is over het algemeen niet geprogrammeerd in de aanwezige chips of hardware, want dan zou een operating system weinig flexibel zijn. Besturingssystemen zoals **Android, iOS, Mac OS X, Linux, Windows**… moet je op een computer installeren. In de meeste gevallen koop je zo'n toestel en is de installatie al vooraf gebeurd.

Cool, maar wat doet zo'n besturingssysteem nu eigenlijk? Vergelijk het met de manager in een bedrijf. Een OS bekijkt welke taken aan de gang zijn en verdeelt de middelen die hiervoor beschikbaar zijn. Het verdeelt het RAM-geheugen (het tijdelijke werkgeheugen dat wordt gewist als je het toestel uitschakelt), de benodigde opslagruimte, toegang tot randapparatuur zoals externe harde schijven en printers, rekenkracht van de processor...

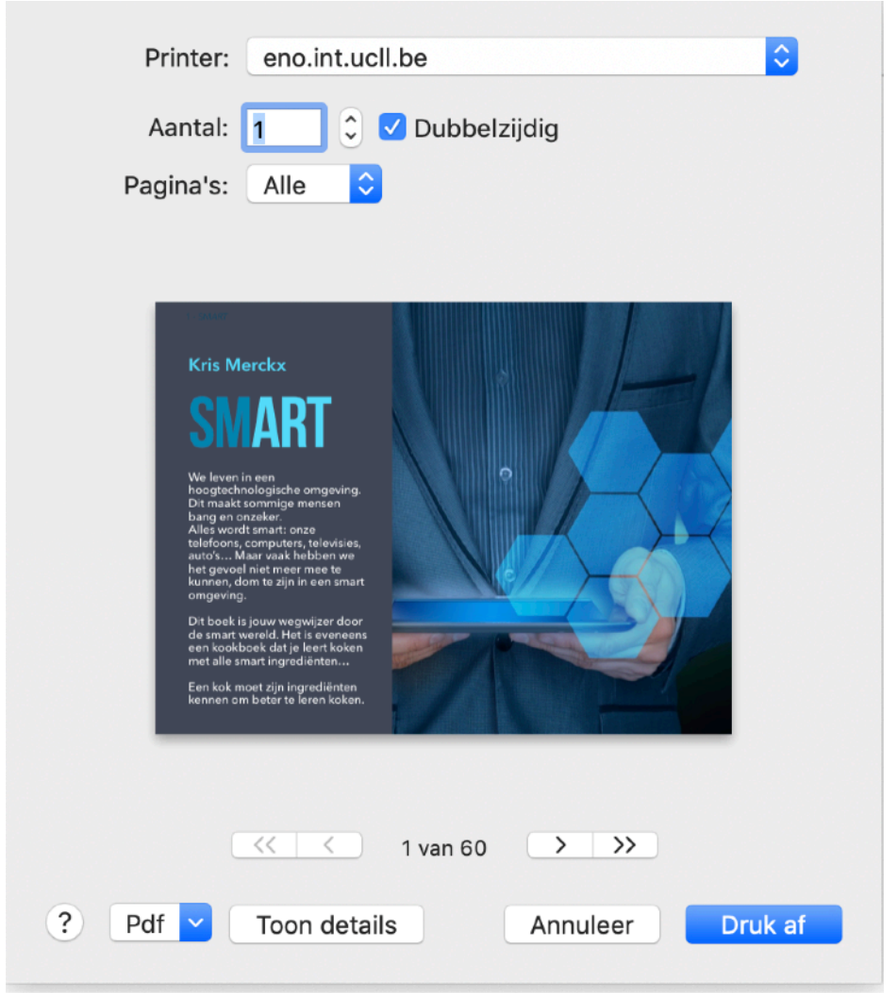
In het ideale geval zorgt het OS ervoor dat iedereen (elke gebruiker, elk stuk software...) zijn taak naar wens kan uitvoeren en het systeem niet crasht wegens overbelast

API's

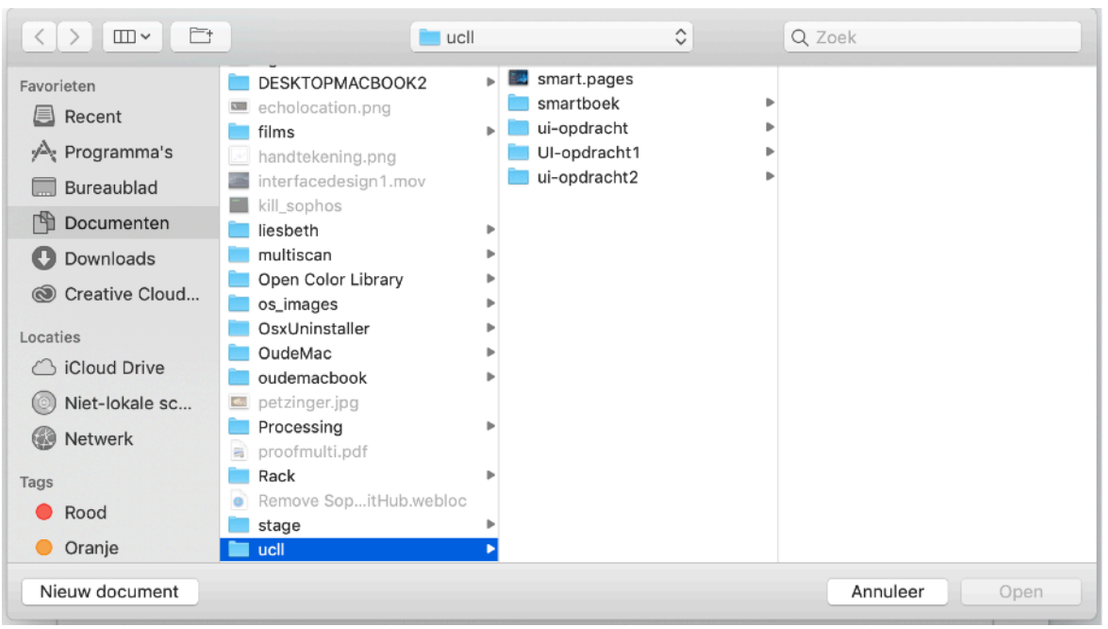
Een programmeur die een programma schrijft voor een specifieke taak, moet niet van nul beginnen. Hij beschikt reeds over een aantal in het systeem aanwezige functies of mogelijkheden. Vergelijk het naar analogie even met iemand die een huis bouwt: je moet niet zelf je bakstenen bakken, planken zagen, kalk blussen of je eigen truweel smeden. Je kan heel eenvoudig bestaande elementen gebruiken om het gewenste eindresultaat te bereiken. Als een programmeur ervoor wil zorgen dat de eindgebruiker een bestand kan afdrukken, openen of bewaren, dan moet hij hiervoor niet alle code zelf schrijven.

Hij kan daarvoor gebruik maken van functies die al in het OS zitten. Het OS vormt een laagje tussen de software van de programmeur en de hardware. Die "laag" noemen programmeurs een **API of Application Programming Interface**.

Een voorbeeldje: wanneer je onder Mac OS X een bestand wil openen, krijg je in zo wat elk programma een gelijkaardig **dialoogvenster** (=window) te zien, met dezelfde functionaliteit. Die mogelijkheden om bestanden te openen, te bewaren, af te drukken... worden geleverd door de API's van het OS.



Het dialoogvenster om bestanden af te drukken, ziet er hetzelfde uit in alle programma's. De programmeurs kunnen die functionaliteit oproepen uit de "API" van het OS.



Het dialoogvenster om bestanden te openen in Mac OS X

Omdat niet alleen de processor, maar ook het besturingssysteem en de corresponderende api's verschillen, is het niet zonder meer mogelijk om een Windowsprogramma op pakweg een Mac OS X of Linuxmachine te installeren en omgekeerd.

Basisbesturing: BIOS, UEFI, firmware en bootloader

Meestal koop je een toestel met een kant-en-klaar geïnstalleerd besturingssysteem. Een PC of laptop komt met Windows, een Applecomputer met Mac OS X. Bij tablets of smartphones stellen we ons nog minder vragen: een iPad of iPhone komt met iOS, een andere tablet of smartphone met Android.

Als je een desktopcomputer zelf samenstelt, moet je het besturingssysteem zelf nog installeren. Toch toont zo'n OS-loze computer ook de nodige output als je hem opstart. Bij het opstarten kan je één of een desktop of laptop een of andere andere functietoets indrukken om in een setup- of bootscherm terecht te komen. Hetzelfde geldt wanneer je voor de eerste keer een Raspberry Pi aan je televisiescherm koppelt.

Wat je op dat moment te zien krijgt is de **firmware**. De firmware is een interface die zich als een dun schilletje over de hardware heen legt. Het vormt dus geen onderdeel van het besturingsysteem zelf. Het doel van dit firmwareprogramma is te controleren of alles ok is met de hardware. Zo'n beetje als de conciërge van een winkel of school die 's morgens de deuren opent en alles nog eens controleert om te zien of zich nergens problemen hebben voorgedaan. Op die manier controleert de firmware of de processor, het geheugen, de schijfstations en poorten geen fouten vertonen.

Op een Mac heet deze firmware **UEFI** (Unified Extensible Firmware Interface)22, op een Windowstoestel sprak men tot voor kort steevast over het **BIOS** (Basic Input Output System).

Na de firmware laat standaard de **bootloader** die op zoek gaat naar beschikbare besturingssystemen (op de harde schijf, in een schijfstation, op een USB-stick). Op de meeste systemen is slechts één besturingssysteem aanwezig, vindt de bootloader er meerdere, dan laat hij de gebruiker de keuze welk systeem hij wil opstarten.

Closed source en open source software & OS

Windows en Mac OS X zijn lang niet de enige operating systems. Op smartphone en tablet overheersen op dit moment Android en iOS.

Apple zet iOS en Mac OS X op de markt als commerciële software. Microsoft doet hetzelfde met Windows. De programmeercode of broncode val al deze commerciële software is gesloten (**closed source**). Je mag de code niet inzien of aanpassen. Je kan enkel de gecompileerde versie kopen of installeren. Dit principe geldt ook voor andere software. Zo zijn MS Office en Adobe Photoshop "closed source"-softwarepakketten.

Dat is niet altijd zo. Heel wat programmeurs publiceren de broncode van hun software online als "**open source**"-software. Je kan de gecompileerde versie downloaden evenals de originele broncode. In veel gevallen is het toegestaan om aanpassingen te doen in de code. Wat je er vervolgens mee mag doen, hangt af van de licentie. Sommige "**licenties**" laten toe dat je jouw eigen versie ook mag verkopen, maar verplichten je wel om je eigen aanpassingen ook als open source te publiceren. Soms mag je de code gebruiken, voor persoonlijke of niet-winstgevende doeleinden, maar niet verder verkopen. Andere licenties laten je volledig vrij. Tot de bekendste open source-softwarelicenties behoren: **GNU, MIT, Creative Commons**. In de wereld spreekt men ook vaak van "**copyleft**"-licenties (tegenover "**copyright**").

Een van de belangrijkste opensourceprojecten is gnu/Linux. Dit besturingssysteem werd als hobby ontwikkeld door Linus Thorvalds als alternatief voor Microsoft Windows. Hij gooide de basiscode van zijn systeem in 1991 op het internet. Sindsdien werken wereldwijd ontwikkelaars en bedrijven aan verbeteringen en uitbreidingen van het systeem. Om programmeurs die elkaar niet persoonlijk kennen via internet met elkaar te laten samenwerken en overlappingen te voorkomen, werden **versiecontrolesystemen** ontwikkeld. Dit zijn softwareprogramma's die het beheer van de code automatiseren en controleren. Daar de code vrij is, is het aan anderen toegestaan hun eigen versie te schrijven en publiceren. **Linux wordt vooral op internetservers** en door kenners gebruikt. Linux vormt ook de basis van het mobiele besturingssysteem Android. In de bedrijfswereld (banken, grote bedrijven…) gebruikt men vaak ook **UNIX** als besturingssysteem, ontwikkeld in 1969!

UNIX is een bijzonder krachtig, veilig en stabiel systeem. Het vormt de basis van ondermeer Mac OS X. In netwerken en bij bedrijven moet het echter steeds meer plaats ruimen voor Linux, dat zelf is gebaseerd op UNIX.

Op mobiele toestellen overheerst op dit moment Android. Android is een open source besturingssysteem ontwikkeld door Google Inc. op basis van de Linux-kernel. De open source-licentie heeft ertoe geleid dat ook andere fabrikanten het OS inzetten op hun toestellen. Terwijl iOS van Apple enkel samen met de iPhone of iPad mag verkocht worden.

Daarnaast bestaan er nog tal van kleine besturingssystemen, vaak hobbyprojecten van individuele ontwikkelaars of 'community's (groepen van ontwikkelaars die zich organiseren via internet).

Applicaties

Klassieke desktopprogramma's, maar ook mobiele applicaties, bestaan vaak uit duizenden, zo niet honderdduizenden regels code. Voordat die code door een computer uitgevoerd kan worden, moet ze – zoals eerder werd gezegd – worden gecompileerd. Dit wil zeggen dat de code door een stuk software (de compiler) in machinetaal (nullen en enen) wordt omgezet. De gecompileerde code kan rechtstreeks door de processor worden uitgevoerd.

Doorgaans downloadt en installeer je een gecompileerd programma een zogenaamde "**binary**". Het is onbegonnen werk om alle commerciële en "open/free" software op te lijsten.

In de open source wereld zegt men vaak dat hun software *"free"* is *"as in freedom, not as in beer"*. Je bent als gebruiker m.a.w. vrij om met de software te doen wat je wil, maar het is niet noodzakelijk bedoeld als gratis. Toch betekent het in de praktijk dat je voor de meeste commerciële software wel een gratis open source variant kan vinden.

Commerciële software	**Open source alternatief**
MS Office	LibreOffice
Adobe Photoshop	GIMP
Adobe Illustrator	Inkscape
Adobe Indesign	Scribus
Maya, 3D Studio Max	Blender
iMovie, Adobe Premiere	Shotcut
Windows	Linux
...	...

SAAS en webapplicaties

Meer en meer software werkt online of maakt op zijn minst de combinatie tussen offline en online mogelijkheden. Meer en meer zien commerciële bedrijven hun software niet als een te verkopen afgewerkt product, maar als dienstverlening (**SaaS= software as a service**). De software staat in de "cloud" en kan, mits je over de juiste licentie beschikt (en voldoende betaalt), van waar ook gebruikt worden.

Heel wat softwaretoepassingen draaien volledig op het web of in de cloud: Gmail, Google Docs, Facebook enz. zijn enkele van die veelgebruikte online services. Die dienstverlening kan tegen betaling gebeuren of volkomen "gratis" zijn. De producent/leverancier haalt zijn inkomsten uit reclame die vaak op basis van de inhouden van de gebruiker gegenereerd worden.

Die evolutie naar het zien van software als het leveren van een dienst is overal aanwezig. Weinig eindgebruikers zijn nog bereid om een paar duizend euro (of zelfs een paar tientallen euro) te betalen voor één licentie op een bepaald softwareproduct. Die evolutie kent vermoedelijke meerdere oorzaken:

1. De groei van het internet en de toegenomen snelheid door snelle verbindingen.

2. De kracht van open source software: Waarom betalen als je even goed kan werken met een gratis product?

3. Software die als "product" werd verkocht op bijvoorbeeld een CD-rom kon gemakkelijk gekopieerd worden. Software die je via een "huiseigen" installatieprogramma op je computer plaatst en bij elk gebruik de licentiegegevens online controleert, is veel minder makkelijk te kopiëren.

4. Lang niet iedereen heeft alle functionaliteit nodig. Veel gebruikers zijn tevreden met kleine apps met een welbepaalde mogelijkheid. Kijk maar naar het succes van de appstores van Apple en Google.

5. Software in de cloud laat je toe om van overal te werken en zelfs makkelijk samen te werken.

DEEL 3 VERWERKING VAN DATA

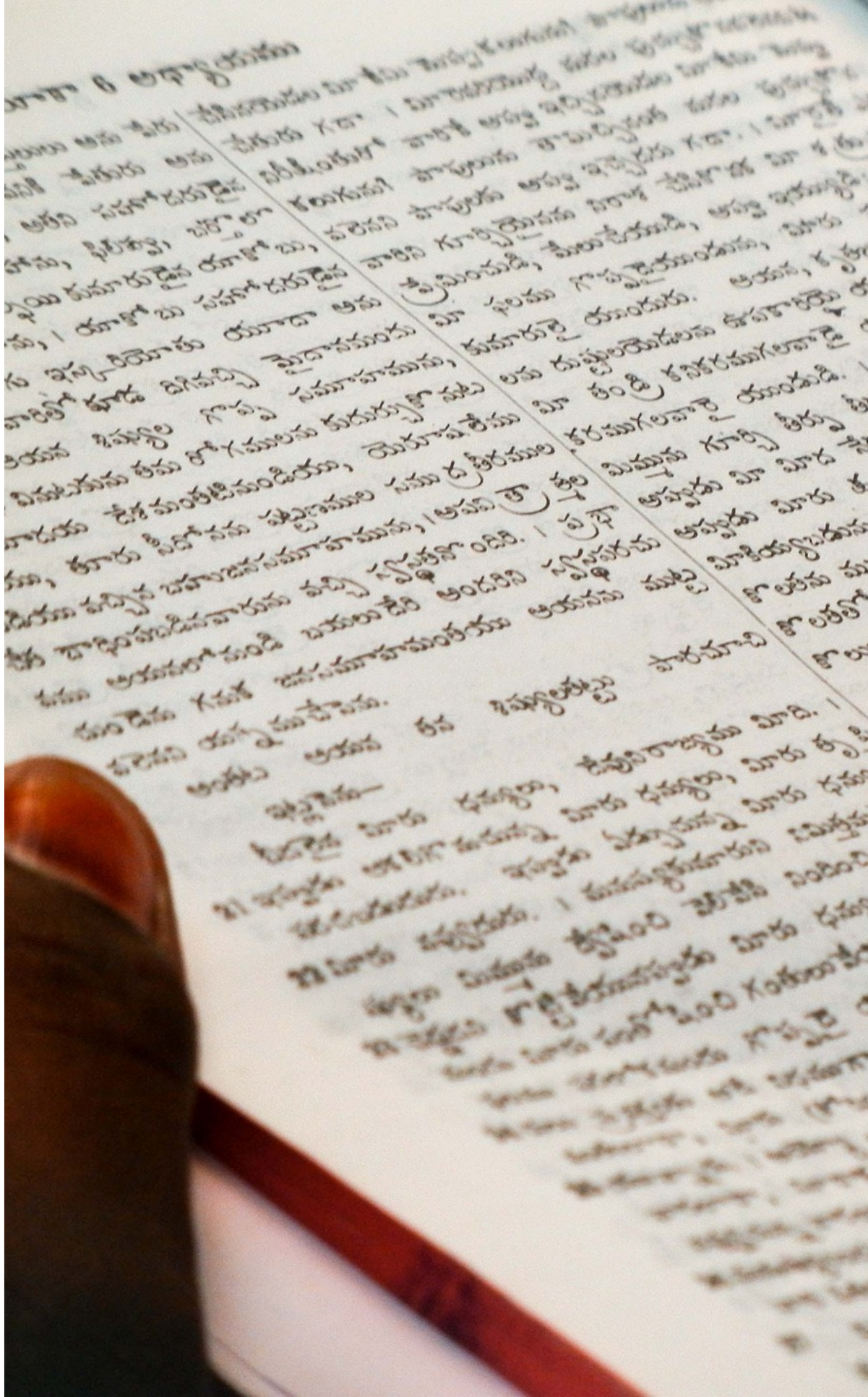
- Waarom hebben computers het moeilijk met menselijke taal?
- Hoe kunnen we er voor zorgen dat computers de inhoud van tekst en afbeeldingen echt leren begrijpen?
- …

VERWERKING VAN TEKST EN TAAL

Computers hebben het moeilijk met menselijke taal. Zeggen dat dit komt omdat computers enkel binaire getallen kennen, is wat te simplistisch. Het klopt slechts voor een stuk. Als je een tekst invoert op je toetsenbord, vertaalt de computer elke toetsaanslag naar een code. Elke toets op een toetsenbord krijgt intern een cijfer toegewezen. Een computer begrijpt dus niet wat je invoert.

Maar het probleem is veel ruimer. Wereldwijd bestaan er ongeveer 6500 menselijke talen, elk met hun eigen woordenschat, spelling en spraakkunst. Elke taal kent in de vorm van streekdialecten nog diverse varianten. Ook al zijn heel wat talen verwant met elkaar (bijvoorbeeld de Indo-Europese talen), dan nog is het niet eenvoudig om software al die talen te leren begrijpen. Naast het westerse alfabel bestaan nog andere lettersystemen.

Bovendien verschillen de geschreven en gesproken versies van één taal. Naast het begrijpen van een taal, is het genereren van taal nog een ander heikel punt.



Wanneer een docent je op een examen een vraag stelt, dan kan je daar in het beste geval vlot op antwoorden. Je begrijpt de vraag. Je maakt in je hoofd de juiste analogie en vergelijkingen. Je kan de inhoud simultaan *vertalen* in je eigen woorden. Je leert het niet uit je hoofd en je rammelt het niet woord voor woord betekenisloos af. Dat zou je natuurlijk kunnen doen, maar een docent verwacht dat je intrinsiek verbanden kan leggen en de betekenis snapt.

Sommige woorden hebben meerdere betekenissen ('haar' of 'bal' bijvoorbeeld). Soms kan je de betekenis van een woord pas afleiden uit de context (de rest van de zin of tekst).

Lees het kleine verhaaltje hieronder:

> De hond slaapt in zijn mandje. Jacky is heel erg lief. Als ik thuis kom, loopt hij al kwispelstaartend naar me toe. Waar blijft hij vandaag? Weet jij het? Ik ben zo bang dat hij weggelopen is.

Als menselijke lezer snap je meteen waarover dit korte tekstje gaat. Hoe kunnen we een machine leren om deze tekst te begrijpen? Tussen mens en machine is er in dit geval al meteen een immens verschil. Een kind leert al snel een visueel beeld van een hond te koppelen aan het begrip "hond" en de bijhorende geluiden. Denk maar aan de mama's en de papa's die vragen: "En, hoe doet de hond? Nee, niet miauw."

Het taalmechanisme in het hoofd van het kind koppelt aan de hond bepaalde acties zoals blaffen, lopen, kwispelstaarten, maar ook onderdelen zoals staart, poten, tong. Een heel kader, onderdeel van de "wereldontologie" in het hoofd van het kind.

Maar hoe vertalen we die taalverwerking naar een computer? Oeps, dat is wat anders.

Laat ons eerst eens de tekst scheiden in zijn onderdelen: zinnen (**sentence breaking**). Dat kan eenvoudig door te zoeken naar leestekens zoals punten, vraagtekens of uitroeptekens. Dit werkt perfect voor heel wat talen, maar niet voor pakweg Arabisch, Chinees of Japans, want daar is het gebruik van leestekens onbekend. Vervolgens kunnen we de zinnen splitsen in hun diverse woorden (**word segmentation**), maar dat werkt al evenmin voor de genoemde talen: Arabisch kent bijvoorbeeld geen door spaties onderscheiden woorden. Daar bepaalt het uitzicht van een bepaald karakter (letter) of het gaat om het begin, het midden of het einde van een woord.

Zinnen bepalen of ze iets meedelen of eerder iets vragen (het vraagteken), en vragen op hun beurt kunnen open of gesloten (ja of neen, if of else, 1 of 0) zijn. De soort zinnen wordt bepaald door de "discourse analysis". De machine kijkt vervolgens naar de individuele woorden en probeert die te herleiden tot hun basis of stam in een proces dat "**stemming**" en "**morphological segmentation**" heet. In een taal als het Engels is dit een relatief eenvoudig proces:

open is de "stam" van open, opens, opening, opened…

In talen zoals het Turks is dit een uiterst moeizame techniek. De stam is niet noodzakelijk hetzelfde als de morfologische "wortel" van het woord. Meestal volstaat het dat verwante woorden dezelfde stam bevatten. Heel wat zoekmachines behandelen woorden met dezelfde stam als synoniemen om de zoekterm enigszins uit te breiden (conflatie). Google gebruikt "stemming" sinds 2003. Voordien zou de zoekterm "vis" niet "vissen" of "gevist" opgeleverd hebben.

Dit leidt soms tot foute resultaten. De machine kan teveel resultaten leveren (overstemming) zoals wanneer je "universiteit" zoekt en ook "universum" krijgt. In dit geval behandelt de machine beide woorden door hun gemeenschappelijke stam als synoniemen. In andere gevallen krijg je te weinig zoekresultaten (understemming).

Bij woorden zoekt het NLP-systeem niet alleen naar de stam, maar bovendien ook naar de woordsoort: gaat het om een adjectief of een zelfstandig naamwoord? Bovendien kunnen zelfstandige naamwoorden ook eigennamen zijn.

Jacky *is een hond.*

Deze techniek van "**named entity recognition**" werkt simpelweg door naar hoofdletters te kijken. Maar ook dit levert problemen op. Bij het begin van een zin schrijf je steevast een hoofdletter. In het Duits krijgen dan weer alle naamwoorden een hoofdletter.

Jacky *ist ein* ***Hund****.*

En in het Engels schrijf je ook een hoofdletter als je het over jezelf hebt:

I*,* ***I*** *wanna be your dog.*

De woordsoort wordt vaak bijgehouden in een soort lexicon, een soort lijst waarin op basis van de stam en eventuele voor- en achtervoegsels de woordsoort wordt bepaald (**Part-of-speech tagging**).

Maar eenvoudig is dit niet. Immers, afhankelijk van de zin kan een woord iets compleet anders betekenen:

Ik speel met de bal op het bal.

Je merkt het al. De betekenis zit niet in het woord alleen, maar ook in zijn relatie met andere woorden: homoniemen (**word sense disambiguation**). De volgende zin klinkt afhankelijk van tegen wie (je docent) of wat (een eik) je het zegt helemaal anders, maar een NLP-systeem heeft het er moeilijk mee:

Je bent een eikel.

Via de techniek van "**natural language understanding**" probeert een NLP-systeem logica te vinden in korte fragmenten tekst. Welke adjectieven horen bij zelfstandige naamwoorden (conference resolution)? Wat zijn de onderlinge relaties tussen objecten in een zin (relationship extraction)? Op die manier probeert een NLP-systeem stilaan de zin te "parsen", een stamboom te maken van een zin en tenslotte van het hele verhaal.

> "Je bent een eikel," zei ik tegen de directeur. Ik was woedend. Hij werd door mijn uitspraak eveneens razend.

Deze zin verklapt hoe beide partijen zich voelen. Een NLP-systeem kan op basis van de betekenis van woorden en de frequentie waarmee ze voorkomen, raden wat het gevoel is dat uit een tekst spreekt: **sentiment analysis**.

Spelling- en spraakkunstcorrectie

In heel wat tekstverwerkers zit spelling- en spraakkunstcorrectie. MS Word bijvoorbeeld duidt met een rood lijntje aan waar je woorden fout hebt gespeld. De programmeurs hebben de spellingregels in de software geprogrammeerd. Open bijvoorbeeld eens een Duitse of Franse tekst in een tekstverwerker die in het Nederlands is ingesteld. Dan lijkt het alsof je alles fout hebt geschreven. Dit komt omdat op dat moment enkel de Nederlandstalige spellingcorrectie is geladen.

Dankzij de evolutie van **AI (artificiële intelligentie)** heeft **NLP (natural Language processing)** grote stappen voorwaarts genomen. Spelling- en spraakkunstregels worden niet langer per taal gecodeerd (rule based). De software leert uit andere teksten zelf hoe de regels in elkaar zitten. Het helft de computer ook om de inhoud van de teksten **semantisch** te begrijpen. Op AI komen we in een later deel nog uitgebreid terug.

Robotjournalisme

Tekstbegrip kan dan voor digitale systemen nog relatief moeilijk zijn, toch zijn steeds meer artikels niet meerdoor mensen geschreven. Google kocht Jetpac, een app die op basis van image recognition-algoritmes automatisch stadsgidsen genereert. Associated Press laat duizenden artikels schrijven door robotschrijvers.

Betekent dit dat het einde van de "menselijke journalist" in zicht is? Het onderstaande artikel werd door een robotjournalist geschreven:

Aerie Pharmaceuticals Inc. (AERI) on Tuesday reported a loss of $13.1 million in its third quarter. The Research Triangle Park, North Carolinabased company said it had a loss of 54 cents per share. Losses, adjusted for stock option expense, came to 44 cents per share.

Het artikel voldoet aan de vereisten voor dit soort artikels: het is correct en gedetailleerd. Uiteraard moet je je bij een robotjournalist niet een humanoïde robot voorstellen die op een toetsenbord teksten zit in te tikken. Het gaat om een stuk software dat "natuurlijke taal" produceert. Auteur van dienst is in dit geval WordSmith van de softwareproducent Automated Insights. In 2013 produceerde WordSmith wereldwijd 300 miljoen nieuwsverslagen voor diverse klanten, meer dan alle "menselijke" journalisten samen en vooral veel goedkoper...

Volgens Automated Insights komen de jobs van echte journalisten niet in gevaar:

"We're producing articles that never would have existed in the first place," he says. "AP was doing 300 corporate-earnings stories per quarter; they're now doing about 4,440. So 4,100 of these stories would not exist without Wordsmith." "The computer handles the who, what, where and when, and humans are freed up to ask why and how."

De LA Times zet Quakebot in om geautomatiseerd artikels af te leveren op basis van data van de US Geological Survey (aarbevingen). Inleving, gevoel, medeleven.... zijn vreemd bij dit soort artikels, maar dat is voor "objectieve" verslaggeving ook niet nodig. "Creatief schrijven" ligt voorlopig nog buiten de mogelijkheden, want de algoritmes voeden zich met gestandaardiseerde informatie, een beetje te vergelijken met gestructureerde data in een rekenbladprogramma. Een van de mogelijke pistes om meer menselijke "warmte" in een tekst te steken is door de data aan te vullen met informatie geleverd door sensoren."

Vaak hoor je dat Wikipedia-informatie onbetrouwbaar is. Nochtans zijn een massa artikels niet meer door menselijke auteurs afgeleverd maar door software-algoritmes. De meest bekende Wikibot luistert naar de naam Lsjbot, geprogrammeerd door de Zweed Sverker Johansson. Lsjbot schraapt informatie van betrouwbare bronnen en schrijft korte artikels over onderwerpen gerelateerd met dieren. Per dag levert de bot ongeveer 10000 artikels af. In totaal zouden ongeveer 8.5% van de artikels van de (Zweedse) Wikipedia geschreven zijn doorLsjbot.

Wikipedia zette voor het eerst een bot in in 2002. Die "Rambot" leverde per dag duizenden artikels op over zo wat elk(e) dorp, stad of staat van de Verenigde Staten en ook van een aantal andere landen. Een robot onder de gepaste naam Asteroids schraapte data van de NASA en leverde artikels over... asteroïden. Op dit moment zijn er meer dan duizend verschillende Wikibots aan het werk: ze verbeteren teksten en tegenstrijdigheden... De meest actieve is Cydebot die meer dan 4.5 miljoen aanpassingen aanbracht. 15

Toch getuigt de techniek voor het geautomatiseerd schrijven van artikels niet altijd van “kunstmatige intelligentie”. Quakebot gebruikt bijvoorbeeld door mensen opgestelde sjablonen en vult die met vooraf verzamelde en gestructureerd bewaarde data over aardbevingen. Het eindresultaat klinkt als een door mensen geschreven artikel, en dat is het voor een groot deel natuurlijk ook. Maar de robot doet in wezen niet veel meer dan het vullen van sjablonen met gestructureerde data. Andere ontwikkelaars, zoals Narrative Science proberen robotjournalisme naar een ander niveau te tillen. Zij pogen de software zelf te laten uitmaken waarover de data gaan en op basis daarvan “content” te laten genereren. De software beslist in dit geval zelf hoe de inhoud moet klinken.

Gesproken tekst en spraaksynthese

Vooraleer machines vlekkeloos in menselijke talen kunnen communiceren, is er nog een hele weg af te leggen. Het draait niet enkel om de semantiek. In de Studio 100-reeks ROX praat de gelijknamige auto vlekkeloos met de andere (menselijke) personages. Hij herkent en interpreteert hun vragen en antwoorden en reageert er vaak laconiek op. Onze vriend “ROX” combineert het beste op het vlak van NLP met speech recognition, speech segmentation en speech processing en als kers op de taart **natural language generation**.

Het verstaan van gesproken taal (speech recognition) is nog een aardig stuk moeilijker dan het lezen of interpreteren van gedrukte tekst omdat er in gesproken tekst nauwelijks pauzes zijn tussen woorden. Ook letters vloeien onhoorbaar in elkaar over. Het verdelen van een uitspraak in afzonderlijke woorden (speech segmentation) is vreselijk moeilijk te programmeren.

Natural language generation, het genereren van taal in geluidsvorm, is daarentegen al aardig ingeburgerd in bijvoorbeeld GPS-systemen. De software kan in dat geval gebruik maken van vooraf opgenomen geluid (afzonderlijke woorden of zinnen) dat het tot de gewenste zinnen samenvoegt. Dat is echter nog heel wat anders dan het kunstmatig "toon per toon" genereren van spraak (**speech synthesis**).

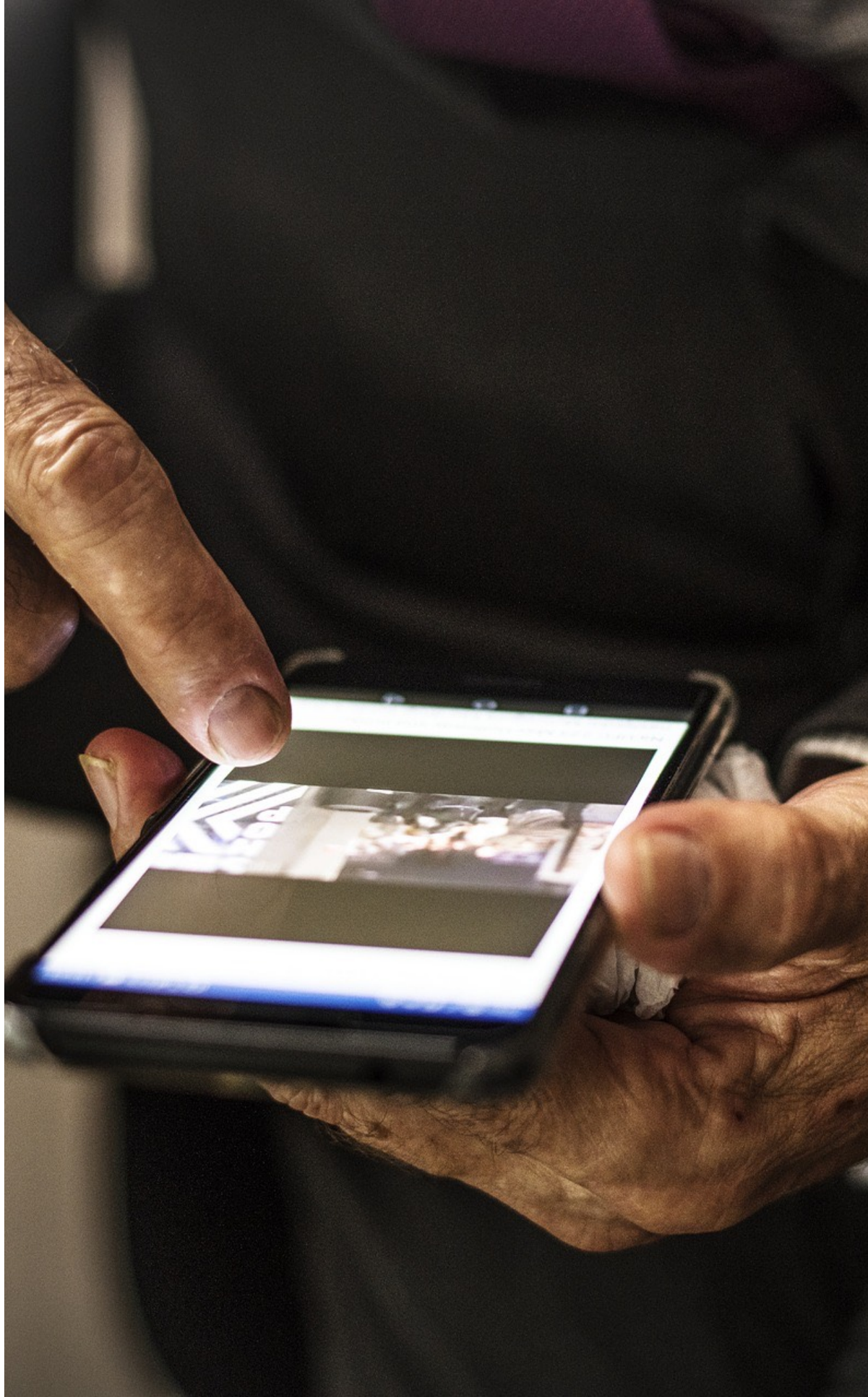
In machine translation komen alle toeters en bellen van NLP-systemen en natural language generation samen. Zinvolle vertalingen laten gebeuren door een machine is dus allesbehalve evident. Het vraagt niet alleen begrip van beide talen, maar ook het betekenisvol begrijpen en converteren van het ene bekenissysteem (taal) naar het andere (een andere taal). Een voorbeeld: In het Chinees plaatst een spreker geen "tijd" in zijn zin. Je kan dus niet aan de persoonsvorm zien wanneer de actie zich heeft afgespeeld. Tijd heeft in het Chinees geen belang wanneer je schrijft of spreekt, een volledig andere manier van "denken" dan sprekers van Indo-Europese talen. Allesbehalve eenvoudig dus om te vertalen van pakweg het Chinees naar het Nederlands of omgekeerd. Denk dus zeker nog eens na wanneer mensen spotten met de vertalingen van bijvoorbeeld Google Translate. Google Translate werkt bovendien met machine learning. De software leert dus voortdurend zelf bij en wordt alsmaar beter. Over machine learning en artificial intelligence lees je meer in de rest van dit boek.

VERWERKING VAN BEELD

We denken er niet altijd meer over na, maar toch wekt het wel verbazing. Een computerscherm kan om het even welk beeld uit de werkelijkheid (een foto, een tekening, een film…) op een beeldscherm weergeven. Hoe werkt dit nu precies?

Bovendien kan je zo'n digitale afbeeldingen ook afdrukken. Het lijkt misschien allemaal één pot nat, maar er is een groot verschil tussen afgedrukte digitale afbeeldingen en digitale afbeeldingen op een beeldscherm.

Ook al kan een computer iedere denkbare afbeelding weergeven, begrijpen wat op een afbeelding staat, is nog een ander paar mouwen.



Afbeeldingen digitaliseren

Hoe zet een computer een beeld van de werkelijkheid (een foto) om in digitale codes? Het principe achter digitalisering is even eenvoudig als geniaal, en zeker niet nieuw of modern. Vergelijk het met de werking van een kruiswoordraadsel met heel veel vakjes (en dus ook heel veel rijen en kolommen). Elke vakje in zo'n "kruiswoordraadsel" kan je inkleuren met een bepaalde kleur. Op die manier kan je elk gewenst beeld uit de werkelijkheid in een "kruiswoordraadsel" weergeven. Digitale afbeeldingen werken op een gelijkaardige manier.

In dit kruiswoordraadsel herken je een smiley Een afbeelding van een smiley werkt precies hetzelfde, maar bevat veel meer pixels (kleine vakjes, kolommen en rijen).

Uiteraard spreken we bij digitale afbeeldingen niet van *kruiswoordafbeeldingen*. In wiskundige termen noemen we zo'n **raster** of "tabel" een **matrix**. Een afbeelding van 800 bij 600 pixels telt op die manier 800 kolommen en 600 rijen. Elke vakje in het raster of de matrix noemen we een **pixel** (afgeleid van picture element). Zo'n afbeelding van 800 bij 600 telt op die manier 480 000 pixels (800*600). Wanneer het totale aantal pixels in een afbeelding een miljoen bedraagt, spreken we van een **megapixel**. *Omdat meer pixels zorgen voor meer detail, bepaalt het aantal mexapixels ook de kwaliteit van de afbeelding.*

Kleuren mengen met RGB

Een vakje in een kruiswoordraadsel is doorgaans of zwart of wit. Een afbeelding of foto bevat in de meeste gevallen een hele resem aan kleuren. Elke pixel krijgt een eigen kleur. Aan de binnenkant van een beeldscherm zitten de zogenaamde pixels (picture elements), kleine "punten" diehet licht in een bepaalde kleur kunnen uitzenden.Een beeldscherm (computerscherm, smartphonescherm, display in een auto…) geeft aan elke beschikbare pixel een kleur door de waardes voor ROOD, GROEN en BLAUW met elkaar te mengen. Als een beeldschermpixel *uit* staat, *dan zendt hij geen licht uit*.

Dit betekent dat de waardes voor de **drie hoofdkleuren (RGB)** op 0 staan (R = 0, G = 0, B = 0). Een beeldscherm is standaard zwart, maar als je de pixels "aanzet", zenden ze licht uit door rood, groen en blauw met elkaar op te tellen. Daarom spreken we ook van **additieve (Engels: to add) kleuren.**

De computer probeert een afbeelding zo compact mogelijk te bewaren om niet te veel opslagruimte in beslag te nemen. Iedere beeldschermpixel krijgt voor elke hoofdkleur maximum 8 bits opslagruimte. Zwart komt overeen met de waarde R = 0000 0000, G = 0000 0000, B = 0000 0000. Als je al die waardes voor de drie kleuren op 1 zet (dus: 1111 1111) , dan krijg je een witte pixel. De waarde 1111 1111 komt decimaal overeen met 255. Als je de waarde voor ROOD op 255 zet en je doet dit ook voor GROEN en BLAUW, dan krijg je dus… wit. ROOD bekom je met combinatie R = 255, G = 0, B = 0 enz.

Hexadecimaal

Om RGB-kleurwaardes te bewaren, gebruik je vaak hexadecimale waarden. Wit kan je op die manier decimaal uitdrukken als rgb(255,255,255) en hexadecimaal als #FFFFFF. "FF" staat immers voor het getal 255

Color	Color Name	Hex Code #RRGGBB	Decimal Code R,G,B
	maroon	#800000	(128,0,0)
	dark red	#8B0000	(139,0,0)
	brown	#A52A2A	(165,42,42)
	firebrick	#B22222	(178,34,34)
	crimson	#DC143C	(220,20,60)
	red	#FF0000	(255,0,0)
	tomato	#FF6347	(255,99,71)
	coral	#FF7F50	(255,127,80)
	indian red	#CD5C5C	(205,92,92)
	light coral	#F08080	(240,128,128)
	dark salmon	#E9967A	(233,150,122)
	salmon	#FA8072	(250,128,114)
	light salmon	#FFA07A	(255,160,122)
	orange red	#FF4500	(255,69,0)
	dark orange	#FF8C00	(255,140,0)
	orange	#FFA500	(255,165,0)
	gold	#FFD700	(255,215,0)
	dark golden rod	#B8860B	(184,134,11)

Een aantal RGB-kleurwaardes in hexadecimale en decimale notatie.

Het kleurenpalet in GIMP De hexadecimale notatie heet hier HTML-notatie.

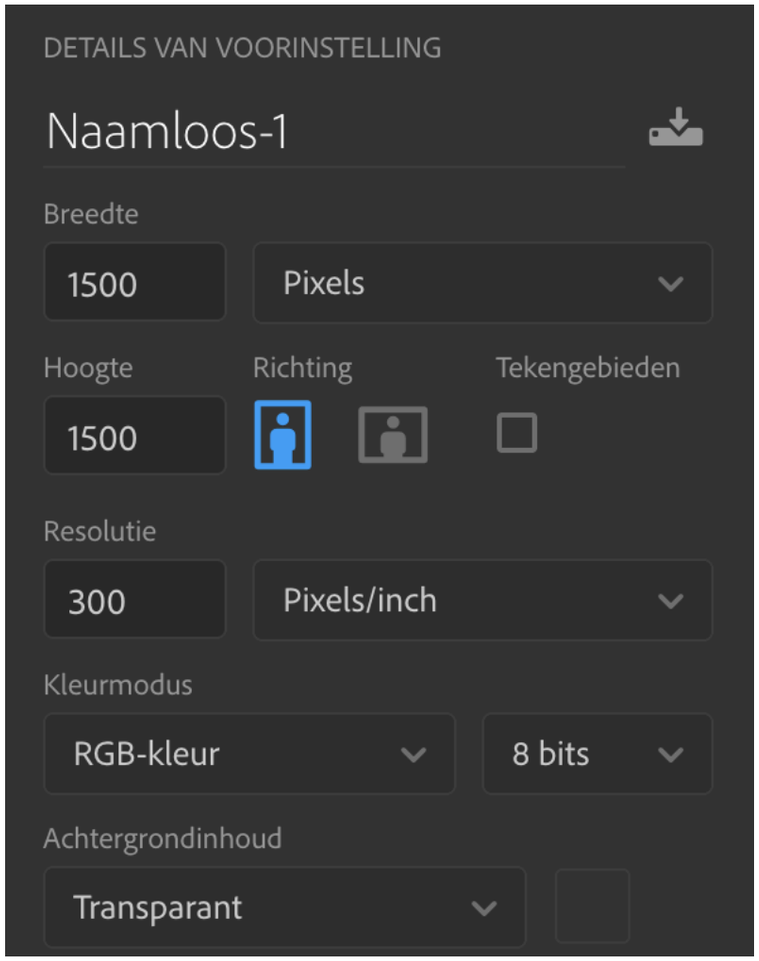
Resolutie en DPI

De **resolutie** van een beeldscherm vertelt hoeveel pixels een scherm horizontaal en verticaal kan weergeven. De resolutie kan je op twee manieren uitdrukken. Ofwel gebruikt men hiervoor de verhouding tussen hoogte en breedte (bijvoorbeeld: 1920x1280), ofwel drukt men de resolutie uit met het aantal pixels per inch (bijvoorbeeld: 72 dpi) of **dots per inch (DPI)**.

De resolutie van een digitale camera wordt vaak in **megapixels** aangegeven. Dit is dan de resolutie (of het aantal pixels) die de CCD- sensor aankan. Om het correcte aantal pixels te weten vermenigvuldig je het aantal horizontale lijnen met het aantal verticale lijnen van de digitale foto. Zo is dus een camera die een foto produceert met een resolutie van 1280 x 960 pixels een 1.3 megapixel camera en een 5 megapixels camera zal dan een foto produceren van 2560 x 1920 pixels.

Hoe meer pixels, hoe groter het "bestand". Hoe meer pixels een afbeelding bevat, hoe meer informatie over de individuele kleuren moet bewaard worden. Dit leidt dan ook tot een "groter" bestand (grotere **bestandsgrootte**) dat meer ruimte inneemt op het opslagmedium.

De bestandsgrootte hangt ook af van de kleurcodering. Als een afbeelding enkel zwart en wit bevat, dan is ze beduidend kleiner, dan wanneer je de **RGB-kleurcodering** gebruikt.



De resolutie van een afbeelding in Adobe Photoshop.

Beeldbewerking

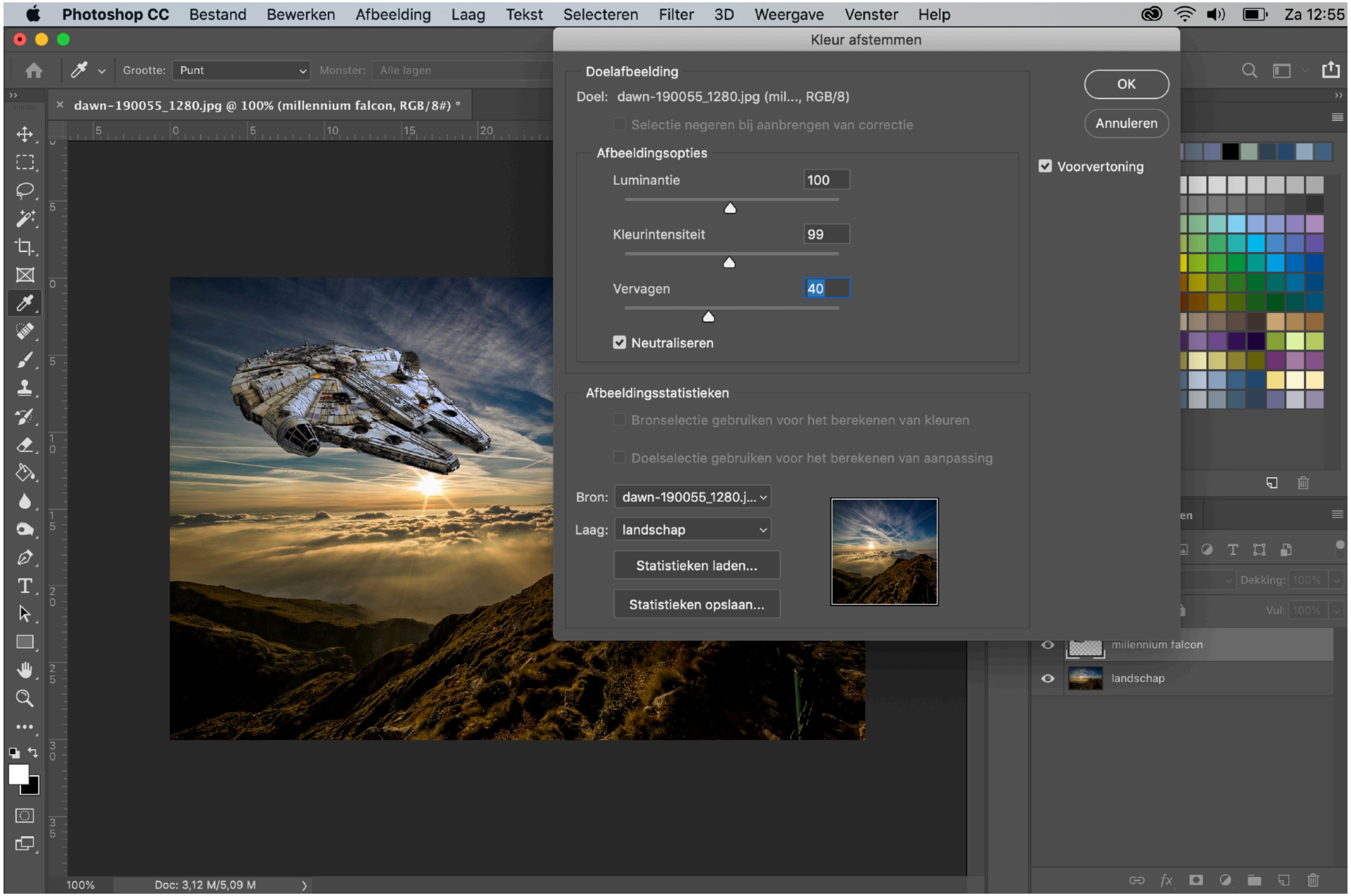
Een een beeld gedigitaliseerd, kan het makkelijk worden bewerkt. Elke pixel krijgt binnen een matrix een unieke positie op basis van zijn x- en y-waarde in die "tabel". Elke pixel heeft een kleurwaarde. In beeldbewerkingssoftware kan je op basis van de positie van elke pixel een bepaalde "actie" uitvoeren.

Een "gom" in Adobe Photoshop zal bijvoorbeeld de positie van de muiscursor vergelijken met die van de pixels waarboven de muis zich bevindt vervolgens de kleurwaarde van die pixels vervangen door "wit". Als je de "digitale gom" groter maakt, zal Photoshop niet alleen de pixel onder de muiscursor wit maken, maar ook alles in een straal daar rond.

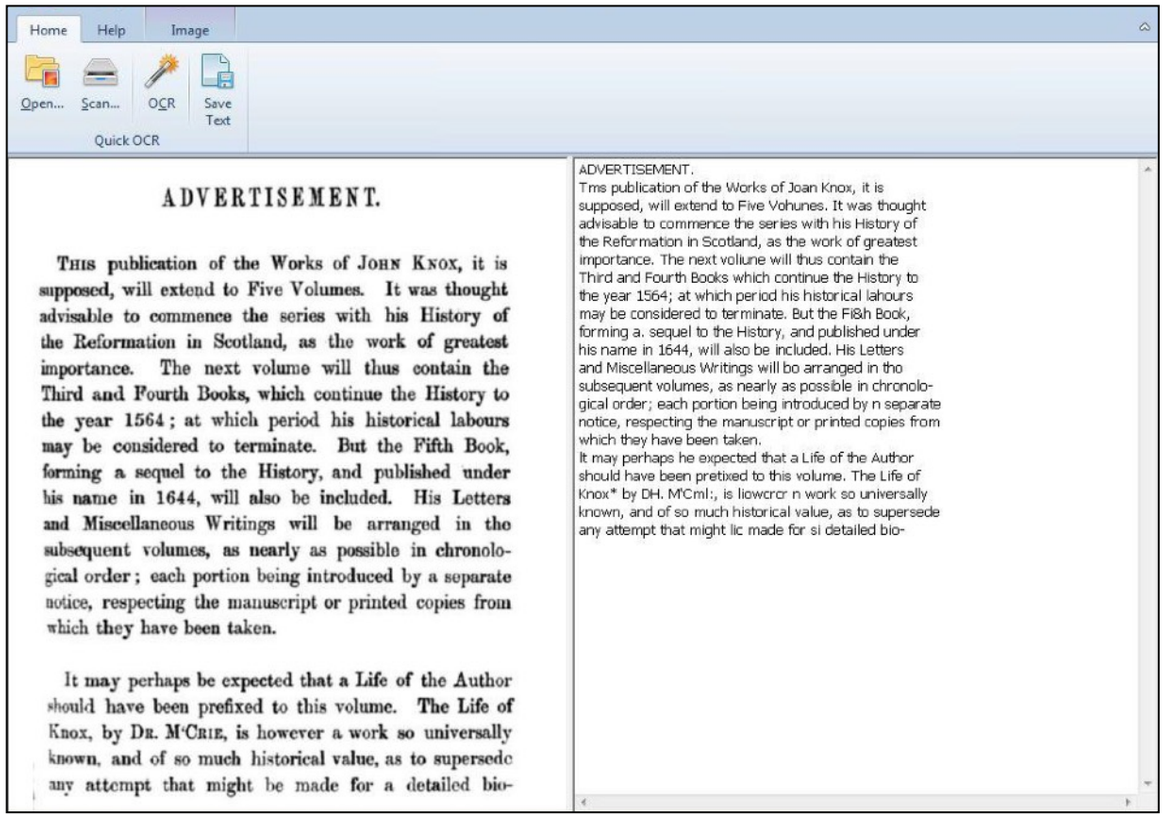
Daarnaast kan je ook op alle pixels tegelijk (of op een selectie daarvan) een bepaalde actie uitvoeren. Heel bekend zijn filtereffecten. Die kan je toepassen door de RGB-waarde van de pixels te wijzigen. Zo kan je bijvoorbeeld de waardes verhogen of verlagen om ze lichter of donkerder te maken. Of je kan met "**filters**" kleureffecten toepassen door bijvoorbeeld alle roodwaarden te vervangen door de waarde van de blauwwaarde enz. Immers, elke pixel heeft naast zijn x- en y-positie ook een waarde voor R, G en B. Door met die waardes te spelen, kan je allerlei effecten toepassen.

Professionele beeldbewerkingsprogramma's kunnen diverse virtuele **lagen** met pixels boven elkaar plaatsen. Dat laat het toe om de verschillende lagen los van elkaar te manipuleren. Zo kan je een afbeelding van een appel boven een laag met een fruitschaal leggen en er nog een extra laag met tekst aan toevoegen.

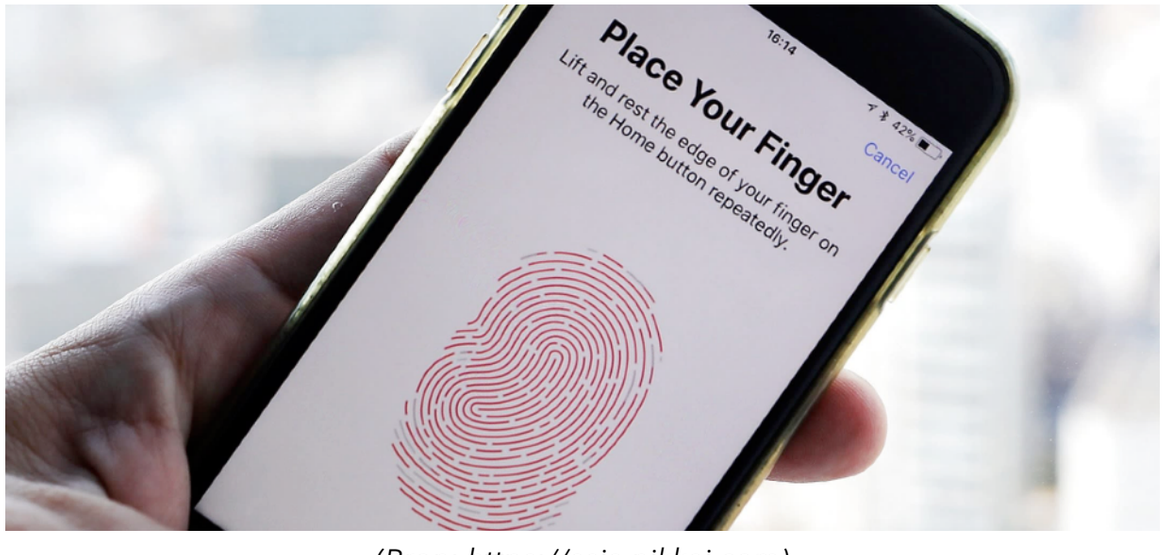
Het werken met lagen in Adobe Photoshop.



De kleuren van verschillende lagen op elkaar afstemmen in Adobe Photoshop.



Sjabloon-gebaseerde OCR.
(Bron: https://www.malavida.com/en/soft/supergeek-free-document-ocr/#gref)



(Bron: https://asia.nikkei.com)

Image recognition

Je smartphone beschikt doorgaans over minstens één camera, veel laptops zijn uitgerust met een webcam. Maar dit betekent nog niet dat die toestellen kunnen zien zoals mensen of dieren. Met onze ogen kunnen we niet enkel zien, maar we herkennen ook objecten. Willen we een computer informatie in beelden leren herkennen, dan spreken we van **computer vision** en **image recognition**.

In de warenhuizen "herkennen" scanners streepjescodes. De meeste smartphones herkennen informatie in QR-codes. Smartphones herkennen het gezicht of de vingerafdruk van de gebruiker. Om de ruwe beeldinformatie te herkennen, is software nodig. Net zoals het bij mensen niet de ogen zijn, maar de hersenen die objecten en mensen herkennen.

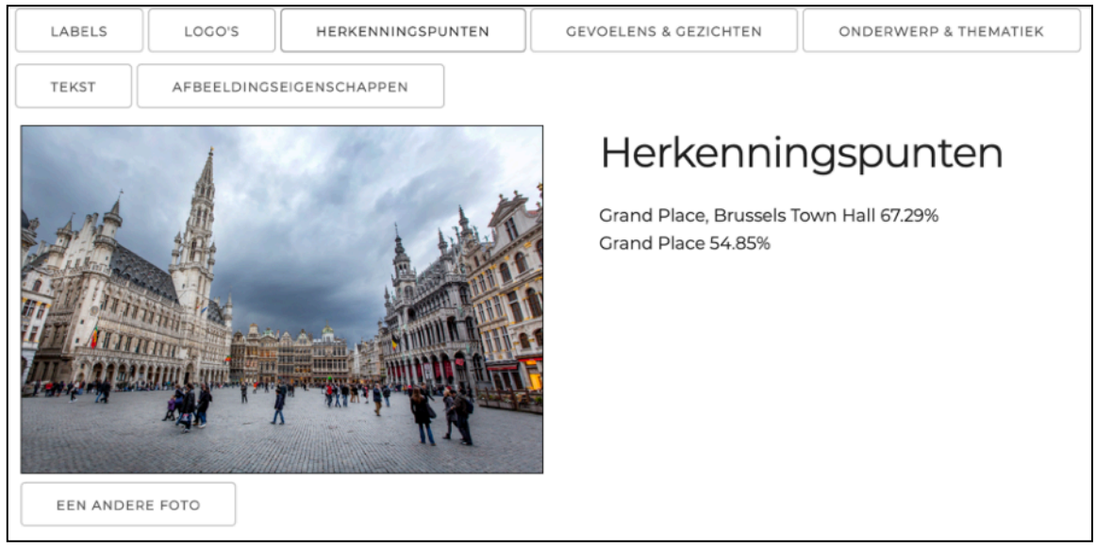
Streepjescodes, QR-codes en tekstherkenning zijn reeds relatief oude technieken. De software herleidt het beeld tot een zwart-witafbeelding. Bij **OCR (optical character recognition** vergelijkt de software de rasters van zwarte pixels met bestaande letters. Op die manier kan OCR-software teksten herkennen. Dit betekent echter niet dat de software snapt waarover de tekst gaat.

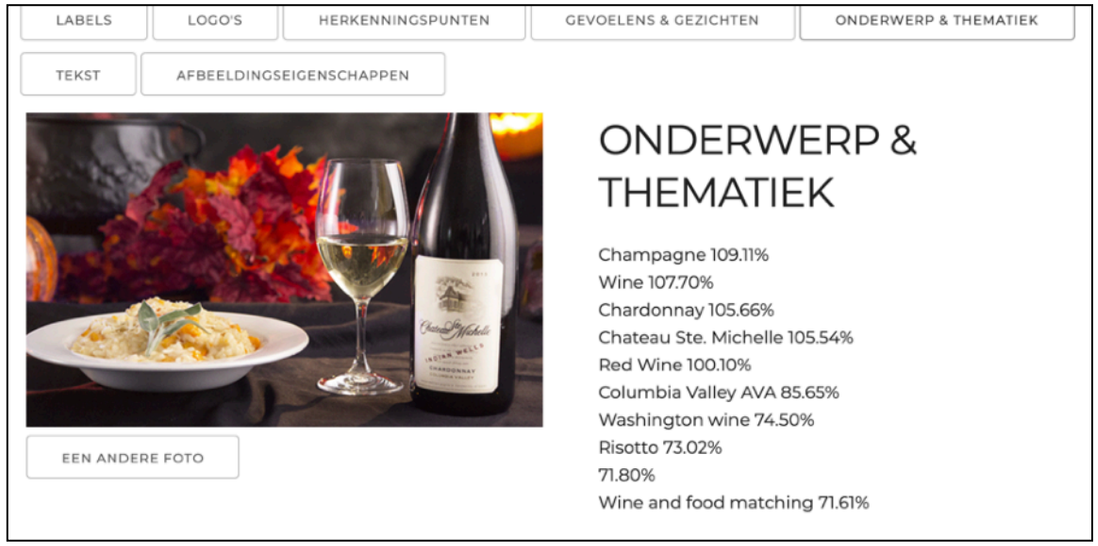
Om computers echt te leren herkennen wat op een afbeelding staat, gebruikt men **AI (kunstmatige intelligentie)**. Net zoals een klein kind de naam moet leren van alle objecten en levende wezens in de fysische wereld, moet AI-software stapsgewijs elementen leren herkennen. Als je een kind vaak genoeg bepaalde letters toont, zal het die letters leren kennen. Net zo gaat het met AI. Klassieke OCR kan geen letters herkennen die ondersteboven zijn afgedrukt, maar een kind of AI kan dit wel.

In het onderdeel over AI kom je te weten dat AI beelden kan herkennen door de werking van neuronen (in de hersenen) na te bootsen. Een "**neural network**" leert via vooraf gelabelde afbeeldingen, gelijkaardige afbeeldingen herkennen.

Als je een stuk software een hond wil leren herkennen, dan moet je de software eerst voeden met honderden of duizenden afbeeldingen van honden. Zo leert de software stapsgewijs dat zowel een Pekinees als een Pitbull honden zijn. Bovendien moet de software leren dat zowel Disney's Pluto als Lassie een hond is.

Enkel als een **robot** snel en accuraat objecten kan herkennen, kan hij (zij?) net zoals echte mensen "rondlopen". *Zelfrijdende auto's bijvoorbeeld gebruiken AI-gebaseerde computer vision en image recognition om voetgangers, verkeersborden, voertuigen… te herkennen.*





AI-gebaseerde computer vision. De software maakt gebruik van de Google Vision API.
(Bron: https://www.schoolvoorbeeld.be/nodig/cloudvision, Kris Merckx)

Kleuren afdrukken: CMYK

Een beeldscherm zendt licht uit en is standaard zwart (tenzij bij een e-bookreader). Papier is standaard wit en weerkaatst licht. Dit verschil zorgt ervoor dat je voor drukwerk een andere kleurcodering moet gebruiken.

Bij alles wat licht weerkaatst vb. professioneel drukwerk, tijdschriften... gebruikt je de **CMYK-kleurmodus**. CMYK zijn **subtractieve kleuren** (to subtract = aftrekken). Op papier druk je geen "wit", het papier is al wit. Alle andere kleuren krijg je door de vier basiskleuren C, M, Y en K (*c*yaan, *m*agenta, *y*ellow, bla*c*k of *k*ey) te gebruiken. RGB geeft samen wit. Bij CMYK is dit net het omgekeerde: hoe meer van een kleur, hoe donkerder.

C + M + Y + K = zwart

Alles wat je ontwerpt voor "druk", moet opgesteld zijn in CMYK-kleuren (we hebben het over "druk", niet over een copytheek of een eigen printer). U kan RGB- composities in bijvoorbeeld Photoshop eenvoudig converteren naar CMYK. Wanneer uw scherm echter niet goed is afgesteld (gekalibreerd) kan dit erg 'vreemde' kleuren opleveren op uw beeldscherm. Een beeldscherm zendt licht uit, het is immers niet bedoeld voor subtractieve, maar voor additieve kleurweergave.

De kleurmodi van een afbeelding in Adobe Photoshop.

Wat moet je je bij die CMYK-kleuren voorstellen bij additieve kleurmenging (dus op een beeldscherm?)

C	Cyaan	een combinatie van blauw en groen licht
M	Magenta	een combinatie van rood en blauw licht
Y	Yellow	een combinatie van rood en groen licht
K	Black / key	zwart, geen licht

Vectorafbeeldingen

Als je een foto maakt met je smartphone, dan bestaat die foto dus uit een aantal rijen en kolommen die samen pixels vormen. De kwaliteit van de afbeelding hangt af van de totale hoeveelheid pixels. Hoe meer pixels, hoe meer detail én hoe meer kwaliteit. Meer pixels betekent ook meer opslagruimte. Je kan nadien de resolutie (het aantal pixels van de afbeelding) wel verlagen, maar niet verhogen. Opgelet: technisch gezien kan je het aantal pixels wel verhogen, maar het zal zeker leiden tot kwaliteitsverlies. Je merkt het als je op een foto inzoomt. Digitale zoom betekent immers dat je de pixels vergroot. Maar software kan geen details toevoegen die al niet van bij het begin in de afbeelding stonden. Als je een foto maakt vanaf een festivalpodium waarbij je duizenden mensen ziet staan, kan je niet achteraf elk individueel gezicht uit die foto in hoge kwaliteit bewaren.

Soms is het echter noodzakelijk om een afbeelding in elk mogelijk formaat (elke mogelijke afmeting) te kunnen afdrukken. Een **logo** bijvoorbeeld moet je zonder kwaliteitsverlies op een gevel, vrachtwagen en visitekaartje kunnen afdrukken. In die gevallen gebruik je best geen **pixelafbeeldingen**, maar **vectorafbeeldingen**. Een vectorafbeelding bevat geen pixels, maar wiskundige vormen. Een blauwe cirkel bijvoorbeeld kan op die manier heel eenvoudig (met heel weinig code en dus ook heel weinig benodigde opslagruimte) omschreven worden.

In Processing bijvoorbeeld kan je heel eenvoudig een cirkel tekenen:

```
fill(0,0,255);
ellipse(200, 200, 100, 100);
```

De vulling van de cirkel wordt uitgedrukt in een RGB-waarde. De kleur van de cirkel is dus “blauw”. De instructie ellipse() bevat een viertal parameters. De eerste paar waarden behandelen de x- en de y-positie t.o.v. het programmavenster. De twee volgende waarden bepalen hoe groot de “horizontale” en de “verticale” straal moet zijn. Maak je die beide waarden gelijk, dan teken je een perfecte cirkel, maak je de ene waarde groter dan de andere, dan teken je een ellips.

Omdat enkel de coördinaten en de vormen en vullingen dienen opgeslagen te worden, blijven de bestanden uiterst klein. Een groot voordeel aan vectorafbeeldingen is dat zij oneindig kunnen geschaald worden zonder kwaliteitsverlies. Om de bovenstaande “ellips” te vergroten, moet je enkel de waardes aanpassen:

```
fill(0,0,255);
ellipse(200, 200, 10000, 10000);
```

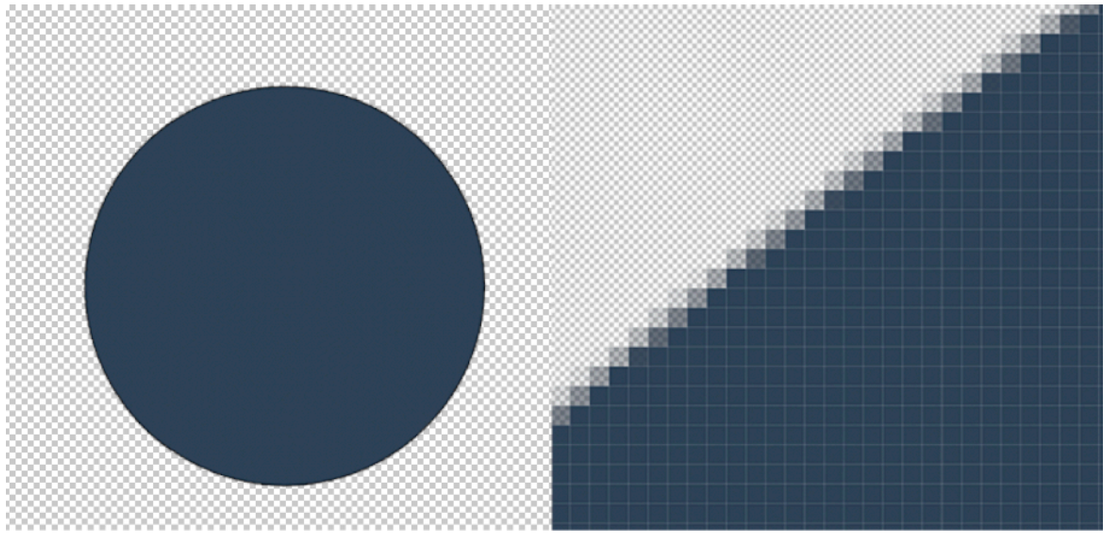
Zoomen op een pixelafbeelding smaakt de individuele pixels zichtbaar.

Lettertypes zijn over het algemeen eveneens vectorieel. Door deze voordelen zijn vectorafbeeldingen uitermate geschikt voor logo's, maar niet voor foto's.

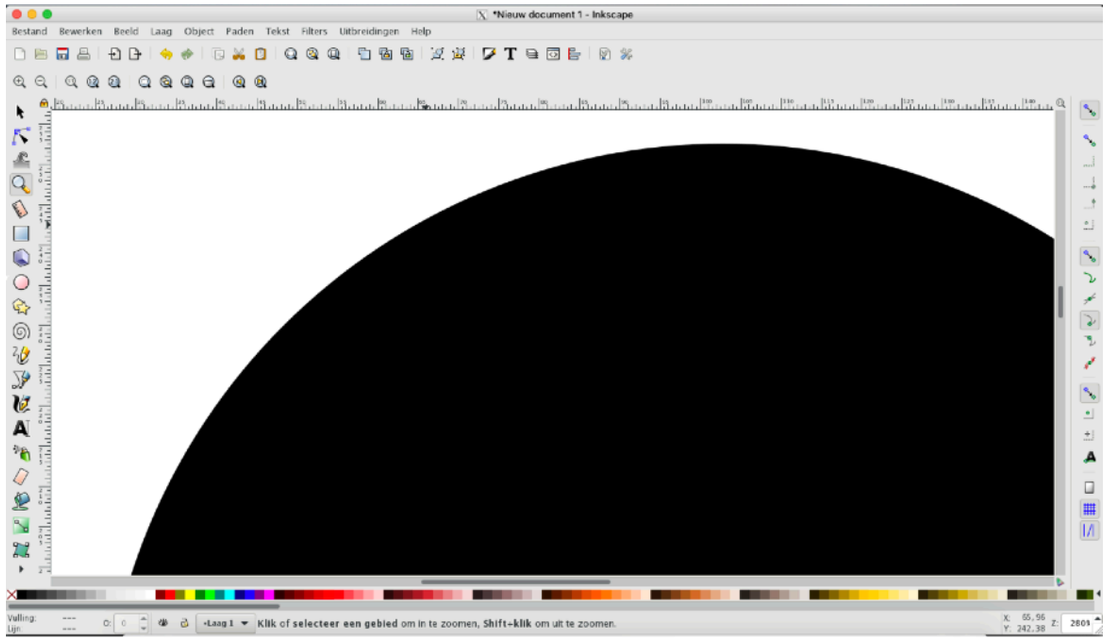
Vectortekenprogramma's:

- Adobe Illustrator (commercieel)
- CorelDraw (commercieel)
- Processing (open source, vectoranimatie)
- Inkscape (open source en gratis)

Terwijl je pixelafbeeldingen bewaart als JPG- of PNG-bestanden, bestaat er voor vectorafbeeldingen één belangrijk standaardformaat nl. **SVG (scalable vector graphics)**.



Een cirkel in Adobe Photoshop is geen echte cirkel. Dat merk je als je er op inzoomt.



Een vectortekening blijft altijd haar kwaliteit behouden, hoe sterk je ze ook uitvergroot (schermafbeelding van Inkscape).

DEEL 4 OPSLAG VAN DATA

- Hoe kan een computer informatie bewaren?
- Wat zijn databanken?
- …

TEKSTBESTANDEN

Sorry, je begrijpt het verkeerd. Met tekstbestanden bedoelen we in dit geval niet de documenten die je intikt in MS Word of LibreOffice. Uiteraard hebben ze wel wat elkaar gemeen. Een bestand uit MS Word is een teksverwerkerbestanxd. Als je het bewaart op je harde schijf, wordt het een binair of een gecomprimeerd bestand.

Als we het over tekstbestanden hebben in de titel, dan doelen we op “platte tekst”-bestanden. Bestanden die enkel karakters bevatten die je op een toetsenbord kan intikken, maar geen opmaak of afbeeldingen.

Als multimedia-expert in spe is het van bijzonder groot belang dat je weet wat we met platte tekst bedoelen. Als webontwikkelaar of programmeur zal je sowieso in teksteditors met platte tekstbestanden werken. Wat dit precies is, lees je zo meteen…

ASCII

De computer bewaart alle bestanden binair, als een reeks nullen en enen, want dat is de enige code die de meeste moderne computers verstaan. Elk bestand bestaat uit een vaak erg lange, maar toch eindige reeks nullen en enen. De computer bewaart ze op een opslagmedium (harde schijf, USB-stick…).

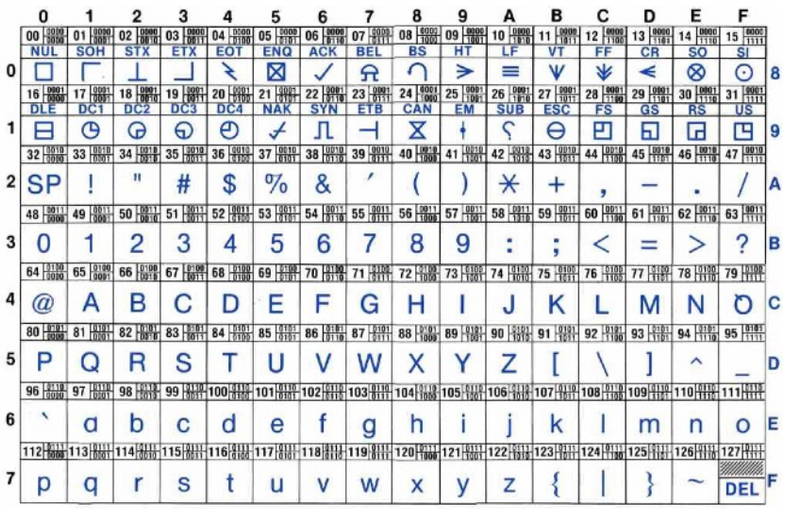
Terwijl ik deze zin intik op mijn toetsenbord, vertaalt de computer de toetsaanslagen in nullen en enen, in bits die op hun beurt gegroepeerd zijn in bytes. Eén byte is een "groepje" van acht bits, de letter a vertaalt de computer bijvoorbeeld als de byte 01100001. In decimale getallen is dit 97. Elke toets en ook combinatie van toetsen (bijvoorbeeld SHIFT + A voor de hoofdletter A) krijgt zo een byte toegewezen. Vermits één byte uit 8 bits bestaat, beschikken we over 28 combinaties (2 tot de achtste macht), want elke bit kan ofwel 1 ofwel 0 zijn.

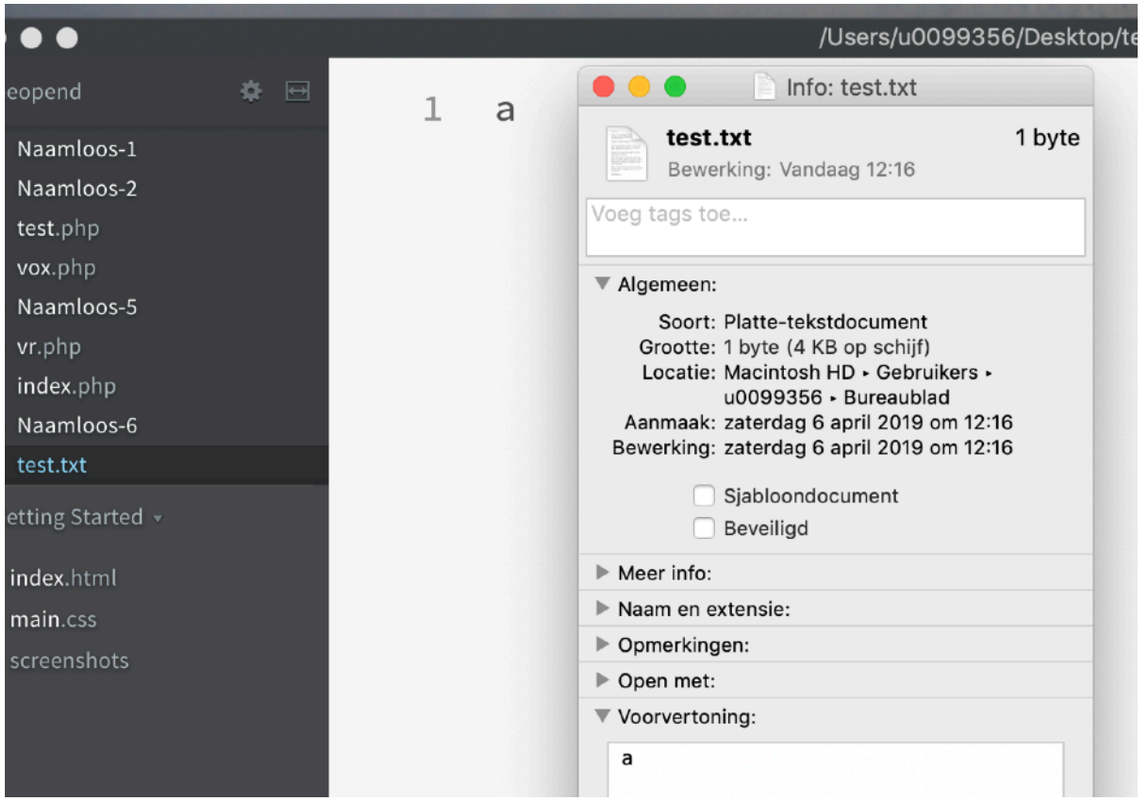
In totaal kan je op die manier 256 combinaties vormen, of alle combinaties tussen

```
00000000 = 0 en 11111111 = 255
```

Hiervoor zijn natuurlijk afspraken gemaakt. In principe kan je om het even welk toetsenbord op om het even welk digitaal systeem dat tekstinvoer toelaat, aansluiten. De toetsaanslagen worden automatisch herkend. Dit komt omdat ze zo goed als allemaal ASCII ondersteunen. ASCII (American Standard Code for Information Interchange) is een standaard om Latijnse letters, cijfers, leestekens en een aantal andere tekens en een reeks stuurcodes te koppelen aan een getal.

De ASCII-tabel is een tweedimensionale rij of tabel. Zoek het gewenste teken en neem de binaire code die in de kop van de rij is aangeduid. Plak daarachter vervolgens de inhoud van de kolomkop. Een spatie bestaat dus uit 0010 gevolgd door 0000 of samengevoegd 00100000, wat overeenkomt met het decimale getal 32.





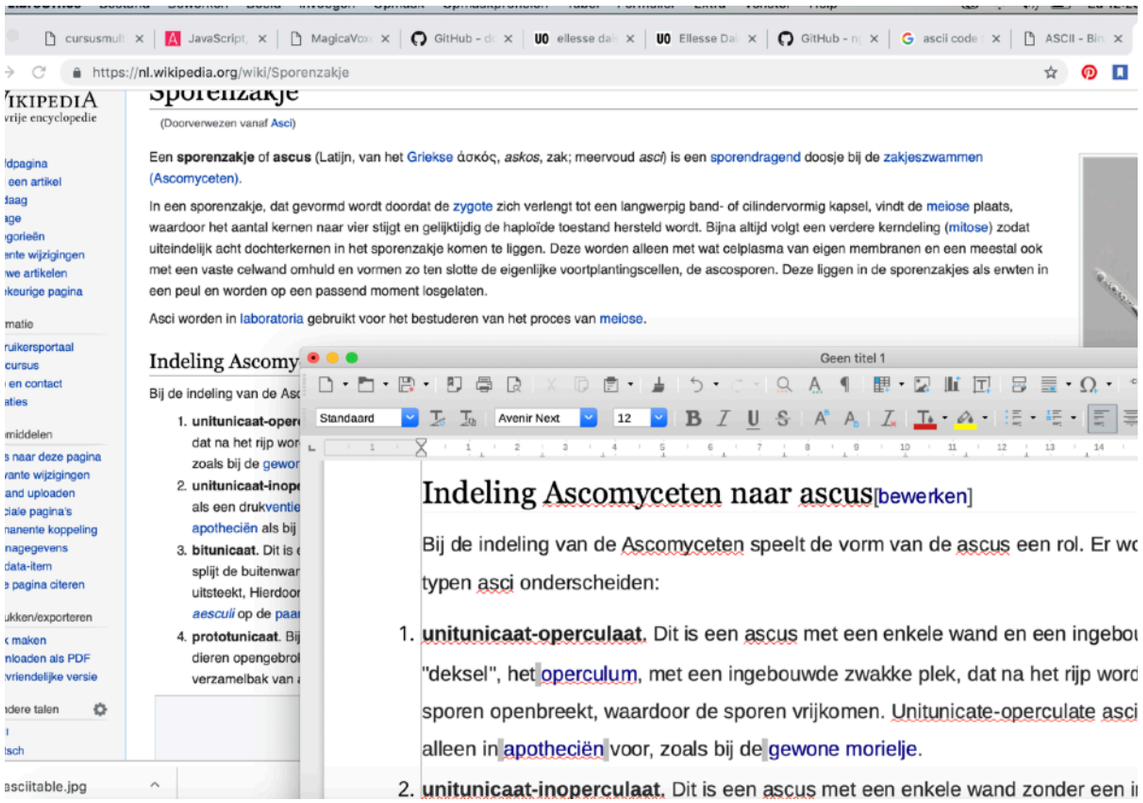
Test: bewaar eens een bestand met een teksteditor (zoals bijvoorbeeld Kladblok) en tik er welgeteld één teken in. Kijk vervolgens naar de bestandsgrootte (zie afbeelding links).

In een teksteditor kan je enkel ASCII-tekens (of UTF-8-karakters, waarover later meer) invoeren. Je kan er je tekst niet mee opmaken of er bijvoorbeeld foto's aan toevoegen. Ik hoor je al denken: wat is daarvan het nut?

Het grote voordeel is dat dit soort bestanden perfect uitwisselbaar zijn. Je hebt er geen speciale software voor nodig om de inhoud van die documenten te kunnen lezen. Een eenvoudige editor zoals Windows Kladblok of Adobe Brackets volstaan om het bestand te openen. Programmeercode of webpagina's kan je perfect in een editor openen, omdat ze enkel karakters kunnen opnemen die ondersteund worden door de ASCII of UTF-standaarden.

Afbeelding: een tekst in de editor Adobe Brackets.

Je hebt het misschien al meegemaakt. Je kopieert een tekst van Wikipedia en je plakt het in een MS Worddocument. Ook alle links die in de webpagina zaten en de opmaak, zijn mee gekopieerd. Wil je dit vermijden, dan plak je de tekst eerst in een teksteditor. Op die manier heb je enkel de “platte tekst”.



Als je een tekst van een webpagina kopieert en plakt in een tekstverwerker, kopieer je ook heel de opmaak mee.

UNICODE en UTF-8

ASCII werkt zoals je merkt perfect, maar pakweg Arabieren, Chinezen en Japanners, kortom iedereen die geen standaard Europees georiënteerde taal schrijft, bleef met ASCII wel aardig op zijn honger zitten. Er bestonden geen binaire codes voor het Arabisch of Chinees (een Chinese Jan met de pet kent ongeveer 7000 verschillende karakters). Uitbreidingen op de de ASCII-tekenset waren noodzakelijk. De noodzaak om alle andere taalsystemen en codes op te nemen leidde tot UNICODE.

De UNICODE-tekenset maakt het gebruik van alle mogelijke denkbare alfabetten en schriftsystemen in één document mogelijk. Niet dat je dit vaak nodig hebt, want je schrijft of tikt niet alle dagen in het Pau Cin Hau, het Bamum of Sharada, maar het gebruik van dingbats, wiskundige symbolen enz. kom je al wat vaker tegen. UNICODE is een ISO-standaard.

De meest bekende tekenset van UNICODE is UTF-8 (8 bit Unicode Transformation Format). Moderne webpagina's zijn volgens deze norm gecodeerd waardoor ze om het even welk symbool standaard kunnen weergeven.

BINAIRE BESTANDEN

Ook al bieden ASCII-bestanden heel wat voordelen, toch is het niet altijd de beste manier om bestanden te bewaren.

Wanneer men het over BINAIRE bestanden heeft, dan gaat het over bestanden die niet volgens de ASCII-tekenset zijn gecodeerd. En er bestaan meerdere goede redenen om dit niet te doen.

Voor het bewaren van de meeste bestanden zoals foto's, documenten uit een tekstverwerker, presentaties, films… gebruikt men een "binaire codering". ASCII zou in dat geval leiden tot te grote bestanden.

Heel vaak gebruikt men bovendien compressietechnieken om bestanden een stuk kleiner te maken zodat ze minder ruimte in beslag nemen. Denk maar eens hoe snel je smartphone vol is, als je alle dagen foto's of filmpjes maakt.

Waarom geen ASCII?

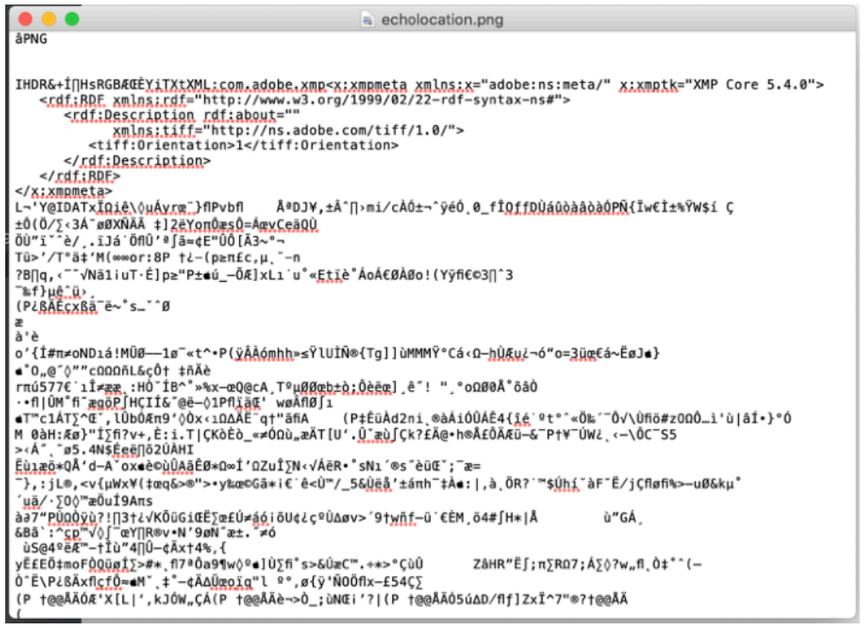
Een voorbeeld: een letter “a” (niet de hoofdletter) op je toetsenbord vertaalt zich als de byte 01100001. Eén teken wordt vervangen door een reeks van 8 nullen en/of enen. Een computer gebruikt in dit geval dus 8 bits waar een mens slechts één teken nodig heeft. Dit geldt ook voor de cijfers op je toetsenbord. Tik je een 0 dan wordt dit volgens de ASCII-regels 00110000 en een 1 wordt 00110001. Het getal 256 behandelt hij volgens de ASCII-tekenset als 3 afzonderlijke ASCII-karakters of 3 bytes en ziet er dan als volgt uit: 00110010 00110101 00110110.

Een nogal gekke manier van doen als we enkel getallen willen bewaren, want intern maakt ASCII reeds de vertaalslag naar decimale cijfers door aan elke toets of karakter een decimaal getal te koppelen.

Getallen kunnen veel eenvoudiger bewaard worden door ze niet als een reeks afzonderlijke karakters, maar als één geheel, als een echt getal te bewaren. 256 zal er dan uitzien als 00000001 00000000.

Eén enkele byte 256 verschillende waardes bevatten. Gaan we naar 4 bytes dan krijgen we 4 miljard mogelijke combinaties (232). Het getal 4000000000 (4 miljard) zou op die manier slechts 4 bytes in beslag nemen. Bewaren we het als ASCII, dan zou het 10 bytes aan ruimte oppeuzelen.

Om die reden gebruikt men heel wat vormen van codering. Het zou ronduit dom zijn om bijvoorbeeld een afbeelding als een reeks ASCII-waardes (weliswaar binair gecodeerd) te bewaren. Een afbeelding van 1024 bij 768 pixels telt in totaal 786432 pixels. Stel dat elke pixel zou kunnen bestaan uit een mix van 256 waardes rood, 256 waardes groen en 256 waardes blauw, dan zou het gecodeerd volgens de ASCII-tekenset onwaarschijnlijk veel ruimte in beslag nemen.



Een binair bestand (PNG-afbeelding) in een teksteditor.

BESTANDEN VERKLEINEN EN CONVERTEREN

Je weet al waarom niet alle bestanden worden bewaard in ASCII-formaat. Soms is het nuttige om bestanden binair te coderen zodat ze minder ruimte in beslag nemen. Sommigen bestanden, afbeeldingen en films bijvoorbeeld, nemen echter uit zichzelf al veel plaats in. Een afbeelding kan immers bestaan uit miljoenen pixels. Elk van die pixels bevat een 8-bits waarde voor rood, groen en blauw. Voor een eenvoudige afbeelding moet er dus al snel heel wat informatie bewaard worden. De hoeveelheid opslagruimte neemt bovendien nog toe naargelang de resolutie van de afbeeldingen. Hoe vaak krijg je niet te maken met een gebrek aan opslagruimte bij het maken van foto's met een smartphone? De meeste afbeeldingsformaten passen daarom algoritmes toe om de hoeveelheid data te verkleinen.

Ook om bestanden te mailen, kan het handig zijn ze eerst te verkleinen. Dat kan, zelfs zonder kwaliteitsverlies.

Afbeeldingen comprimeren

Afbeeldingen nemen veel plaats in door verschillende redenen:

- Hoeveelheid pixels (resolutie).
- Aantal verschillende kleuren voor RGB.

Als je één van die twee vermindert, verkleint eveneens de benodigde opslagruimte.

1. Je kan het aantal pixels in een afbeelding reduceren. Met beeldbewerkingssoftware kan je het aantal pixels verlagen. Zo kan je een afbeelding van 1920 pixels breedte, verlagen naar bijvoorbeeld 300 pixels. Deze techniek wordt vaak toegepast bij het klaarstomen van afbeeldingen voor het gebruik op internet. Ook het aantal pixels per inch kan verlaagd worden (dpi of dots per inch). Voor drukwerk heb je afbeeldingen nodig van 300 dpi, voor internet volstaan afbeeldingen met een resolutie van 72 dpi. Het aantal pixels of dpi verhogen is eveneens mogelijk, maar dit betekent niet dat de kwaliteit van de afbeelding groter wordt. De software voegt dan kunstmatig pixels toe door bepaalde pixels te verdubbelen. Verloren informatie kan echter niet worden "toegevoegd". In politieseries op TV zie je bijvoorbeeld hoe men een "duidelijk" beeld tevoorschijn tovert uit een lageresolutiebeeld van een digitale camera. Dit kan in feite niet.

2. Je zou het aantal beschikbare kleuren kunnen verminderen. Blj het RGB-kleursysteem kan elke pixel ingekleurd worden met 265 waardes voor rood, 256 voor groen en 256 voor blauw. Dit betekent dat je voor elke pixel over 16 777 216 verschillende combinaties beschikt, want 256 * 256 * 256 = 16 777 216.

Je zou kunnen afspreken dat je niet alle combinaties mogelijk maakt. Dat is precies wat het GIF-formaat doet. Een GIF-afbeelding laat je enkel toe om in totaal 256 verschillende combinaties te gebruiken. Dit zorgt ervoor dat de afbeelding een pak kleiner wordt. Immers, je kan niet meer kiezen uit een "verfwinkel" met 16,7 miljoen verschillende kleuren. In de GIF-"verfwinkel" kan je enkel kiezen uit 256 verschillende kleuren.

3. Als je twee afbeelding met even veel pixels met elkaar vergelijkt, zal je merken dat ze meestal niet even veel ruimte in beslag nemen. Immers, niet alle afbeeldingen bevatten even veel kleuren.

foto van een grasveld bevat minder kleurvariatie dan een foto van een schilderij van Pieter Paul Rubens.

Het aantal pixels reduceren in Adobe Photoshop.

4. Het menselijk oog is gevoeliger voor helderheid dan voor kleur. Door hier op in te spelen, kan je de bestandsgrootte van een afbeelding verkleinen. Een JPEG-afbeelding behoudt de helderheid van de pixels, maar laat heel wat kleurvariatie weg. Simpel voorgesteld: een jpeg verdeelt een afbeelding in een raster waarbij gemiddelde waardes worden gemeten. Veel variatie in kleur wordt weggelaten. Hoe meer verschillen weggelaten worden, hoe groter de compressie.

5. Men past in de meeste gevallen dis wiskundige algoritmes toe om afbeeldingen te comprimeren. Afbeeldingsbestanden kunnen op diverse manieren bewaard worden. TIFF, jpg, gif, png... zijn bekende bestandsformaten voor pixelafbeeldingen. Ze hebben allemaal hun specifieke doeleinden, maar eveneens voor- en nadelen. Een TIFF-afbeelding kan je zonder compressie bewaren, d.w.z. dat alle pixelinformatie en alle informatie over de kleuren behouden blijft in het bestand. Uiteraard zorgt dit ervoor dat een TIFF-bestand veel meer ruimte inneemt op een opslagmedium.

Bij andere afbeeldingsformaten past de gebruikte digitale camera of software een compressiealgoritme toe. In zo goed als alle gevallen betekent dit dat bepaalde pixelinformatie verloren gaat en nadien niet meer kan worden toegevoegd.

Opties voor JPEG-compressie in Adobe Photoshop.

Tekstinformatie comprimeren

Niet elke vorm van compressie leidt tot dataverlies. Heel wat compressie-algoritmes werken verliesloos (lossless). Neem bijvoorbeeld de volgende zin:

"Ask not what your country can do for you -- ask what you can do for your country."

Die zin kan je eenvoudig comprimeren door een lijst op te stellen van alle woorden of tekstfragmenten die meer dan één keer voorkomen.

Eerst maken we een genummerde lijst van alle woorden of fragmenten die meerdere keren voorkomen. Vervolgens vervangen we in de originele zin/tekst de woorden/fragmenten door hun corresponderend cijfer in de lijst.

```
1. ask
2. what you
3. can do for you
4. country
5. --
```

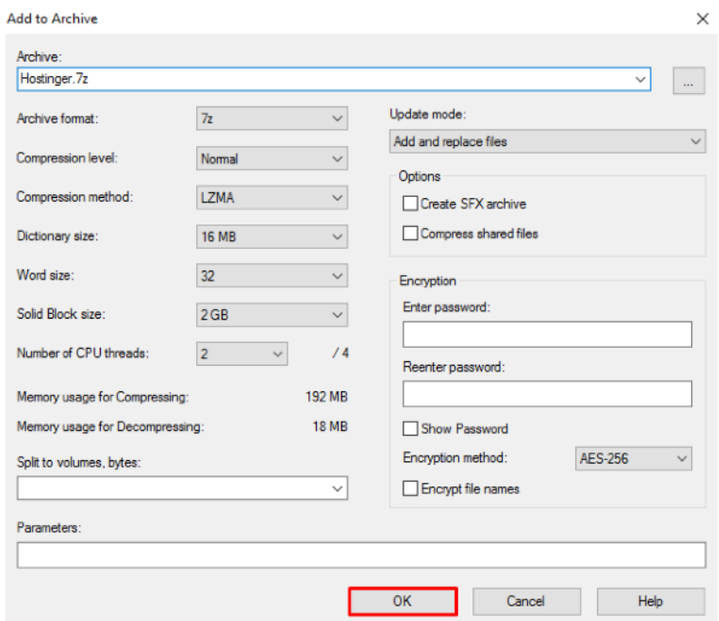
De gecomprimeerde zin ziet er nu uit als volgt:

```
"1 not 2r 4 3 5 1 2 3r 4."
```

Wil je het originele bestand opnieuw herstellen (**decomprimeren**), dan moet je de omgekeerde beweging doen.

Uiteraard zal ons voorbeeld niet veel data uitsparen, want je moet ook het lijstje met woorden bewaren. Maar bij lange tekstbestanden, kan op die manier heel veel "ruimte" uitgespaard worden. Een bekende **compressietechniek** die hiervan gebruik maakt, is **ZIP**.

Een ander voordeel van het ZIP-formaat is dat je er meerdere bestanden tegelijk mee kan comprimeren in één enkel ZIP-bestand. Dit vergemakkelijkt het uitwisselen van ganse mappen met bestanden.



Het gratis en open source 7zip ondersteunt diverse compressietechnieken.

Geluid comprimeren

Ook bij geluidsopname kan een algoritme het geluid digitaal comprimeren:

- Het menselijk gehoor (dat bijzonder goed is), kan bepaalde geluiden niet waarnemen.
- Sommige geluiden kunnen beter waargenomen worden dan andere.
- Wanneer twee geluiden tegelijkertijd hoorbaar zijn, horen we enkel het luidste.

Mp3 maakt de benodigde opslagcapaciteit voor het bewaren van gedigitaliseerde audiogegevens beduidend kleiner. Een mens hoort tijdens het beluisteren van muziek niet alles. Wanneer twee luide tonen heel dicht tegen elkaar liggen, registreert hij enkel bewust de luidste toon. Het mp3-algoritme filtert de iets lagere toon er gewoon uit. Het heeft ook weinig zin om tonen te bewaren die hoger of lager zijn dan wat het menselijk gehoor kan waarnemen. mp3 sluit ook klassieke compressietechnieken in zoals joint stereo, waarbij audiogegevens die in het linker- en rechterkanaal voorkomen slechts één keer worden opgeslagen. Deze techniek heeft echter wel wat nadelen: mp3 en jpg (jpeg) zijn 'lossy'. Dit betekent dat de gegevens die tijdens de compressie worden weggelaten, niet meer kunnen worden teruggehaald.

Film comprimeren

Bij analoge film (25 beelden per seconde) wordt elk beeldje afzonderlijk opgeslagen, ook al richt je je camera een uur lang op hetzelfde schilderij. Bij digitale compressie gaat de software op zoek naar patronen in de opeenvolgende beelden. Enkel die beeldfragmenten van het beeld die werkelijk veranderen, worden opgeslagen. Maar ook daar kan compressie gebeuren. Stel je voor dat je een landschap filmt vanaf een vast statief. Door het landschap rijdt een auto. Het algoritme zal enkel die onderdelen van het landschap die wijzigen onthouden, maar de pixels van de auto blijven ondanks hun veranderde positie in het beeld vrijwel ongewijzigd. Dus: wanneer je je camera een uur lang op hetzelfde schilderij heeft gericht, zal de film na compressie bijzonder klein zijn. De software vergelijkt voortdurend opeenvolgende beelden. Als er geen verschil tussen twee opeenvolgende beelden zit, zal het algoritme geen nieuwe informatie meer opslaan.

DATA REDDEN OF VERLIEZEN

Ben je al eens wat data kwijtgeraakt? Je smartphone is om zeep en je bent al je foto's kwijt. Of je maakt een bestand op één computer en je kan het bestand op een ander toestel niet openen omdat je daar niet beschikt over de benodigde software.

Zullen we de bestanden die we nu maken, over pakweg 10 of 20 jaar nog kunnen openen?

We maken massa's foto's en films, meer dan ooit tevoren, maar kan je in die gigantische hoeveelheid bestanden net die foto van je kleine zusje terugvinden van net 5 jaar geleden?

Rotte gegevens en dataverlies

Nauwelijks 25 jaar geleden maakte ik mijn teksten in de tekstverwerker WordPerfect onder het besturingssysteem Windows 3.11. Omdat ik bang was gegevens kwijt te spelen, bewaarde ik steevast een kopie (backup) op een kleine diskette. Gelukkig bleven op die manier mijn kostbare data bewaard voor het nageslacht (ook al zit er vermoedelijk niemand op te wachten)! Vandaag heb ik problemen met die gegevens. In de eerste plaats is het besturingssysteem al lang naar het softwarekerkhof verhuisd. Daarnaast kan zelfs de nieuwste versie van WordPerfect de bestanden van zijn verre voorloper niet meer correct openen. Omdat ik wel wat IT-ervaring heb (denk ik toch van mezelf) en de moed niet snel opgeef, blijf ik proberen. Uiteindelijk gaan die oude bestanden wel weer open, maar de lay-out ziet er niet meer uit. Kolommen zijn verschoven en de software meldt problemen met ontbrekende of onbekende lettertypes. Bovendien moet ik hiervoor een ietwat oudere computer met een diskettestation van de zolder halen, want de moderne generatie computers beschikt al lang niet meer over de benodigde uitleesapparatuur.

Ook bedrijven botsen regelmatig tegen dit probleem aan. De firma's achter de **uitleesapparatuur** en/of de **uitleessoftware** bestaan vaak al lang niet meer en/of de gegevens kunnen niet meer worden geopend. Vaarwel informatie en sorry voor al dat dure en tijdrovende invoeren van informatie. De harde schijf met magnetische opslag lijkt de reddende engel, maar ook daar stuiten we op problemen. De manier waarop de bestanden op de harde schijf worden opgeslagen, hangt af van de **gebruikte software en het besturingssysteem**. Een harde schijf die is 'ingedeeld' door het systeem Windows, kan je niet zonder problemen uitlezen en gebruiken onder bijvoorbeeld Linux of Mac os en omgekeerd. Uiteraard bestaan er altijd wel oplossingen voor dit soort problemen, maar je moet ze natuurlijk ook kennen, kunnen hanteren en goedkoop zijn ze niet altijd.

Een **crash van de software of het besturingssysteem** of een **technisch mankement** aan de schijf kan eveneens leiden tot een onherroepelijk verlies van data. Tot slot is geen enkel optisch of magnetisch opslagmedium het eeuwige leven beschoren. Ook zij verliezen stukje bij beetje de opgeslagen informatie onder invloed van allerlei fysische omstandigheden.

De technische term voor al deze vormen van verlies van digitale gegevens heet '**bit rot**', ook al moet je hierbij natuurlijk niet meteen denken aan schimmels.

Digitalisering biedt op zich een zeer robuuste vorm van opslag, maar vaak is '**miniaturisering**' de boosdoener. Eén bit wordt vastgehouden door eenpaar honderd elektronen. Hoe kleiner (hoe minder elektronen per bit), hoe fragieler de informatie wordt. Alle voordelen van digitale opslag ten spijt, zit er dus ook een keerzijde aan de medaille. De *moderne bibliotheek van Alexandrië* vergaat dus niet door een brand, maar van binnenuit, als door een klein 'vuurtje' dat de data van binnenuit kan opvreten.

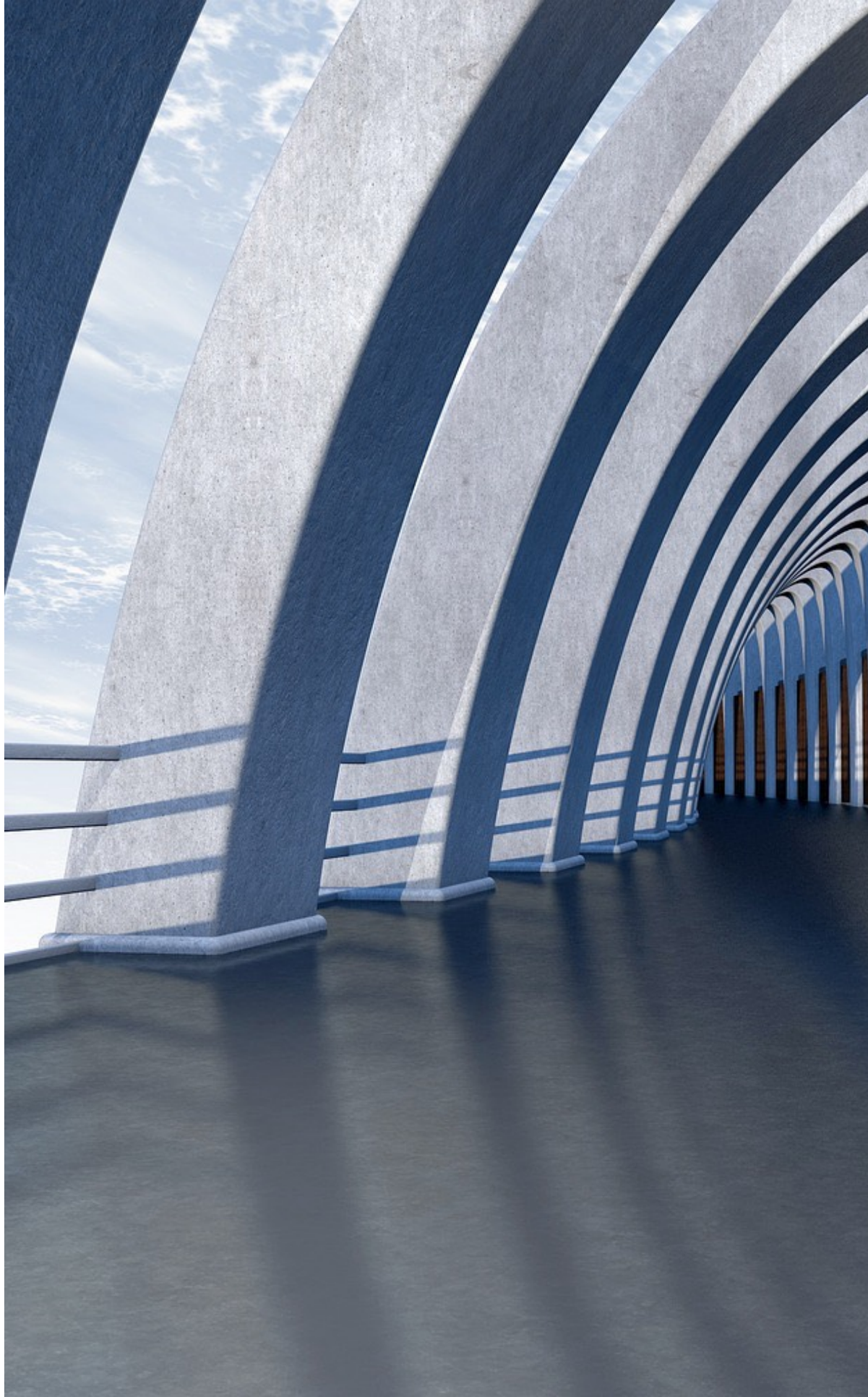
En de toekomst, wat brengt die? Wat gebeurt er met alle data die jij gisteren, vandaag, morgen... invoer(t)(de) in **digitale leeromgevingen, webapplicaties, sociale media**? Blijven die data bewaard? Gaan ze een eigen leven leiden? Heb je er zelf nog weet van of toegang toe?

'Obsolete', buiten gebruik geraakte, digitale media (ponskaarten, magnetische opslag, optische opslag…).

SEMANTISCHE EN GESTRUCTUREERDE DATA

HTML, XML, JSON…

Leerstof tweede en derde jaar (nog niet beschikbaar)



BESTANDSSYSTEMEN

Hoe bewaart een operating system zoals Android, iOS, Mac OS X, Linux, Windows… bestanden op een harde schijf of een SSD-geheugen? Zelfs al bewaar je bestanden in de “cloud”, dan nog staan ze *ergens* op een opslagmedium?

Hoe gaat dit in zijn werk? Hoe kan een computer zoveel gegevens op een minuscuul schijfje krijgen? Hoe kan een digitale camera uren filmmateriaal op een verwaarloosbaar klein SD-kaartje opslaan?

Daar komt geen magie aan te pas, maar verbazend is en blijft het natuurlijk wel. Waarom gaan opslagmedia soms gewoon stuk? Kan je de gegevens dan nog redden

Bestanden een plaats geven

We tasten nog grotendeels in het duister over hoe die massa neuronen die in ons hoofd schuilt, herinneringen vasthoudt. We begrijpen nog niet volkomen hoe elektrische pulsen herinneringen vastleggen en hoe nadien gegevens opnieuw worden geadresseerd en opgeroepen.

Voor computers hebben we bestandssystemen aangelegd, waardoor het bewaren en opzoeken van gegevens wordt vergemakkelijkt. Losse bestanden, bijvoorbeeld teksten, audio, film en foto's bewaren computers op elektromagnetische opslagmedia, zoals magneetbanden en harde schijven. Optische media zoals cd's en dvd's vormen het moderne equivalent van de ponskaarten, waar patronen van putjes de binaire codes voorstellen. Een laserstraal leest de putjes tegen een onwaarschijnlijke snelheid uit en reconstrueert de originele bestanden. Gestructureerde data bewaren we in relationele databanken.

Maar hoe krijgen al die bestanden en mappen een plaatje op een opslagmedium en hoe vindt een digitaal bestand al die brokjes binaire informatie terug? M.a.w. er is een systeem nodig om

1. al die data een plaats te geven
2. en die plaats snel terug te vinden.

File system en formatteren

We hebben het over een bestandssysteem of FILE SYSTEM. Het zou handig zijn indien hiervoor één standaard zou bestaan, maar die is er niet, omdat diverse leveranciers en ontwikkelaars hun eigen systeem ontwikkelden. Niet elk systeem is bovendien voor elke vorm van opslag even geschikt én de evolutie van de hardware schept vaak ook nieuwe mogelijkheden die er vroeger niet waren.

Wie een Windowscomputer gebruikt en het meemaakte dat de computer crashte, heeft wellicht ooit al eens gehoord over het begrip FORMATTEREN. **Formatteren is een proces waarbij het besturingssysteem of een stuk andere software, de harde schijf of het opslagmedium gaat indelen.** Zo'n beetje zoals een architect op een plan een huis indeelt in kamers. Trouwens: formatteren blijft niet beperkt tot MS Windows, ook een Apple- of Linuxcomputer moet zijn harde schijf of opslagmedium indelen om er bestanden op te kunnen bewaren.

Een filesystem vormt een virtueel hotel voor data

We maken even de vergelijking met een architect die een hotel uittekent. De architect tekent verschillende verdiepingen (een harde schijf bestaat eveneens uit verschillende schijfjes bovenop elkaar) die hij indeelt in kamers (blocks). Elke kamer is exact even groot en adresseerbaar met een kamernummer. Elke hotelgast komt in zo een kamertje terecht. Soms komt een gans gezin op bezoek en is één zo'n kamertje te klein. Dan moet de hotelbaas (het besturingssysteem) meerdere kamers (blocks) toewijzen. Dat kan een aanpalende kamer zijn, maar misschien zijn alle naburige kamers reeds bezet door andere gasten (bestanden) en verhuisd een deel van het gezin naar een verderaf gelegen kamer of eentje op een ander verdiep. Grote gezinnen (bestanden) kunnen op die manier over meerdere kamers en/of verdiepingen verspreid zitten. De hotelbaas had de architect ook de opdracht kunnen geven om een aantal grotere kamers te voorzien, maar dat zou betekenen dat sommige kamers maar voor een stuk zouden bezet zijn. De hotelbaas houdt een lijst bij van alle kamers en de aanwezige gasten. Die lijst moet hij voortdurend aanpassen. Soms vertrekt een hotelgast (verwijderen van een bestand), soms brengt hij de dag nadien een vriendin of losse scharrel mee (bestand wordt groter)...

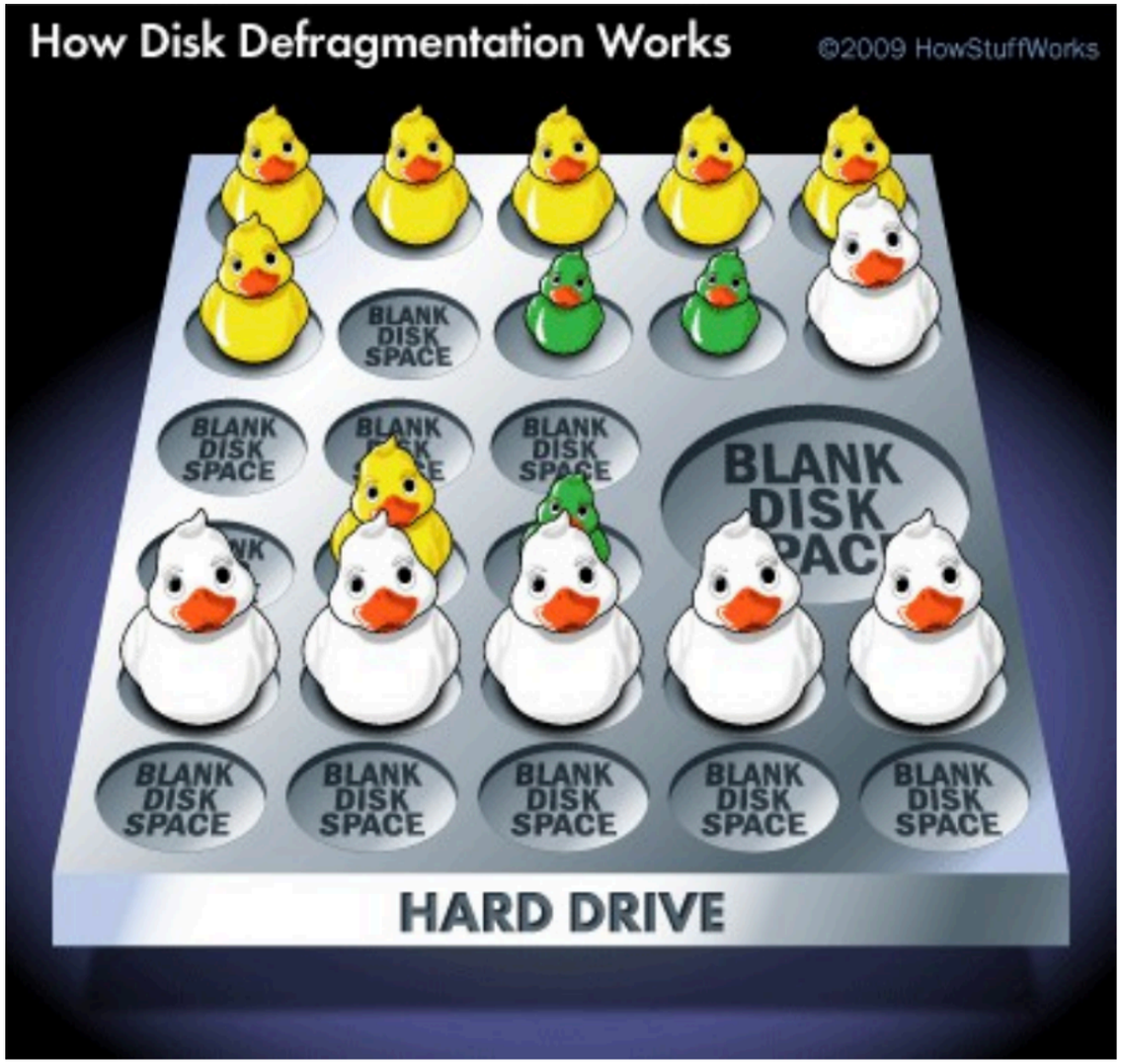
Op een gelijkaardige manier werkt een bestandssysteem: het opslagmedium wordt ingedeeld in gelijke "**blocks**". Het zou handig zijn om alle bestanden samen te houden, maar dat lukt niet. Immers, soms werkt je computer simultaan (multitasking) meerdere taken af en bewaart hij meerdere bestanden op ongeveer hetzelfde moment.

Je kan ook onmogelijk voorspellen hoe groot een bestand gaat worden vooraf. Als je vandaag een tekst intikt in een tekstverwerker, dan neemt dit bijvoorbeeld één block in. Wanneer je het morgen opnieuw opent en er nog wat tekst en afbeeldingen aan toevoegt, wordt het bestand groter, maar is het aanpalende block misschien al bezet door een ander bestand. Als je een bestand verwijdert, komt één of meerdere blocks vrij.

Virtuele gatenkaas: tijd voor defragmentatie

Na een tijdje ziet een opslagmedium er uit als een virtuele gaatjeskaas. Daarom raadt men computergebruikers ook aan om na x tijd de harde schijf te defragmenteren m.a.w. alle stukjes bestanden samenvoegen in aanpalende blocks.

Defragmentatie is onder Linux veel minder een probleem dan onder Windows. Ten eerste bewaart Linux bestanden niet in volgorde (dus niet automatisch in een aanpalend blok). Denk weer aan de hotelbaas: niemand verplicht hem om klant 1 in kamer 1 op het gelijkvloers te zetten, en klant 2 in kamer 2 op het gelijkvloers. Hij kan klant 2 ook in kamer 1 op het tweede verdiep plaatsen. Dan vormt het minder een probleem als er een gans gezin komt opdagen. Ook Apple beweert hetzelfde te doen. Ten tweede plaatst Linux de leeskop van de harde schijf meteen in het midden van de “platter” (het plaatje van de harde schijf) waardoor het systeem sneller bestanden kan vinden.



De werking van schijfdefragmentatie.

Bestanden terugvinden

Hoe onthoudt een computer nu waar al die fragmentjes zich bevinden? Hoe vindt de computer snel alle fragmenten van één bestand terug en maakt hij er weer één geheel van? En als hij er één geheel van maakt, waar bewaart hij dan dat tijdelijke geheel? Om op deze laatste vraag te antwoorden: het tijdelijke geheel bewaart hij in het "tijdelijke" werkgeheugen, het RAM-geheugen.

Tip: als je aan een document werkt, bewaar het dan meteen als een bestand en druk af en toe op de toetsencombinatie CTRL+S (CMD+S op Mac). Zo niet blijft het gehele bestand enkel in het RAM-geheugen en vertraagt de computer aanzienlijk. Als je te lang wacht (bijvoorbeeld bij het werken aan een beeld in Adobe Photoshop) kan het RAM-geheugen zo vollopen dat je computer crasht en je je bestand kwijt raakt.

Dan blijft onze eerste vraag: *Hoe vindt de computer snel alle fragmenten van één bestand terug en maakt hij er weer één geheel van?*

Dat hangt een beetje af van het gebruikte bestandssysteem. Het meest bekende en verbreide bestandssysteem is FAT (van Microsoft) dat ook door andere besturingssystemen kan gelezen worden. Het FAT is dan wel bekend en kent brede ondersteuning, maar het heeft ook tal van grote beperkingen. Dat merk je meteen wanneer je een bestand probeert op te zoeken. Dat kan ergerlijk veel tijd in beslag nemen, zeker als je gewend bent informatie te zoeken op Google.

Net zoals een hotelbaas houdt FAT een tabel bij (File Allocation Table = bestandstoewijzingstabel) waarin hij bijhoudt welk bestand op welk block staat.

Block	Bestand
1	Hond.jpg (1)
2	Eindwerk.doc (1)
3	Hond.jpg (2)
4	Hond.jpg (3)
5	Eindwerk.doc (2)
6	Puntenlijst.xls (1)
7	...

Om alle onderdelen van het bestand terug te vinden moet het OS heel de tabel doorlopen, en alle onderdelen vervolgens samenvoegen. Bij elke zoekopdracht doorzoekt het OS m.a.w. heel de tabel. Het is duidelijk dat dit bijzonder vertragend werkt en dat er een betere manier moet bestaan om een bestandssysteem op te bouwen.

Niet alleen het file system, maar ook de hardware kan hier vertragend werken. Een harde schijf is immers een mechanisch toestel (he hoort de schijfjes draaien. Het is dus niet echt één schijf, eerder een reeks "harde schijfjes") dat moet draaien langs een aantal leeskoppen. Het kan dus even duren vooraleer de mechanica en het file system alles hebben gevonden en samengevoegd.

UNIX en bij uitbreiding Linux en Mac OS X-systemen gebruiken **JOURNALING FILE SYSTEMS** en ook Microsoft deed zijn duit in het zakje met NTFS (New Technology File System). Het probleem van journalingsystemen is dat ze niet zonder meer uit te lezen en nog veel minder te "beschrijven" zijn door andere besturingssystemen.

Tip: wanneer je een harde schijf koopt voor gebruik onder Windows, dan past elke gewenste schijf. Koop je er eentje die je ook wil gebruiken onder Mac OS X, dan moet je goed opletten dat de schijf niet als FAT is geformatteerd. Linux en Mac kunnen NTFS vaak wel "lezen", maar er niet naar "schrijven".

*-NIX-besturingssystemen (UNIX, Linux, Mac OS X...) gebruiken journaling filesystems. Ze bewaren twee soorten informatie:

1. "metadata" over de bestanden en mappen
2. de bestanden zelf

Met "metadata" bedoelen we algemene informatie over het bestand zelf zoals de aanmaakdatum, de laatste bewerkingsdatum, de maker (gebruiker), gebruikersrechten... Per bestand wordt die informatie vastgehouden in een **inode**, op zichzelf ook een soort "bestand". Naast de metadata bevat zo'n inode ook een lijst met verwijzingen naar de plaatsen waar de onderdelen van het bestand zich op het opslagmedium bevinden.40 Een "journaling" filesystem houdt in dat het eveneens de wijzigingen aan de bestanden bijhoudt (net zoals in een "**journal**"). Op die manier kan het bestandssysteem zich relatief makkelijk herstellen (naar een vorige staat) na een crash.

Soorten file systems

FAT	File allocation system, bestandssysteem gebruikt in oude Windowsversies en op heel wat externe harde schijven.
exFAT	(Extended File Allocation Table) is een bestandssysteem speciaal bedoeld voor flash-schijven.
NTFS	New Technology File System, werd als opvolger voor FAT ontwikkeld door Microsoft. Het is een “journaling” systeem. Ondersteuning voor bestanden tot een grootte van 16 TB.
HFS+	“Journaling” bestandssysteem ontwikkeld door Apple voor gebruik onder Mac OS. Het werkt gelijkaardig als NTFS, maar beide systemen zijn niet compatibel. Het biedt ondersteuning voor bestanden en volumes tot meer dan 1 miljoen TB. Ook Linuxsystemen kunnen overweg met HFS, Windows niet.
Ext4	Het “journaling” bestandssysteem ext4 (opvolger van ext2 en ext3) is het meest gebruikte bestandssysteem onder Linux. Het ondersteunt bestanden met een grootte tot 16 TiB (tebibyte!) en volumes tot 1 EiB (exbibyte). Vermits het grootste deel van de webservers (computers waarop websites worden “geplaatst”) op UNIX- en Linux-systemen draait, is ext4 zeker niet te onderschatten.
Btrfs	Lees je als “B-Tree FS”, “Better FS”, of “Butter FS”, is een alternatief voor ext onder Linux en groeit aan populariteit. De maximaal ondersteunde bestandsgrootte is 16 EiB. Btrfs heeft een aantal andere krachtige eigenschappen zoals “data pooling (bestanden spreiden over meerdere harde schijven), compressie en encryptie, het maken van snapshots, data-deduplicatie enz.
ZFS	ZFS is enkel bekend in de UNIX-wereld. Het biedt ondersteuning voor bestanden tot 16 Exabyte. Net zoals Btrfs ondersteunt het compressie, encryptie, data-dediplicatie en daarnaast ook ondermeer bescherming tegen datacorruptie. Ook OS'en zoals MAc OS X 10.5 Server en een aantal Linuxversies ondersteunen ZFS. Toch is het eerder in gebruik bij grote ondernemingen dan bij thuisgebruikers.
ISO9660	Bestandssysteem van CD en DVD. Joliet, Rock Ridge en El Torito zijn hier "uitbreidingen" van.
UDF	Universal Disk Format, open vendor-neutraal file system. Vooral gebruikt bij DVD, BluRay en andere optische media.
HDFS	Hadoop Distributed File System, voor gedistribueerde opslag en verwerking van grote hoeveelheden data (big data). Andere big data-opslagsystemen zijn Quantcast File System, Ceph, Lustre, GlusterFS, PVFS.

DATABANKEN

Leerstof tweede en derde jaar (nog niet beschikbaar)

KLEINE DATA EN BIG DATA

Leerstof tweede en derde jaar (nog niet beschikbaar)

DEEL 5 INTERFACES

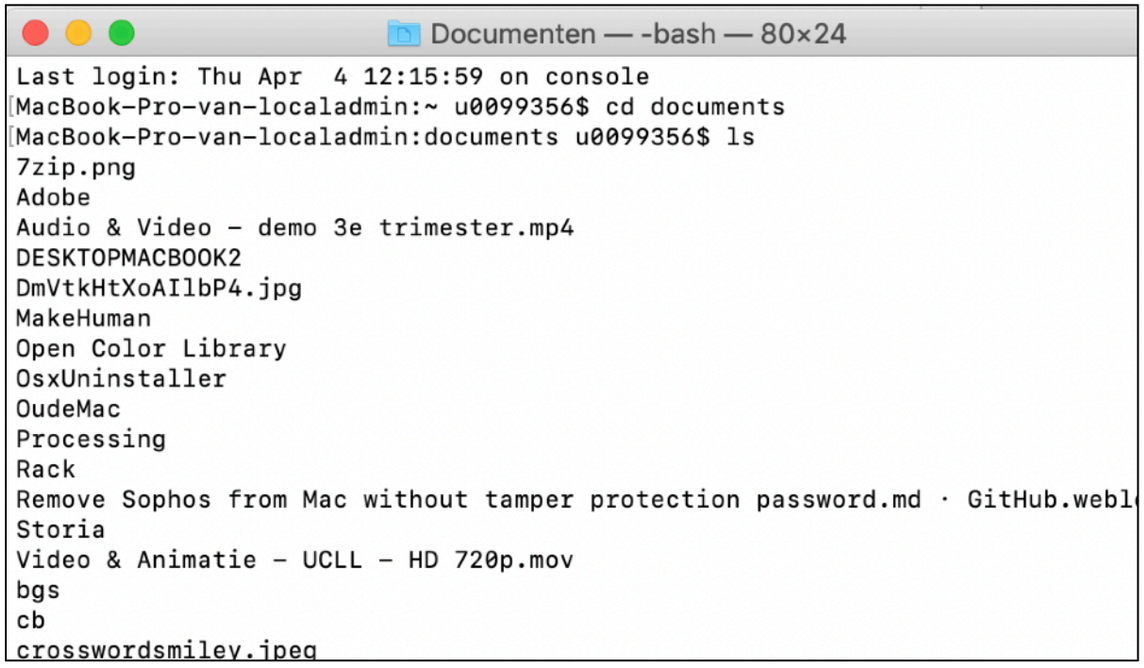
- Op welke manieren kunnen we communiceren met een computer?
- Hoe bouw je gebruiksvriendelijke interfaces voor computersystemen?
- …

SOORTEN INTERFACES

Mensen en computers communiceren met elkaar. Als je op een computer werkt, dan ben je voortdurend in wisselwerking (interactie) met het digitaal systeem. Mens en machine, in dit geval alle denkbare vormen van digitale systemen, kunnen niet zonder problemen met elkaar communiceren. Informatie die voor mensen heel begrijpelijk is, zoals het herkennen van andere mensen en objecten, is voor computers onbegrijpelijk. Om de communicatie en interactie mogelijk te maken, voorziet een digitaal systeem in een interface. Een interface zorgt ervoor dat de informatie van het ene systeem (mens of computer) herkenbaar en begrijpelijk wordt voor het andere.

Meer een meer kunnen digitale systemen direct communiceren met fysieke objecten en hun omgeving. Interfaces beperken zich dus niet enkel tot omgang met de mens.

De interfaces zijn voortdurend in ontwikkeling. Een nieuwe stap in de ontwikkeling betekent niet dat alle vorige interfaces overboord worden gegooid. *Een blik op de evolutie…*



Terminalvenster onder Mac OS X. Met het commando "cd" kan je naar een andere map (change directory) en het commando "ls" (list) geeft een lijst van alle aanwezige bestanden.

Het Command prompt-venster onder Windows.
(Bron: https://www.computerhope.com/issues/chusedos.htm)

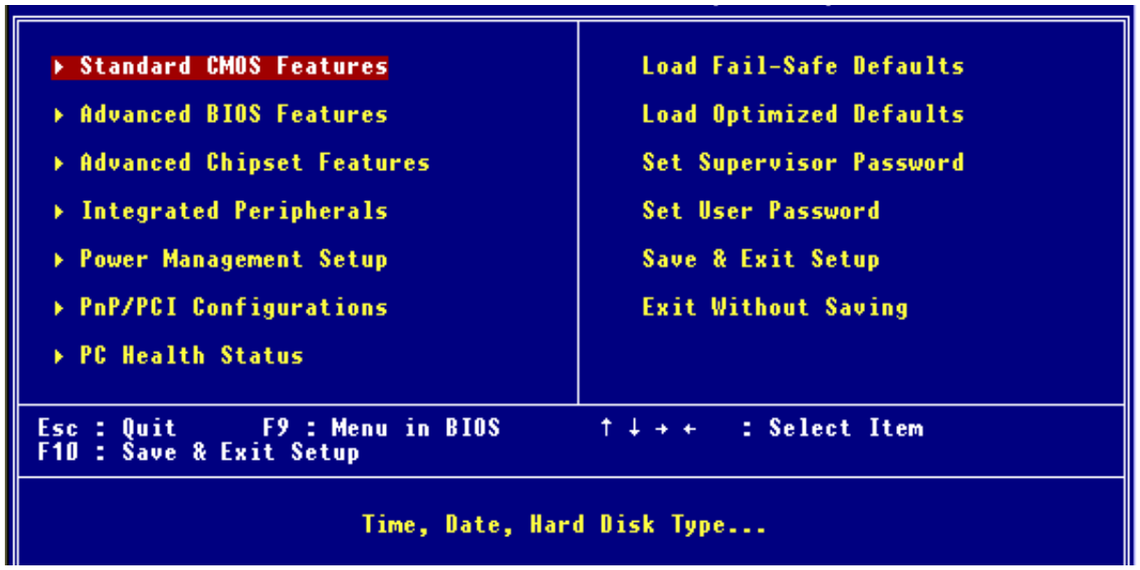
Command line interface (CLI)

Oorspronkelijk vormden **CLI's** de gebruikersinterface voor de meeste computersystemen. Gebruikers gaven via tekstcommando's opdrachten in. Een CLI ondersteunt over het algemeen het gebruik van de muis niet, maar invoer via het toetsenbord.

Nadeel is dat de gebruiker de benodigde **commando's** moet onthouden. Om dit probleem te verhelpen voorzien veel CLI's in een helpfunctie met een overzicht van alle commando's. Die helpfunctie eveneens met een commando opgeroepen worden.

In de meeste besturingssystemen is nog steeds een CLI verborgen. Systeembeheerders en linuxnerds zweren bij het gebruik van de Console. Onder Mac OS X heeft dit net zoals op tal van UNIX- en Linux-systemen de **Terminal** of console. Onder Windows kunnen gebruikers het **opdrachtregelvenster** openen, door velen nog steeds het DOS-venster genoemd (naar het oude besturingssysteem Ms DOS of Microsoft Disk Operating System).

De eerste generatie iPods had een TUI-interface.
(Bron: https://www.cnet.com/pictures/the-complete-history-of-apples-ipod/)



BIOS-venster met TUI.
(Bron: http://www.halict.nl/lcinformatica/klas4/h5/h5_1.htm)

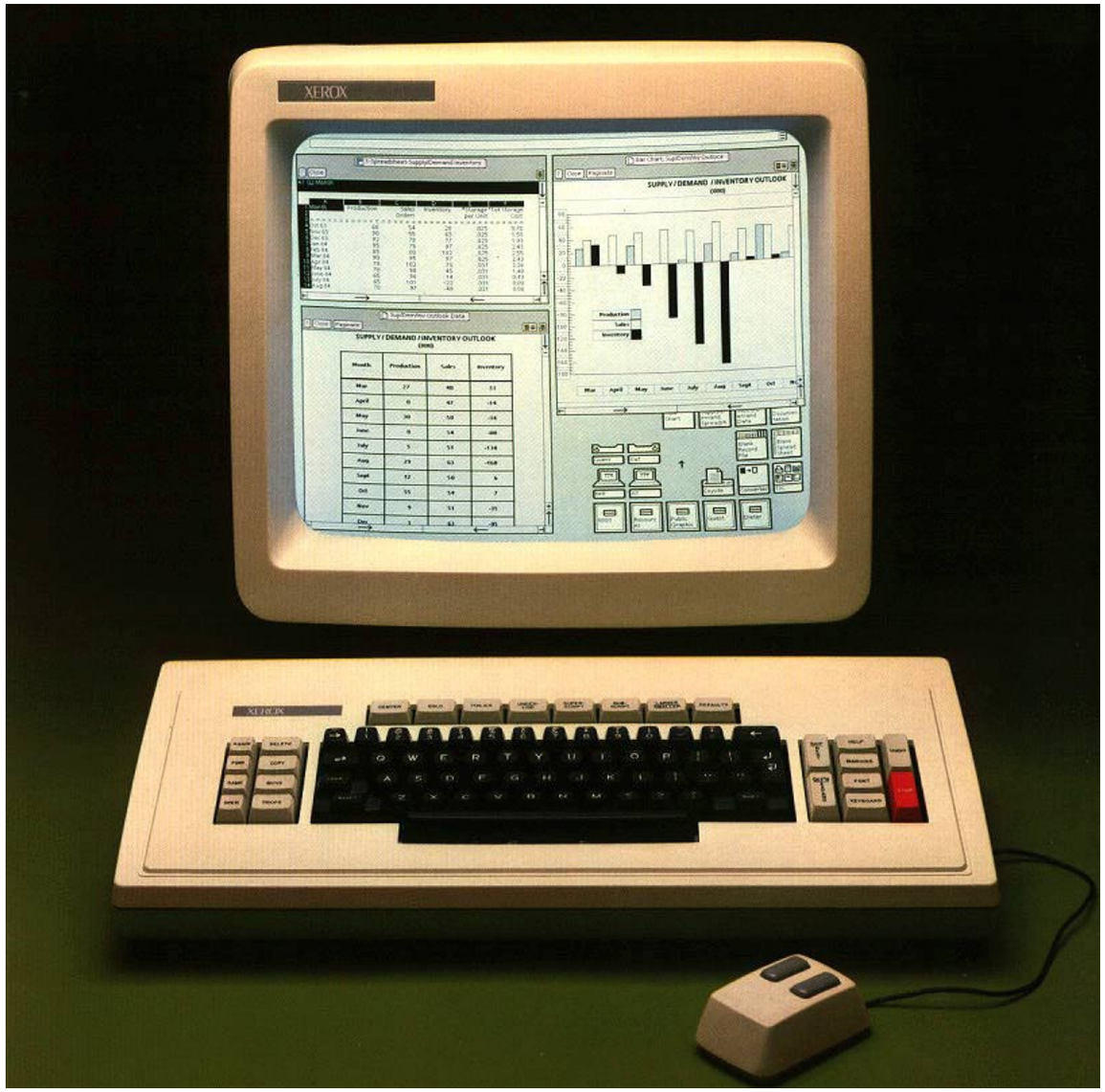
Text user interfaces en mnemonics

De volgende stap of verbetering bestond erin om het ganse scherm te benutten voor de weergave van de output en het gebruik van de muis of een ander pointing device mogelijk te maken.

Primitieve vensters en knoppen werden ingezet. Zulke TUI's zijn nog steeds in gebruik voor het instellen van het basissysteem (**BIOS**) van computers en soms duiken ze ook nog op bij ietwat out-of-date kassasystemen.

De diverse CLI-commando's werden verborgen achter “knoppen”. Een **TUI** kon aangestuurd worden met de muis, maar al even snel via “**mnemonics**”: in het tekstopschrift op de knop werd één letter onderstreept. Wanneer de gebruiker die indrukte op zijn toetsenbord werd de opdracht uitgevoerd.

De eerste versies van de iPod (niet de iPad!) hadden een TUI-interface, net zoals heel wat (oudere) GSM's.



De XEROX STAR-computer (1982-1984) was de eerste computer met een WYSIWYG-interface of GUI. Bekende interface-metaforen als "desktop", "selecting", maar ook vensters, menu's, schermpictogrammen en het gebruik van de muis, werden voor het eerst effectief ingezet op dit toestel. Functies als kopiëren, plakken, verwijderen, internethulp, online bestanden… waren aanwezig.
(Bron:http://blog.iso50.com/5839/your-computers-grandpa/)

Graphical User Interface (GUI)

Een grafische gebruikersinterface of GUI verschilt van de voorgaande interfaces omdat het het scherm aanstuurt op pixelniveau via wat men omschrijft als de schermbuffer. Elke pixel kan hierdoor ingekleurd of aangestuurd worden. Dit vraagt echter veel meer rekenkracht van de processor en de grafische kaart. In onze tijd is de rekenkracht meer dan hoog genoeg om dit mogelijk te maken waardoor zelfs tablets en smartphones over een GUI beschikken.

De leercurve van een GUI is laag omdat het uitgaat van het ontwerpprincipe **"herkenning boven herinnering"**. De gebruiker hoeft geen commando's meer te 'kennen' of in te voeren. De meeste gebruikers maken geen onderscheid meer tussen de GUI en het achterliggende besturingssysteem. Wanneer men spreekt over MS Windows, Mac OS X, iOS... dan stelt men zich dat vooral visueel voor. In commerciële software zijn OS (besturingssysteem) en GUI inderdaad onlosmakelijk met elkaar verbonden. Onder Linux is dit niet zo.

Afhankelijk van de eigen smaak of voorkeur, kan de Linuxgebruiker kiezen tussen diverse desktopomgevingen (desktopomgeving = GUI). De meest bekende GUI's voor Linux zijn Gnome en KDE. Rijzende sterren zijn ondermeer Unity (de desktop van de nieuwe Linux Ubuntu), Xfce, Cinnamon (GUI van Linux Mint). Linux GUI's zijn bijzonder configureerbaar waardoor je het echt op maat kan "versnijden". Het is perfect bruikbaar om in te zetten als **KIOSK**-computer.

Windows, icons, menu's, pointing devices (WIMP)

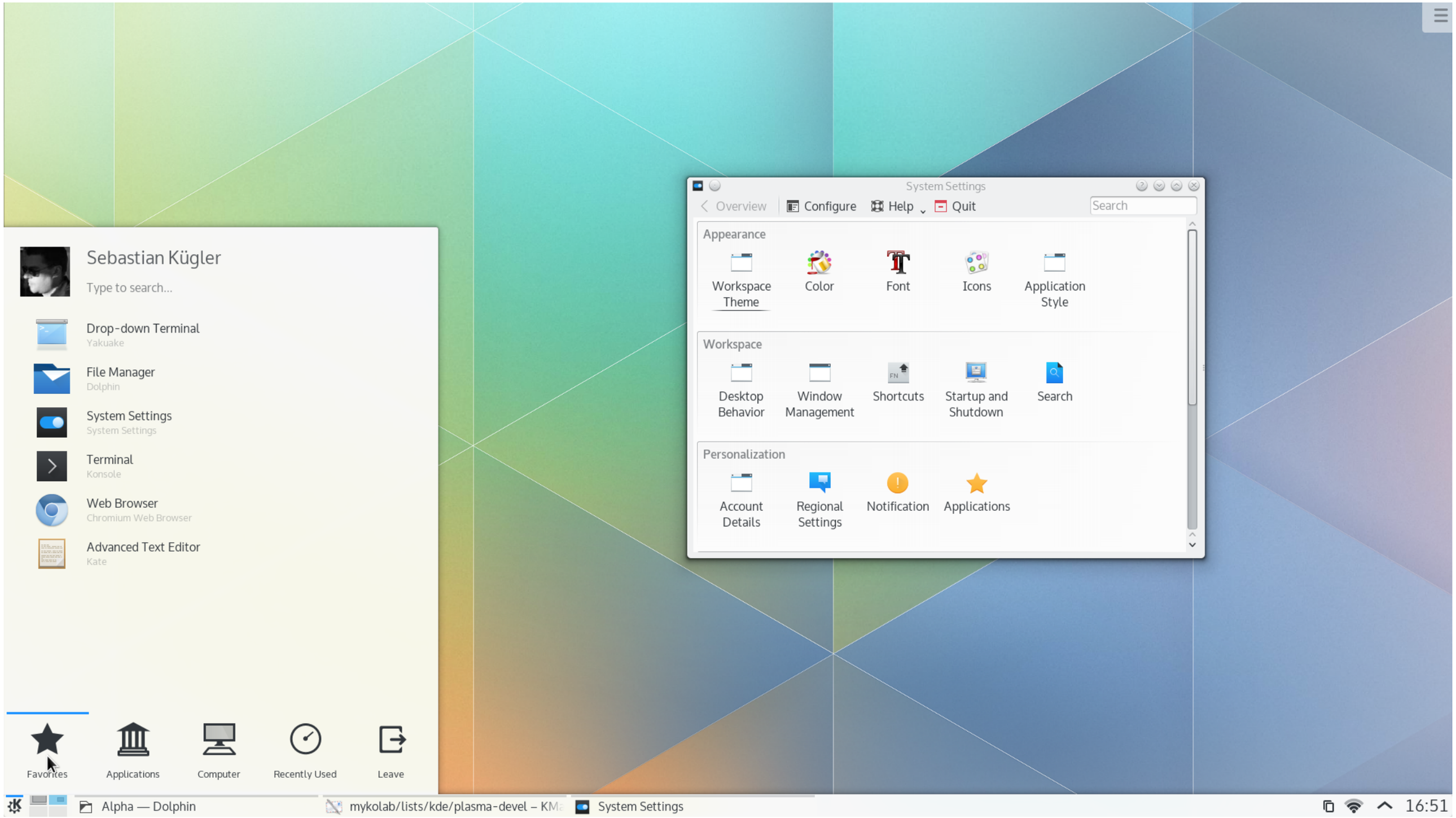
Een GUI heeft een zogenaamde WIMP-interface. Een gebruiker bedient de GUI via vensters (Window), icoontjes (Icon), menu's (Menu) en bijvoorbeeld een muis en toetsenbord (Pointing Device). Daarom gebruikt men voor een GUI ook wel eens de naam WIMP.

Door het gebruik van **vensters** is **multitasking** mogelijk. Je kan meerdere programma's simultaan gebruiken. Binnen een venster voorziet de software-ontwikkelaar een aantal visuele menu's en intuïtief te gebruiken bedieningselementen.

Schermpictogrammen maken het mogelijk om een programma snel te starten. Heel wat besturingssystemen proberen het gebruik van pictogrammen op het virtuele bureaublad te beperken. Bij Mac OS X zijn de pictogrammen ondergebracht in de dock onderaan het scherm, maar toch houdt Apple nog meer vast aan het gebruik dan bijvoorbeeld Microsoft. In Apples iOS voor iPhone en iPad zijn schermpictogrammen de manier om het toestel te bedienen. Microsoft heeft schermpictogrammen vervangen door zijn nieuwe metro-interface waarbij ze zijn vervangen door grote aanklikbare vlakken. Toch merk je dat veel gebruikers verknocht zijn aan het gebruik van pictogrammen.

Door het gebruik van **menubalken en uitklapbare menu's** is het oproepen van bepaalde functies en commando's heel makkelijk. In veel gevallen moet de gebruiker eerst het te bewerken onderdeel (een deel van een tekst, een stuk van een afbeelding, het ganse 'bestand'...) selecteren en vervolgens op een menuknop drukken.

Met een muis of een andere **"pointing device"** zoals een tekentablet, een trackpad enz. kan de gebruik op schermpictogrammen of menuknoppen klikken of dubbelklikken, hij kan elementen selecteren of verslepen.



GUI van de KDE-desktop onder Linux. Menu's, muiscursor, schermpictogrammen, vensters.

(Bron:https://kde.org/announcements/plasma5.0/)

KODI-mediacenter-GUI.

(Bron: https://www.smarthomebeginner.com/kodi-estouchy-skin-review/)

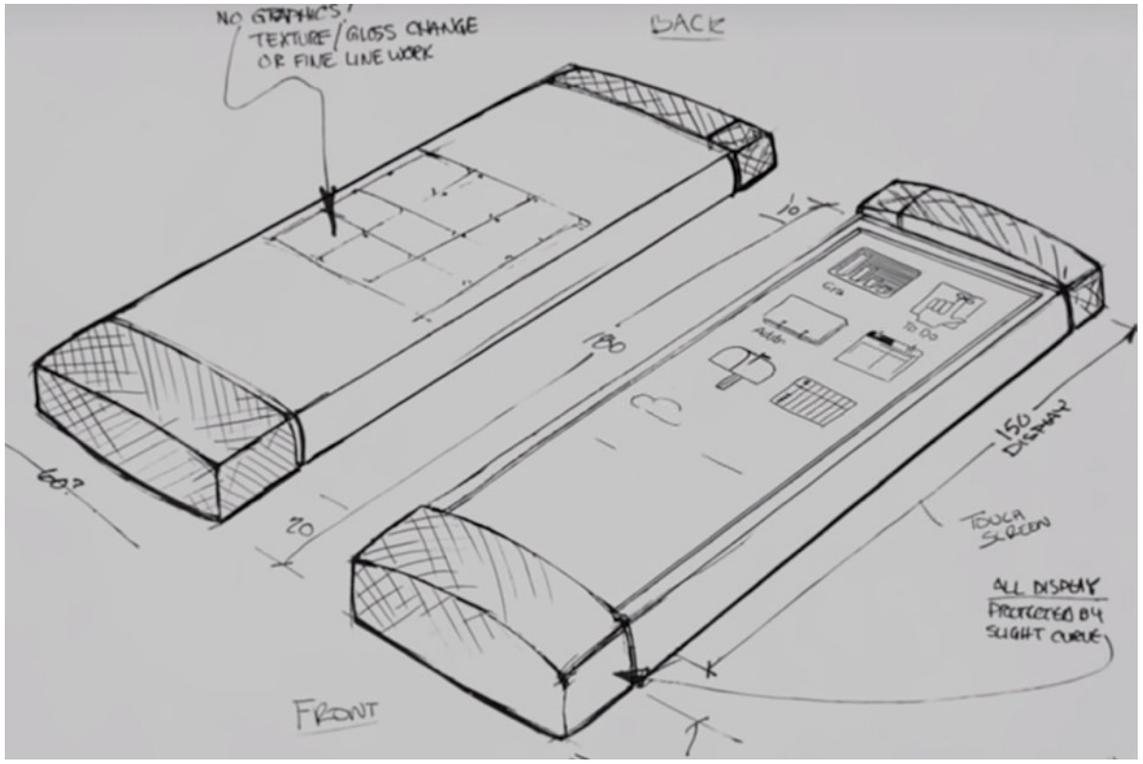
Natural User Interface (NUI)

De manier waarop mensen een digitale interface bedienen is nog ver verwijderd van de wijze waarop mensen met elkaar communiceren: spraak, aanraking, lichaamstaal en gebaren, gevoelens herkennen... *Nieuwe interfaces integreren zoals het aanraakschermen proberen die omgang herkenbaarder en natuurlijker te maken.*

Hierdoor bepekt het aansturen van commando's zich niet langer tot klikken en dubbelklikken, maar kan de computer ook andere "gebaren" herkennen. Apple integreerde een deel van die **gebarenherkenning** ook in de zogenaamde Magic Mouse en trackpad: het toestel herkent niet langer slechts één vinger, maar kan bijvoorbeeld een afbeelding roteren door met twee vingers een draaiende beweging te maken. Het bouwen van **spraakgestuurde interfaces (Voice User Interfaces of VUI)** loopt niet van een leien dakje. Daarvoor moet een computer vooral menselijke taal leren begrijpen.

Op de consumentenmarkt raken **NUI's** steeds meer ingeburgerd, vooral dankzij spelcomputers. De Wii herkent de handbewegingen van de gebruikers/speler zo lang hij een pointing device (de Wiimote, een soort afstandsbediening) vasthoudt. De Kinect is een invoerapparaat voor motion sensing met ingebouwde microfoon en webcam van Microsoft voor gebruik bij de Xbox 360. Dankzij de Kinect wordt interactie mogelijk zonder game controllers. Jammer genoeg is de ontwikkeling van de Kinect stopgezet. Samen met de Wiimote, tablets (iPad) en multitouch-smartphones behoort het tot de eerste generatie **STAG (Speech, touch and Gestures)**-toestellen. Gebaren en gesproken commando's volstaan om instructies door te geven aan de (spel)computer. Sommigen noemen de Kinect ook het eerste voorbeeld van een **Zero interface**, omdat je niets in je "handen" hebt. De software werd ontwikkeld door Rare (een firma die eveneens in handen is van Microsoft) en de cameratechnologie door de Israëlische ontwikkelaars PrimeSense.

De camera bouwt een 3D-beeld op van de omgeving aan de hand van infrarood laserprojecties. De 3D-scanner (Light Coding) gebruikt een soort image-based 3D-reconstructie. Het toestel beschikt over **voice recognition**, **full-body 3D motion capture en gezichtsherkenning**. Volgens Microsoft herkent het toestel simultaan tot personen en kan het bewegingsanalyse uitvoeren op 2 actieve spelers.

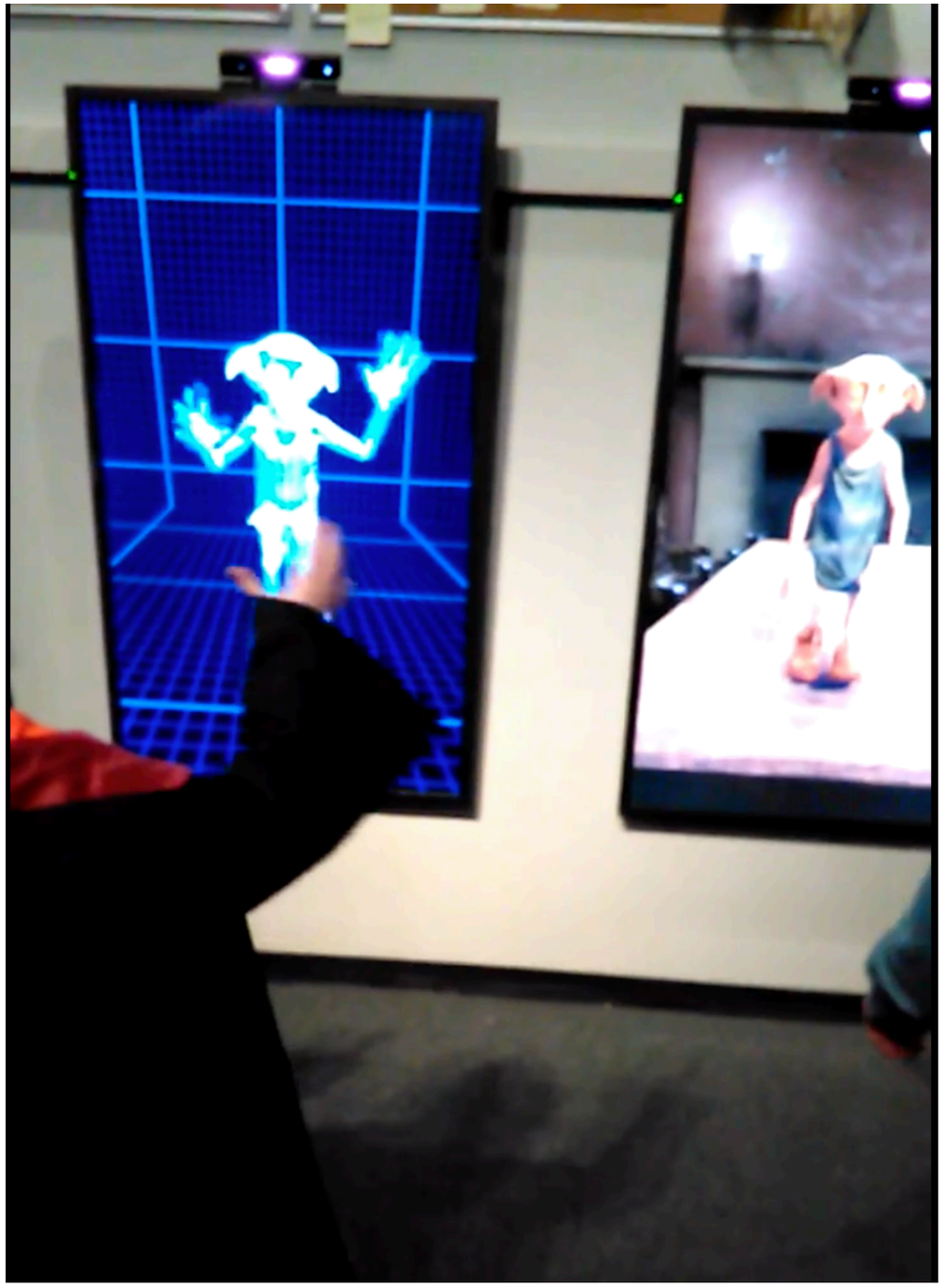


Concepttekening van een smartphone met touchscreen door GeneralMagic. Het project startte in 1989 als een spin-off van Apple. De firma was zijn tijd ver vooruit en ging overkop ondanks de steun van SONY. De ontwikkelaars gingen aan de slag bij Apple waar ze de iPod en iPhone ontwikkelden en bij Google waar ze aan de basis lagen van Android.
(Bron: https://uncrate.com/video/general-magic/)
Hieronder het enige model dat GeneralMagic op de markt bracht.
(Bron: https://www.iculture.nl/nieuws/general-magic-documentaire/)

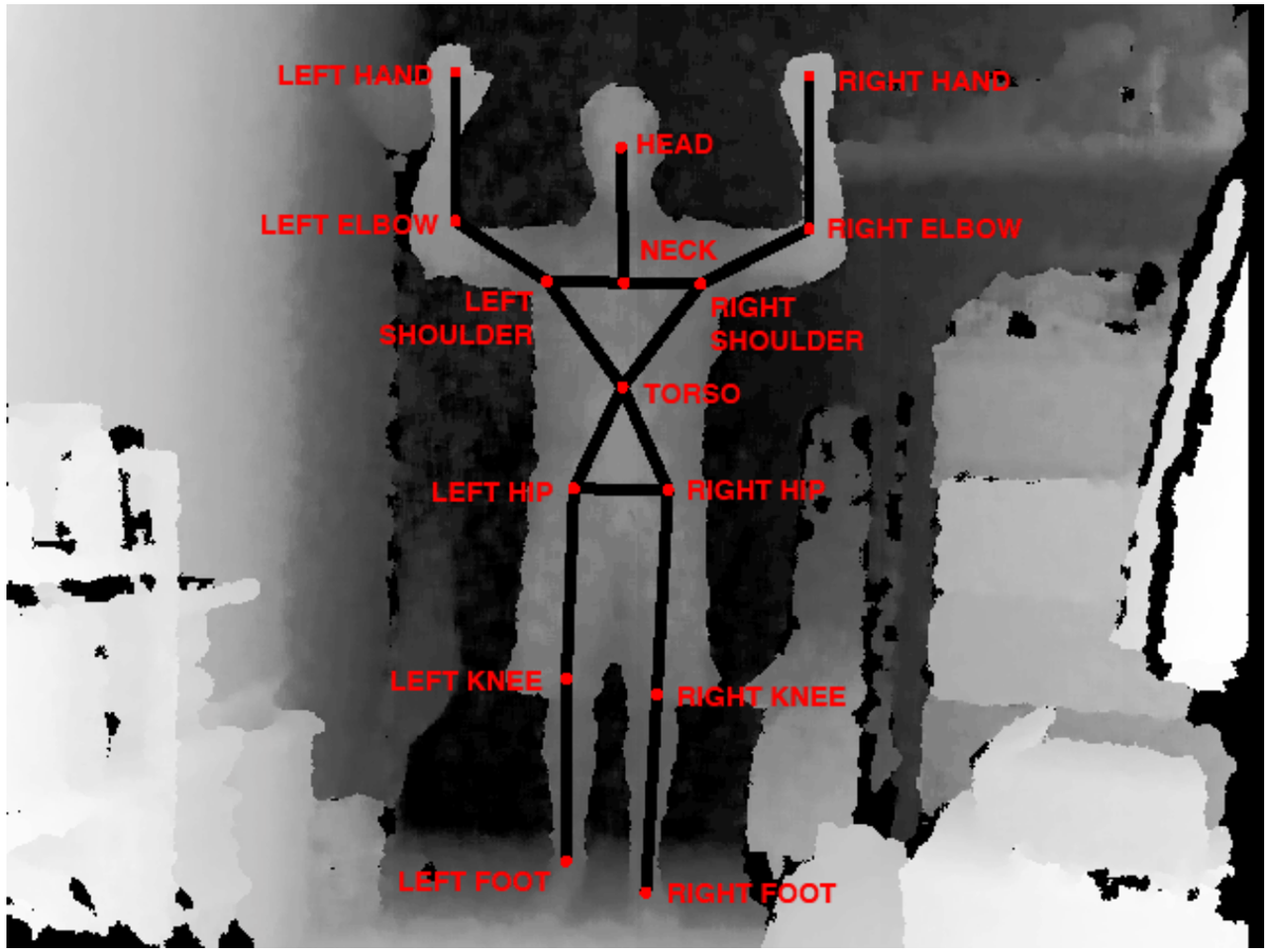
PrimeSense daarentegen beweert dat het aantal herkende personen enkel beperkt is door het aantal dat binnen de field-of-view van de camera valt. De Leap Motion is een via USB aan te sluiten toestel dat **gesture recognition** naar de gewone computers (Linux, Windows, Mac OS X) brengt.

De Leap Motion-controller.
(Bron: https://www.leapmotion.com/)

Uiteraard zullen we de klassieke computer/laptop met GUI en toetsenbord nog een hele tijd blijven gebruiken. Maar aanraakschermen zijn ondertussen al even zeer ingeburgerd. Ook **schermloze interfaces** op basis van **voice recognition** zijn aan een opmars bezig: Amazon Alexa, Google Home… zijn voorbeelden van systemen die op het vlak van user interfaces enkel over een microfoon en luidspreker beschikken, maar toch ook andere apparaten kunnen aansturen.



De KINECT vindt zijn weg naar de filmstudio. Bij Warner Bros Studios in Londen gebruikt men de Kinect voor het animeren van 3D-figuren.



In Processing kan je met de OpenKinect-library bewegingen laten herkennen, maar ook 3D-scans van mensen maken.

(Bron: http://artandcode.com/3d/workshops/1a-using-the-kinect-with-processing/)

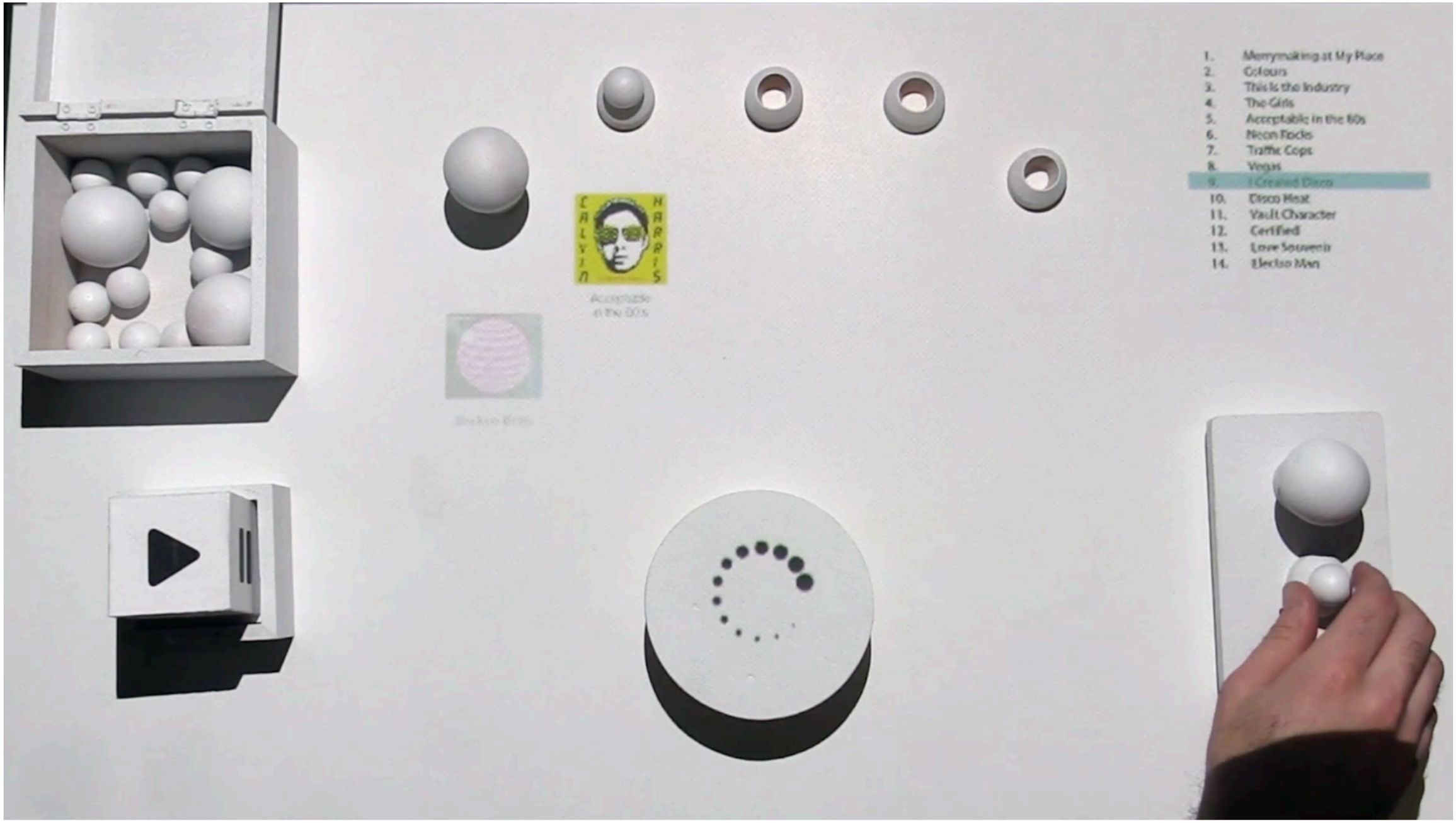
Tangible User Interface (TUI)

De afkorting TUI kan staan voor Text User Interface zoals we kennen uit het oeroude DOS of de Terminal van Mac OS X, maar in deze context staat het voor **Tangible User Interface**. *Bij een Tangible User Interface gebruik je echte voorwerpen als interface-elementen.* Bij een TUI worden echte objecten "verhoogd" (augmented) met digitale functionaliteit! Zo kan je bijvoorbeeld een echte (fysieke) tastbare draaiknop op het scherm plaatsen in plaats van een digitale knop. Wanneer je aan die knop draait, herkent de computer de draaibeweging en voert een daaraan gekoppelde functie uit. Een TUI maakt interactie tussen fysieke objecten en computers mogelijk via **markers** en **computer vision**. De PlayStation Eye bestaat uit een digitale camera en gebruikt software en hardware voor computer vision en "gesture recognition". *Het toestel kan patronen, kleuren, markers en geluid herkennen.*

Microsoft Pixelsense (voorheen bekend als Surface) is een technologie voor TUI's. Een Pixelsensescherm herkent objecten die de gebruiker op het scherm plaatst. In de geavanceerde versie van Samsung SUR40 (uitgerust met Microsoft PixelSense) bevat het scherm sensoren in plaats van camera's. Je kan bijvoorbeeld met echte borstels

virtueel schilderen. Het systeem herkent ook de grootte en de vorm van de objecten. Een kaart met een bepaalde tag, kan door het scherm uitgelezen worden. Als je je smartphone op het scherm neerlegt, toont het al je smartphonefoto's.

Foto bovenaan: Microsoft Pixelsense
(Bron: http://futuretechnebiz.blogspot.com/2017/04/tangible-user-interface.html)



Tangible User Interface voor een muziekspeler.

(Bron: https://www.youtube.com/watch?v=ZQdIxQ_EOLI)

Zoomable User Interface (ZUI)

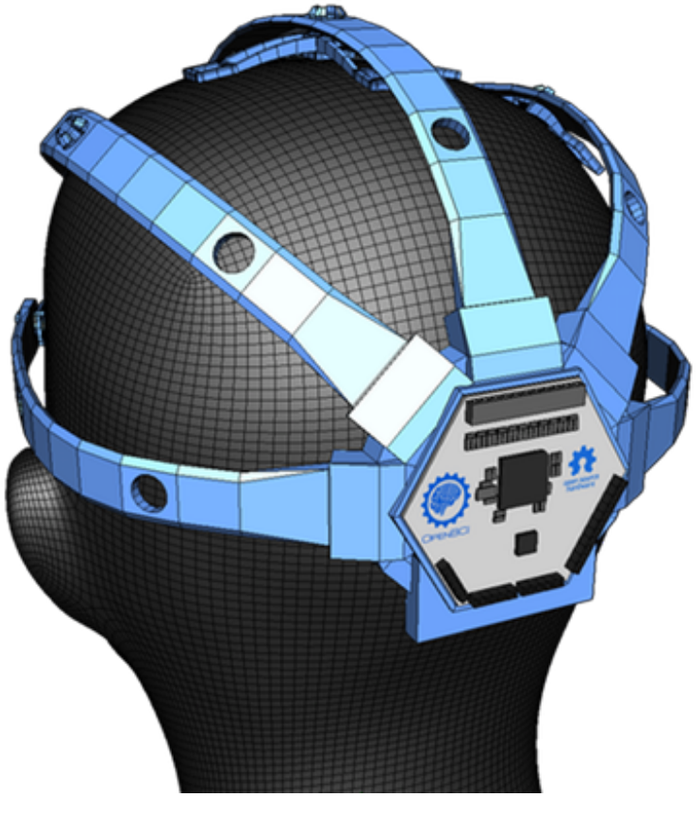
Een klassieke GUI heeft een "eindig" bureaublad. Een zoomable user interface gaat uit van het concept van een "oneindig" bureaublad waar je door in of uit te zoomen, kan navigeren. Ondermeer Jef Raskin, de oorspronkelijke ontwikkelaar van de Mac OS-GUI probeerde zo'n systeem op de markt te zetten. Ook Sun Microsystems probeerde iets gelijkaardigs te doen. met de Looking Glass-GUI voor UNIX-computers. De ZUI is er dus niet in geslaagd om de klassieke GUI te vervangen, maar elementen ervan zijn zeker al doorgedrongen in bestaande software. Google Maps en Google Earth danken een groot deel van hun functionaliteit aan de mogelijkheid om te zoomen. Het bekendste voorbeeld is ongetwijfeld de online presentatietool Prezi, waarin je niet over één slide, maar over een oneindige "slide" beschikt, waar je al zoomend door beweegt.

Brain Computer Interface

Beeld je in dat je commando's kan doorgeven aan een computersysteem door je commando gewoon te "denken". Het klinkt voor sommigen misschien angstaanjagend, maar het is niet moeilijk om hiervoor zinvolle en functionele toepassingen te bedenken. Een verlamd persoon zou op die manier een auto kunnen besturen. Misschien is het mogelijk om op die manier te communiceren met vegetatieve of "locked-in" patiënten.

Om hersenactiviteit te meten bestaan er ruwweg twee technieken. Met **fMRI** (functional magnetic reconance imaging) kan je de plaats van hersenactiviteit lokaliseren. Bij fMRI maakt men een 3D-afbeelding van de hersenen. Hierop kan je zien waar bepaalde hersenactiviteiten plaats vinden. Hiervoor is een grote dure scanner vereist. Een **EEG** (Elektro-encefalografie) is een methode om de elektrische activiteit van de hersenen te meten. Hierbij worden elektroden op de hoofdhuid geplakt. Hoe meer elektroden, hoe hoger de “resolutie”. Als je meet welke elektrische signalen zijn gekoppeld aan bijvoorbeeld armbewegingen, kan je die signalen eveneens gebruiken om een computer aan te sturen.

Voor de consumermarkt zijn de eerste EEG-headsets te koop: *Muse, Epoc, Insight, OpenBCI, Neurosky Mindwave…* Vaak meten ze niet enkel hersengolven, maar eveneens minuscule hoofdbewegingen. De **Emotiv EPOC** (https://www.emotiv.com) is te koop voor 799 dollar.



Het Franse Institut national de recherche en informatique et en automatique (http://openvibe.inria.fr/) ontwikkelt OpenVIBE, een softwarepakket bestemd voor het ontwerpen, uittesten en gebruiken van **Brain-Computer Interfaces (BCI)**. **EMOKIT** daarentegen is een eenmansproject van de hacker DAEKEN en is ultrasimpel in gebruik. De softwarekit is gebaseerd op de programmeertaal Python waardoor je applicaties kan ontwikkelen voor Windows, Mac OS X en Linux. EMOKIT maakt verbinding met de Emotiv via Bluetooth via een HID-interface, leest de versleutelde data uit en stopt die in een queue die je dan weer in je eigen applicatie kan inlezen en gebruiken.

Bovenaan: OpenBCI
Onderaan: Emotiv EPOC
(Bron: http://learn.neurotechedu.com/headsets/)

Immersive interfaces

Bij een **immersive interface** *gaat de computerinterface volledig of gedeeltelijk op in de echte werkelijkheid*. Je kan dit bereiken door bijvoorbeeld een interface te projecteren of weer te geven op een contactlens of bril.

Een Tangible User Interface gaat natuurlijk al die richting uit. Zelfs een smartboard in een klaslokaal biedt gelijkaardige functionaliteit.

Lumo Play, GestureTek en Motion Magix bieden technologieën om de interface van een stuk software of game op een muur of vloer te projecteren. De gebruikers kunnen de software interactief bedienen door aanrakingen (voeten, handen) of door gebaren.

Met **Bare Conductive TouchBoard Starter Kit** en elektrisch geleidende verf (bare conductive paint) kan je zelf je eigen interactieve immersive interfaces bouwen.

Bovenaan: Lumo Play
(Bron: http://www.visionone.com.au)
Onderaan: Bare Conductive
(Bron: https://www.bareconductive.com)

Augmented of *mixed* reality

Augmented reality voegt aan beelden van de werkelijkheid digitale informatie toe. Je kan digitale informatie projecteren op muren van gebouwen. Maar dit betekent nog niet dat elk geprojecteerd beeld een vorm van augmented reality is. *Bij AR herkent het computersysteem op een of andere manier de "realiteit" en voegt daar digitale informatie aan toe.*

Tot de oudste voorbeelden behoren de **head mounted displays (HMD) of head-up displays (HUD)** die gevechtspiloten (F35-piloten) dragen. Het directe beeld van de omgeving dat ze door hun helm en door de ramen van het vliegtuig te zien krijgen, wordt aangevuld met symbolen en/of een kaart. 's Nachts toont de HUD een 3D-wireframe van de omgeving zodat de piloten letterlijk blindelings kunnen vliegen. De software moet m.a.w. "weten" waar de piloot naar kijkt. Hiervoor moet het AR-toestel niet alleen de lokatie van het vliegtuig kennen, maar eveneens voortdurend de **kijkrichting (field of view)** van de piloot in de gaten houden.

De piloot krijgt dus **directe beelden** van de omgeving waarover de computer een gedeeltelijk transparante laag met digitale informatie legt. **Google Glasses** zijn/waren een gelijkaardig voorbeeld. De directe beelden van de omgeving krijgen een extra laag digitale informatie. **AR-contactlenzen** bevatten biosensoren. Ze hinderen het normale zicht niet, maar tonen eveneens een glucosemonitor. Ideaal dus voor mensen met diabetes. De lens/lenzen voorziet/n zichzelf van energie via lage radiofrequentie-signalen (**RF power**) waarmee een kleine hoeveelheid energie kan worden opgewekt, want slechts een paar microwatt zijn nodig om een voor het oog zichtbaar beeld op te bouwen. Het menselijk oog heeft immers geen massa licht nodig om informatie te registreren.

Een van de meest verrassende uitvindingen (uitgevonden door Kazuo Yoshinaka van Nippon Electric Co. in 1986) die stilaan zijn weg naar de markt vindt is de **Virtual Retinal Display (VRD)**. De VRD probjecteert beelden rechtstreeks op het netvlies. Het resulteert in zeer heldere en contrastrijke beelden met hoge resolutie. Omdat aan elk oog een afzonderlijk beeld kan getoond worden, biedt het de mogelijkheid om stereoscopische 3D-beelden te tonen (zie onderdeel "3D"). Het biedt de mogelijkheid om het zicht te "herstellen" van mensen met gezichtsproblemen.

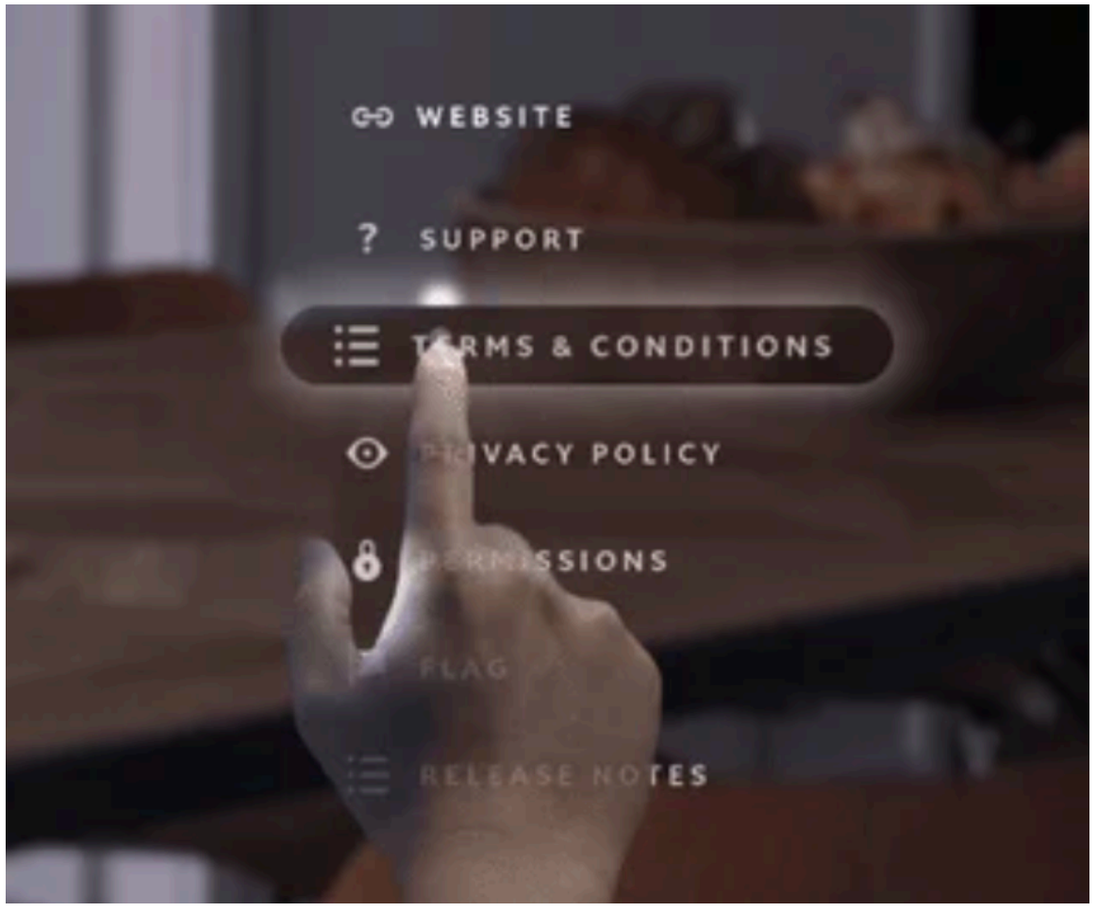
De **Magic Leap** is zonder twijfel de eerste gecommercialiseerde versie van een VRD. Het beeld dat een gebruiker is ronduit indrukwekkend. De firma ontwikkelde het eigen operating system **Lumin OS** voor de weergave van beelden op de VRD. De virtuele objecten lijken echt te bestaan naast de echte objecten in je omgeving.

In vergelijking met de Magic Leap lijken AR-applicaties op een smartphone uit de AR-prehistorie te stammen. Op een smartphone spreken we van **indirecte AR**. De gebruiker kijkt immers niet direct naar de werkelijkheid, maar naar een display waarop camerabeelden worden getoond. *Je richt bijvoorbeeld de camera van je smartphone op een gebouw en de software voegt aan het camerabeeld informatie toe over de bouwgeschiedenis. Of je richt je smartphonecamera op een paar schoenen in een winkel en je krijgt er meteen informatie over de prijs en gelijkaardige modellen bij.*

Augmented reality omvat een reeks invoer- en uitvoertechnieken. Om de lokatie van de gebruiker te weten te komen, gebruikt AR-software informatie van allerlei **sensoren**: GPS-chips, digitaal kompas, tiltsensor, microfoon, camera's… Door die sensordata te combineren en te analyseren, komt de software te weten waar de gebruiker zich bevindt en in welke richting hij "kijkt".

Heel vaak gebruikt AR-software ook **image recognition-algoritmes**. Tot de meest bekende AR-toepassingen behoort de mogelijkheid voor het herkennen van **QR-codes** en **AR-markers**. Je richt je camera op een "marker" en de software tovert een laag digitale informatie op het camerabeeld. Nieuwe technieken zoals **NFC (Near Field Communication) en BLE (Bluetooth Low Energy)** openen nog meer perspectieven.

De manier waarop AR-software digitale informatie toevoegt, loopt uit elkaar van eenvoudige **2D-overlays tot 3D-mapping en zichtbare 3D**. Bij de Magic Leap krijgt het de gebruiker het gevoel dat de digitale/virtuele objecten echt aanwezig zijn in de werkelijkheid. Bekende voorbeelden vonden hun weg naar het klassieke boek. De uitgever neemt in zijn boeken AR-markers (een soort van QR-codes) op. Wanneer de lezer die met een smartphone of tablet scant, verschijnt er op de display bijvoorbeeld een "gemapt" 3D-model. Wanneer de gebruiker met zijn toestel rond het boek wandelt, herrekent (mapping) de software het 3D-model op de marker.

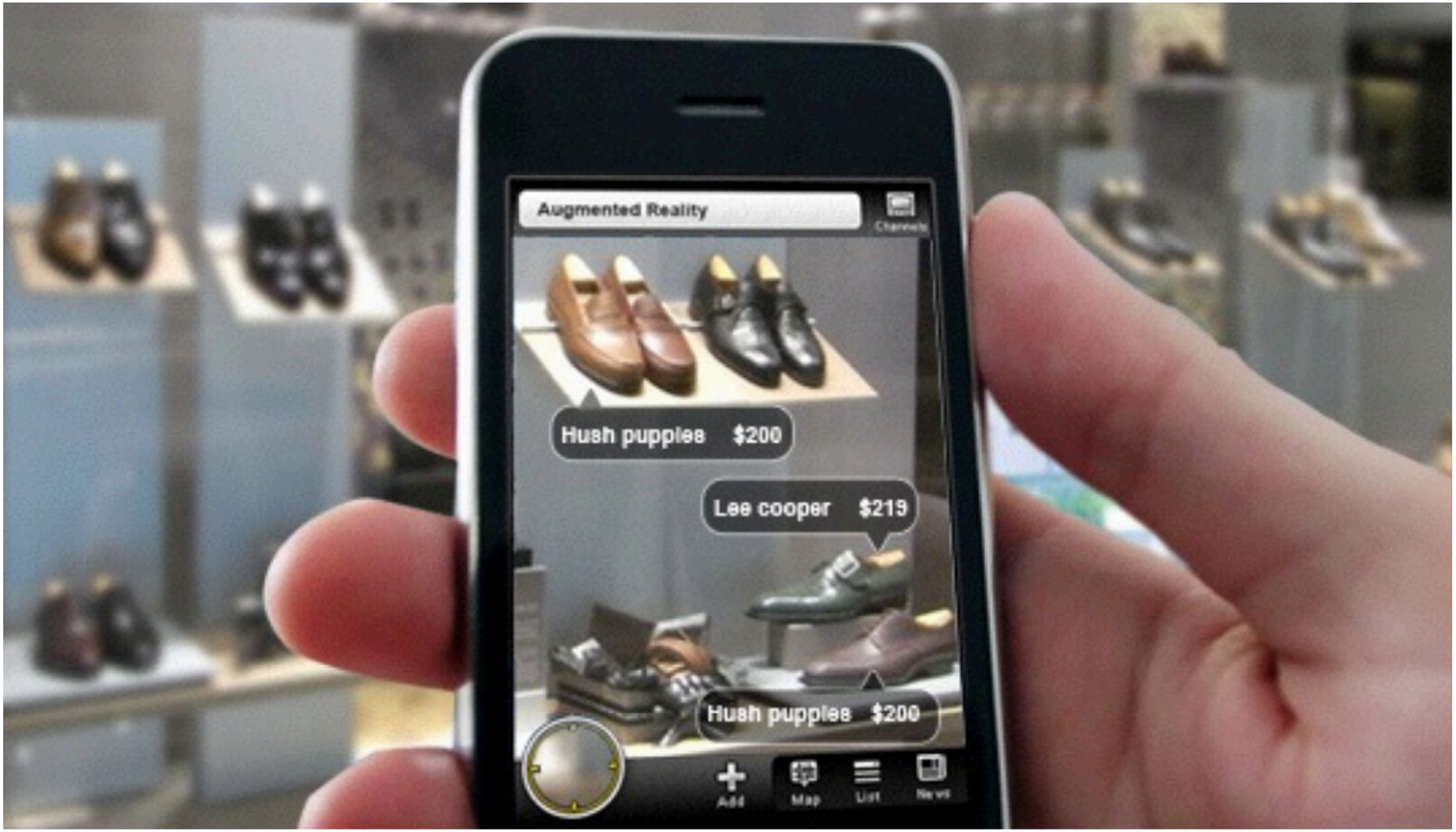


Een user interface van de Magic Leap via 2D-overlay.
(Bron: https://www.gamasutra.com/)

3D-mapping en zichtbare 3D via de Magic Leap.

Een gelijkaardige ervaring als de Magic Leap biedt de **Microsoft Hololens**. Het gaat om een soort HUD dat net als de Kinect via een camerasysteem lengte, breedte en diepte waarneemt. De Hololens reageert op gebaren, gesproken commando's en oogbewegingen.

Door de uitzonderlijke prestaties van toestellen als de Magic Leap en Microsofts Hololens, raakt stilaan meer en meer de term **mixed reality** in zwang.



Augmented reality en 2D-overlays op de smartphone.

Playstation VR voor PS4.
(Bron: http://fortune.com/2016/01/05/virtual-reality-game-industry-to-generate-billions/)

Boeing 737 Flight Simulator Experience in Toronto.
(Bron: https://breakawayexperiences.com/en/flight-simulator-toronto-experience)

Virtual Reality

Virtual reality (VR) dompelt de gebruiker helemaal onder in een soort virtuele gesimuleerde omgeving. We spreken van VR-brillen, maar in wezen gaat het om **HUD (head-up displays)** die het menselijk zicht volledig afsluiten van de echte omgeving. Dat is echter niet altijd het geval. Bij geavanceerde flight simulator neemt de gebruiker plaats in een soort van namaakcockpit. Bij **360* videos/ afbeeldingen** kan de gebruiker navigeren door een beeld. Maar uiteraard ziet hij daarnaast ook nog de echte werkelijkheid.

VR is een geavanceerde vorm van wat tal van musea al meer dan eeuw proberen nl. de bezoeker onderdompelen in een andere wereld, iets wat ook de meeste games trachten te doen. Virtual reality dompelt de gebruiker onder in een andere omgeving, een totale **immersie**. Je kan het begrip immersive interface dus op 2 manieren interpreteren: *eentje waarbij de UI opgaat in de werkelijkheid of een systeem waarbij de gebruiker opgaat in de virtuele omgeving.*

Diverse fabrikanten produceren VR-HUD's: *Oculus Rift, Google Cardboard, HTC Vive, Nokia OZO, Ricoh Theta, GoPro Odyssey, Bublcam, Giroptic...*

Leven we in een computer?

De **simulatiehypothese** stelt dat de zichtbare realiteit niet echt is, maar dat we leven in een computersimulatie. De "gesimuleerden" (wij mensen) zijn zich er totaal niet van bewust. Dit idee vormt de basis achter SF-films zoals The Matrix. Volgens de Britse computerdeskundige Nick Bostrom zijn er argumenten om aan te nemen dat dit wel degelijk het geval is:

"A technologically mature "posthuman" civilization would have enormous computing power. Based on this empirical fact, the simulation argument shows that at least one of the following propositions is true:

1. The fraction of human-level civilizations that reach a posthuman stage is very close to zero;

2. The fraction of posthuman civilizations that are interested in running ancestor-simulations is very close to zero;

3. The fraction of all people with our kind of experiences that are living in a simulation is very close to one.

If (1) is true, then we will almost certainly go extinct before reaching posthumanity. If (2) is true, then there must be a strong convergence among the courses of advanced civilizations so that virtually none contains any relatively wealthy individuals who desire to run ancestor-simulations and are free to do so. If (3) is true, then we almost certainly live in a simulation. In the dark forest of our current ignorance, it seems sensible to apportion one's credence roughly evenly between (1), (2), and (3). Unless we are now living in a simulation, our descendants will almost certainly never run an ancestor-simulation."

In The Matrix komt de hoofdrolspeler er achter dat de mensheid leeft in een volledig virtuele omgeving.

(Bron: https://www.konstantinfo.com/blog/wearables-become-desirables/)

Softwear en e-textile met de LilyPad Arduino-microcontroller.

Wearables

Wearables passen vooral in de gezondheidshype. Een wearable is een gadget dat op het lichaam wordt gedragen en informatie over je lichaam verzamelt. De eerder genoemde AR-contactlens is hiervan een geavanceerd voorbeeld. De meeste van die toestellen verzamzlen persoonlijke data over de gebruiker. Ze geven advies over en voor je gezondheid. Activity trackers zijn slimme toestellen die bijhouden hoeveel je beweegt en hoeveel calorieën je verbrandt. Sportwatches hebben nog extra functies: zoals het meten van je hartslag of je slaapritme,..

Maar de fabrikanten zijn vooral geïnteresseerd in de data die ze over de gebruikers kunnen verzamelen. Ze vormen een onderdeel van de big data die grote bedrijven als Google en Apple verzamelen en verkopen aan verzekeringsmaatschappijen en andere afnemers.

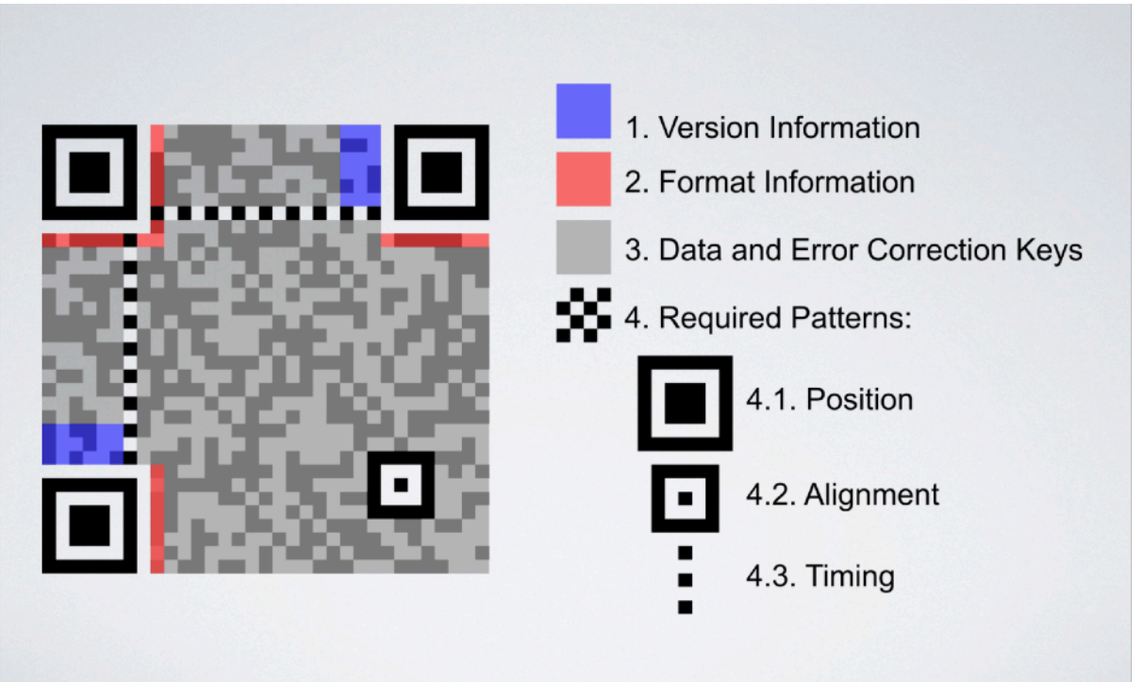
Met de **LilyPad Arduino** beschik je over een flexibele microcontroller om je kleding of textiel om te vormen tot slimme kleren (**e-textile**). Zelf spreken de ontwikkelaars in dit geval van **softwear**.

LOCATION AWARE SERVICES

Communicatie tussen twee apparaten is mogelijk als ze allebei dezelfde protocollen volgen: *infrarood, WiFi, 3G/4G/5G, GPS, Bluetooth*… WiFi maakt het mogelijk om binnen een gebouw de internetverbinding van een router te delen. Met bluetooth kan je bestanden uitwisselen tussen twee toestellen. Bluetooth wordt veelvuldig ingezet voor het afspelen van muziek op draadloze luidsprekers…

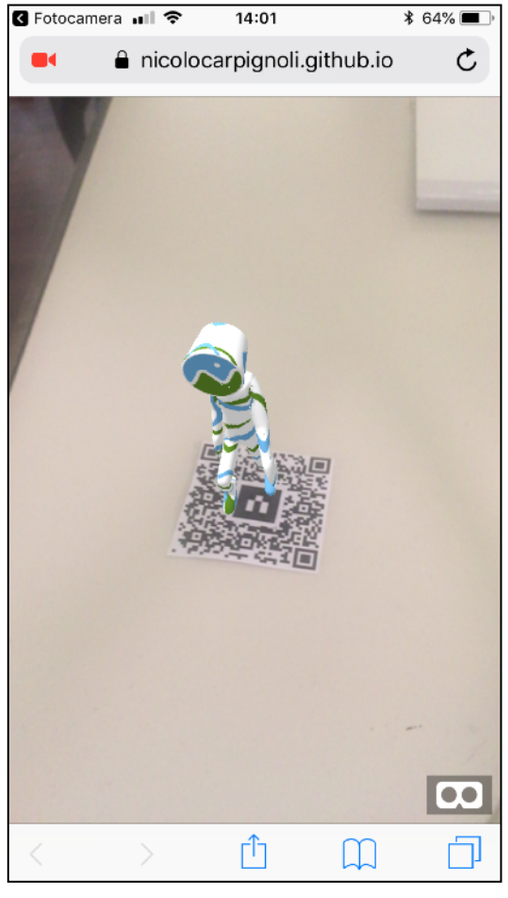
Een slim gebruik van technieken als *BLE, NFC, QR, RFID en beacons*… maakt **communicatie tussen fysieke objecten en digitale apparaten mogelijk**. *Zo kan je informatie uitwisselen tussen een boek en een website, betalingen uitvoeren of kan een kledingstuk in een winkel informatie verzenden naar je smartphone…* **Ze laten eveneens toe om objecten of een gebruiker te lokaliseren.**

Meer nog: je moet geen specialist zijn om er gebruik van te maken.



In een QR-code kan je nog een extra afbeelding of zelfs een ander type van "marker"-code opnemen. Als de gebruiker met zijn smartphone de QR-code scant, gaat hij naar de webpagina die de AR–app bevat. Vervolgens scant de camera de marker in het centrum van de QR-code. De AR-applicatie voegt nu een 3D-model toe.

(Bron: https://medium.com/chialab-open-source/how-to-deliver-ar-on-the-web-only-with-a-qr-code-e24b7b61f8cb)

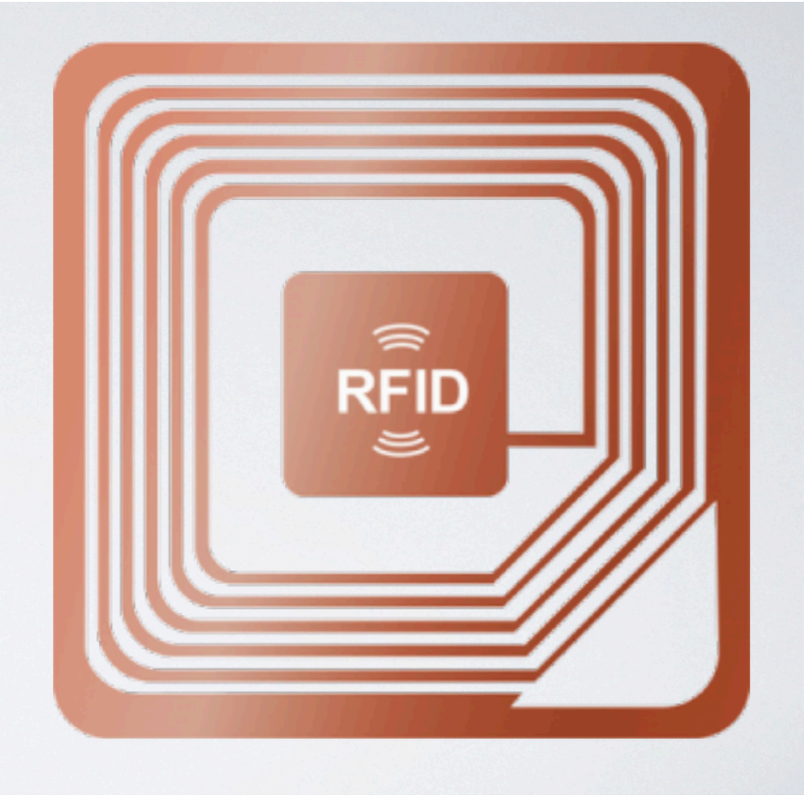


Smart communicatie met de fysieke wereld

De opkomst van de smartphone heeft ervoor gezorgd dat zo goed als iedereen een digitale assistent op zak heeft. Smartphones en andere wearables zijn doorgaans uitgerust met een hele resem aan **sensoren en communicatietechnologieën** die het uitwisselen van informatie zonder directe menselijke tussenkomst mogelijk maken.

Quick response

De camera van je smartphone is eveneens een scanner waarmee je documenten kan fotograferen. Met een gepaste app kan je de documenten via **OCR (optical character recognition)** zelfs weer omzetten naar bewerkbare tekst. Bij heel wat smartphones herkent de camera al automatische QR-codes. De QR (quick response) - code lijkt de opvolger van de streepjescode. Ze kan een prijs, maar ook tekstinformatie of een link bevatten. Banken gebruiken het bij hun mobiele apps om betalingen goed te keuren. Als een uitgever een QR-code toevoegt aan een boek, krijgt de lezer via de link in de QR-code toegang tot extra online informatie over het boek.



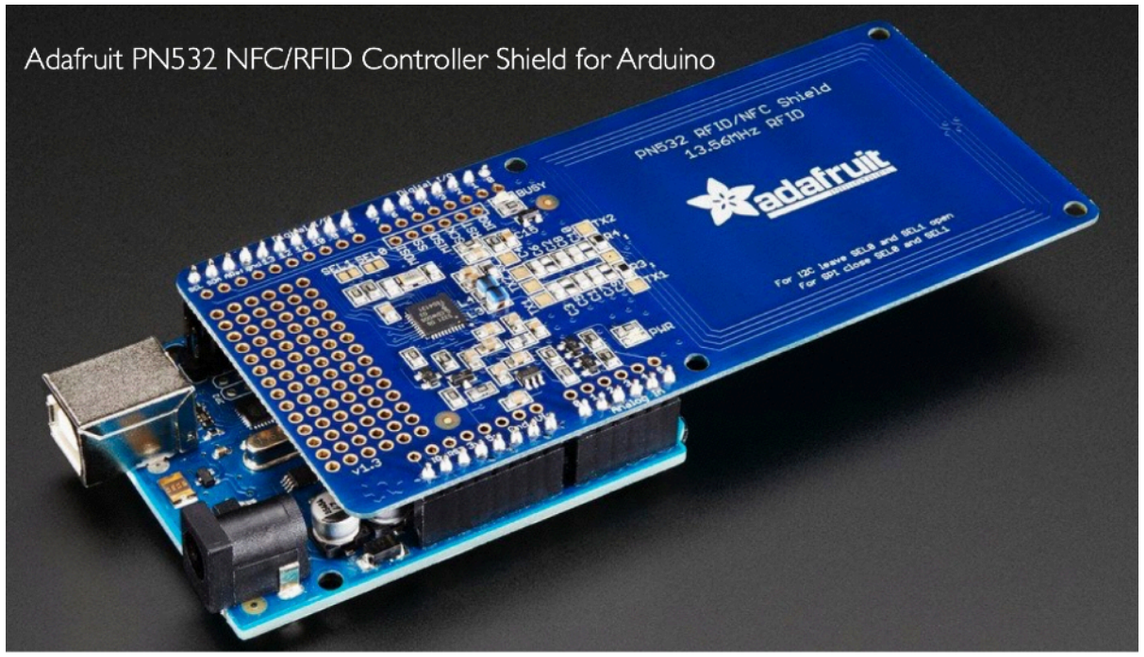
RFID-tags.

QR is een relatief eenvoudig te integreren techniek. Je hebt een stuk software nodig om de codes te genereren. De eindgebruiker heeft een camera-app nodig op zijn smartphone en de nodige software om de code uit te lezen.

Slimme markeringen: RFID, NFC

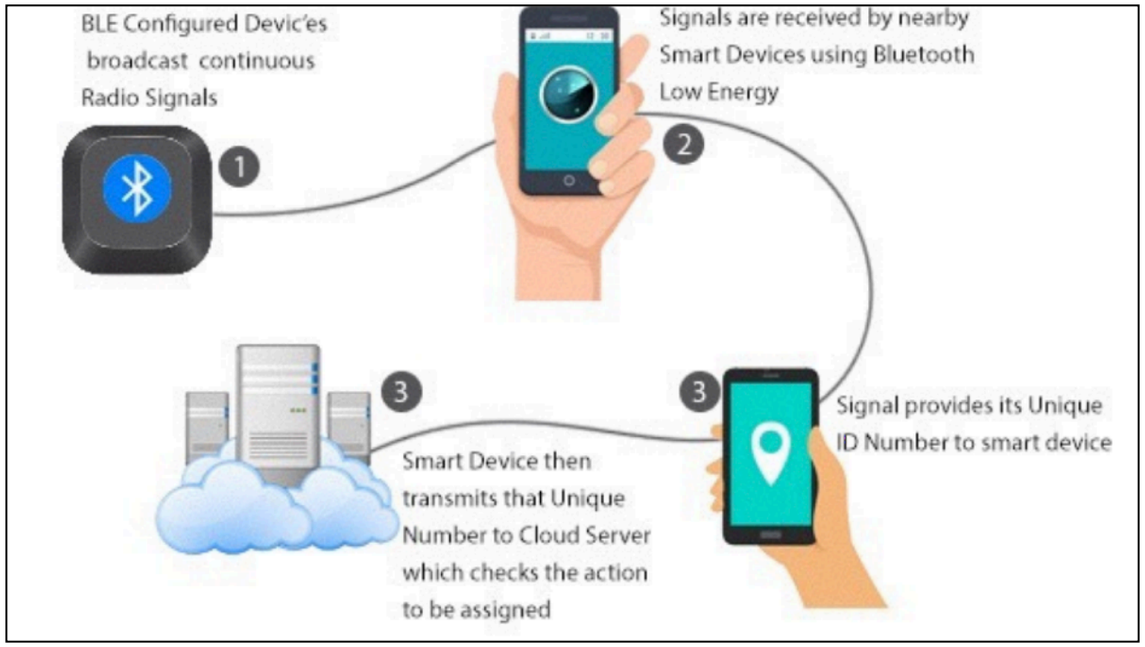
RFID, NFC en BLE daarentegen maken gebruik van radiogolven om informatie uit te wisselen.

RFID (radio frequency identification) is een techniek om fysieke objecten of zelfs levende wezens "uniek" te identificeren. Een volledig RFID-systeem omvat **een tag, een lezer (reader) en een antenne**. De lezer ('reader') stuurt een signaal uit via de antenne. De 'tag' antwoordt daarop met de unieke informatie (bijvoorbeeld een productcode, een prijs… Actieve RFID-tags beschikken over een eigen "energievoorziening" en kunnen zelf radiosignalen uitzenden tot ongeveer 100 m. Passieve RFID-tags krijgen hun energie van de elektromagnetische golven die de RFID-lezer uitzendt. Omdat de radiogolven sterk genoeg moeten zijn, werken passieve tags enkel goed op beperkte afstand (direct contact tot ongeveer 25 meter).



NFC (near field communication) is een gespecialiseerd "familielid" van de RFID-technologie. Een NFC-toestel kan zowel een reader als een tag zijn. NFC-apparaten moeten zich (zoals de naam doet vermoeden: 'near') dicht bij elkaar, op een paar centimeter afstand, bevinden. Daarom gebruiken winkels het graag voor contactloos betalen. Je moet je bankkaart enkel nog tegen de betaalautomaat houden. Veilige uitwisseling van informatie is het hoofddoel van NFC.

NFC vindt ondertussen zijn weg naar de nieuwe generatie smartphones. Als je twee NFC-smartphones in elkaars buurt houdt, kunnen ze foto's uitwisselen (door bijvoorbeeld te 'swipen'). Maar het is ook mogelijk om bijvoorbeeld affiches of posters uit te rusten met een NFC-tag om informatie uit te wisselen met de smartphones van de consumenten. Audioapparatuur dat uitgerust is met NFC kan muziek afspelen door simpelweg je smartphone er op te leggen. Samengevat: NFC (Near Field Communication) is een draadloze manier om kleine hoeveelheden informatie uit te wisselen binnen een straal van 10 centimeter. De chip in je smartphone communiceert met een ander NFC-apparaat, zoals betaalsystemen, luidsprekers of andere smartphones,



Indoor plaatsbepaling

Beacons of iBeacons zijn kleine bluetoothzenders (bluetooth maakt gebruik van radiogolven). Een beacon ziet eruit als een klein doosje en bevat een **Bluetooth Low Energy (BLE)-chip** en een batterij. Elke beacon zendt onophoudelijk een uniek nummer uit. Een smartphone met BLE-ondersteuning kan dit signaal opvangen. Als je weet waar welke beacon staat, kan je binnen een gebouw of winkel te weten komen waar iemand zich bevindt. Beacons zijn dus een ideale manier van indoor positiebepaling, want daar is GPS minder bruikbaar.

Apple omarmt BLE als alternatief voor NFC. Terwijl NFC slechts op enkele centimeters werkt, reikt BLE tot zelfs 70 meter. Belangrijk voordeel is dat er geen ingewikkelde installatie bij komt kijken. Je kan ze plaatsen waar je wil. Een iOS- of Android-toestel weet wie jij bent. In combinatie met beacons weet de smartphone-app nu ook waar je je bevindt. Als we even niet stilstaan met de gevolgen voor de privacy, dan schept deze technologie heel wat nieuwe mogelijkheden: *Indoor plaatsbepaling en navigatie, Sales promotion (in-store promotie), Loyalty (klantenkaarten), Toegangscontrole (alternatief voor toegangspassen), Transacties (contactloos betalen)…*

USER INTERFACE & USER EXPERIENCE DESIGN

Bij het ontwikkelen van een applicatie of een stuk software bevind je je op het knooppunt tussen het digitale systeem en de eindgebruiker. Aan de kant van de machine moet je kennis hebben van grafisch ontwerp, besturingssysteem, programmeertalen en ontwikkelomgevingen. Aan de menselijke kant komen grafisch ontwerp, communicatietheorie, taal, sociale wetenschappen, psychologie, emoties... om de hoek kijken. Daarom is het ontwerp van een interface zelden het werk van één enkel persoon.

Waar moet je allemaal rekening mee houden bij het ontwerpen van een interface? Hoe kan je zowel rekening houden met de verwachtingen van de opdrachtgever, als met de eindgebruiker? Bij het ontwerpen van een interface spreken we daarom van UX-design: de gebruiker staat centraal (user experience), want de “klant is koning”.

Modulariteit

Hardware is op zich een sterk voorbeeld van modulair ontwerp. Wie wat overweg kan met computeronderdelen kan zijn eigen toestel modulair samenstellen. Een externe harde schijf, een USB-stick, een printer/scanner... kan je beschouwen als afzonderlijke modules. Ook moderne programmeertalen (zogenaamde object-georiënteerde talen) werken modulair, met kleine herbruikbare stukjes code. Software en apps op tabletcomputers en smartphones kan je beschouwen als apart te installeren modules. Een modulair ontwerp maakt het voor de gebruiker mogelijk om zijn digitaal systeem uit te breiden of te wijzigen.

Schaalbaarheid

Modulariteit betekent echter niet automatisch dat we een systeem of een interface onbeperkt kunnen schalen. *Iets wat functioneert op kleine schaal werkt bijvoorbeeld niet noodzakelijk op een grote schaal.*

Een touchscreeninterface werkt fijn op een smartphone of een tablet, maar op een computer met een groot beeldscherm is het niet meteen een gebruiksvriendelijke interface. Een toetsenbord werkt handig op een laptop of computer, maar is behoorlijk onhandig op een mobiele telefoon.

Bij webdesign staat **responsive webdesign** centraal: een website moet zowel op een mobiel toestel als op een computer perfect weergegeven worden: een website moet "schalen" naargelang de schermbreedte.

Toegankelijkheid

Toegankelijkheid blijft niet beperkt tot een systeem bruikbaar maken voor mensen met een beperking. Iedere gebruiker heeft baat bij een toegankelijk systeem. Een interface dient zo ontworpen te worden dat hij vlot toegankelijk is voor iedereen. **Bedieningsgemak** en eenvoud in gebruik zijn belangrijke normen. Een goed ontworpen interface zorgt ervoor dat gebruikers zo min mogelijk fouten kunnen maken. Toegankelijkheid en **gebruiksvriendelijkheid** hangen af van een heleboel factoren.

Een gebruikersinterface moet eerst de lagere behoeften van de gebruiker bevredigen, daarna pas de hogere. We hebben het hier uiteraard niet over voedselvoorziening, maar over de **functionaliteit** van een digitaal systeem.

Wat wil de gebruiker?

Behoefte	Omschrijving
Functionaliteit	Bepaalde functies mogelijk maken.
Betrouwbaarheid	Een videorecorder moet echt opnemen als je op de "record"-knop drukt.
Gebruiksgemak	Je kan dingen makkelijk doen.
Bekwaamheid	Je kan dingen beter doen dan voordien.
Creativiteit	Alle behoeften zijn bevredigd. Indien je dit niveau bereikt, verkrijg je van de gebruikers soms een loyaliteit met cultachtige status (vb. Apple).

Bij het ontwerp kies je niet tussen flexibiliteit (veel mogelijkheden) en gebruiksgemak. Je moet zelf afwegen welk onderdeel voor jouw ontwerp of interface het meest doorweegt.

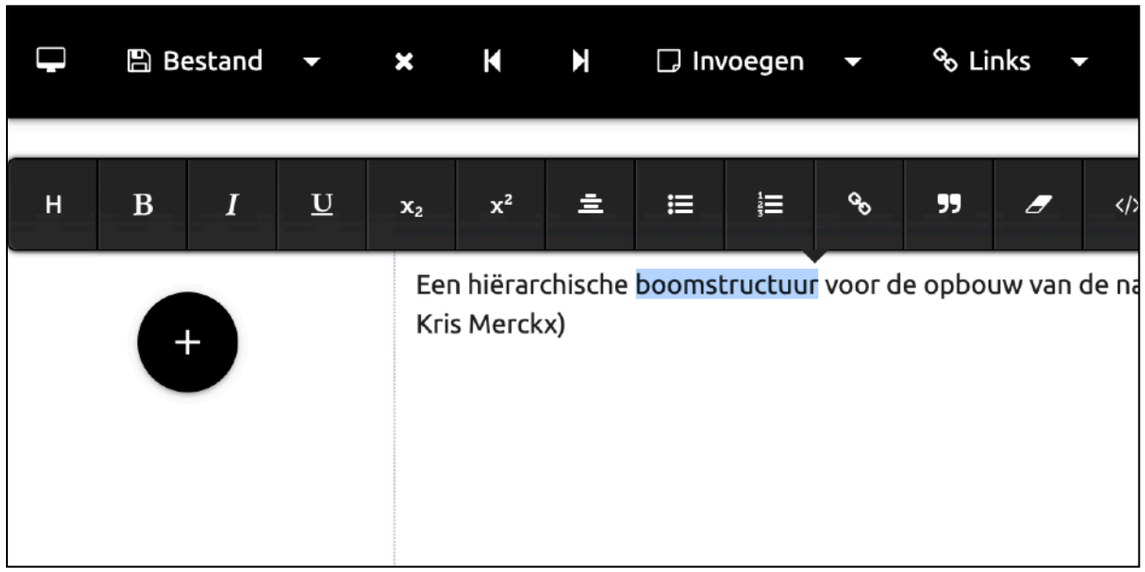
Een flexibel ontwerp biedt de gebruiker meer mogelijkheden, maar is minder efficiënt. Een vaak gemaakte fout is veronderstellen dat flexibiliteit boven gebruiksgemaak gaat.

Bijvoorbeeld:
Bij een PC zijn veel mogelijkheden bij aankoop nog onbekend. Een student die de richting multimedia start, heeft geen idee waarvoor hij/zijn zijn toestel een paar maand later allemaal zal gebruikt hebben. De verkoper/ontwerper moet in dit geval anticiperen op het toekomstig gebruik.

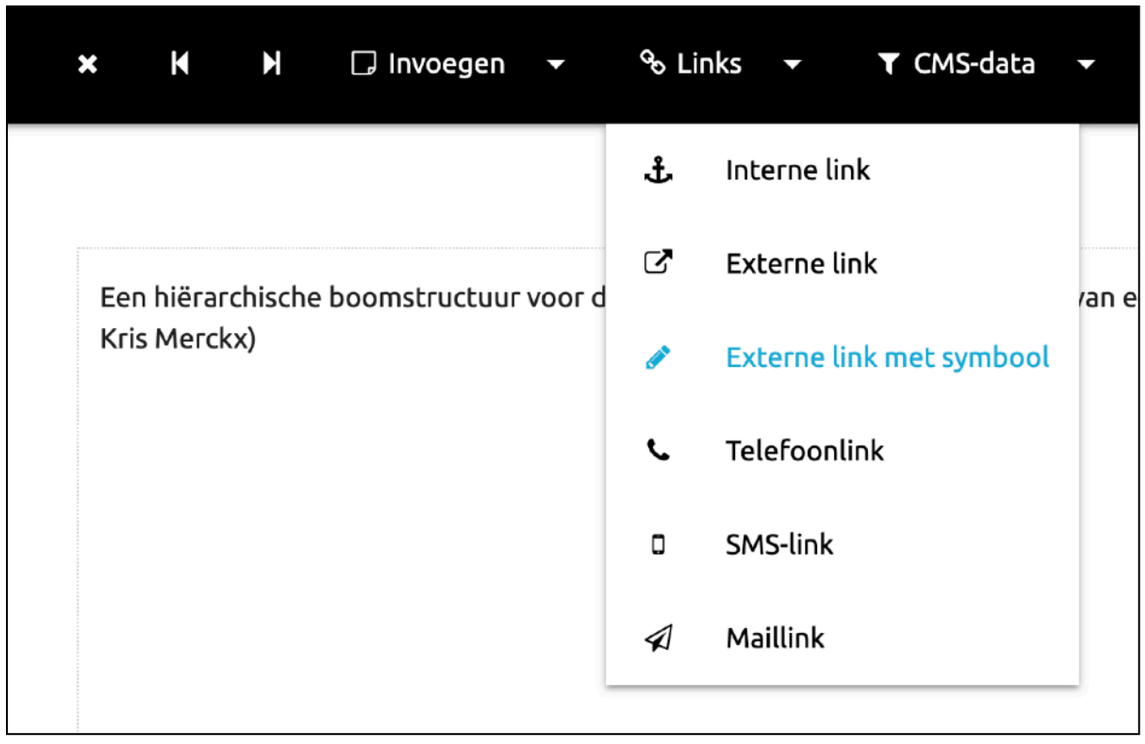
Als het publiek inzicht heeft in wat het wil, heeft specialisatie voorkeur. Voor iemand die klaar en duidelijk weet dat hij enkel af en toe wat wil surfen op het internet, volstaat een tabletcomputer. Wie niet precies weet waarvoor hij een computer allemaal zal inzetten, mikt op flexibiliteit.

Flexibiliteit betekent echter niet dat we gebruiksgemak uit het oog moeten verliezen. Eenvoud in gebruik heeft steeds de voorkeur boven complexiteit. Volg het **KISS-principe** (keep it simple and smooth of keep it simple and stupid). Twijfel je bij het ontwerp van een interface tussen een simpele oplossing en een meer complexe omdat die bijvoorbeeld intelligenter overkomt, kies dan altijd voor de simpele (**de regel van "Ockhams scheermes"**).

Als een gevolg overeenkomt met de verwachting dan zit de **mapping** goed. Wanneer een knop of een reeks acties of commando's in een digitaal systeem precies leidt tot wat de gebruiker verwacht, dan zit het gebruiksgemak goed. Plaats de bedieningsonderdelen (knoppen, sliders...) zodanig dat mensen weten wat er zal gebeuren als ze er gebruik van maken.



De opmaakfuncties verschijnen wanneer je tekst selecteert.
(Bron: Crowl CMS, Kris Merckx)



Extra functies zitten verstopt achter een uitklapmenu.
(Bron: Crowl CMS, Kris Merckx)

Functionaliteit

Hoe meer opties een gebruiker krijgt, hoe langer de benodigde tijd om een beslissing te nemen (**wet van Hick**). Dit wil niet zeggen dat je de mogelijkheden van een interface volledig moet afbouwen en de mensen enkel voorzien van een simplistische interface met nagenoeg geen mogelijkheden. Heel wat software begrenst de mogelijkheden afhankelijk van de kennis en de ervaring van de gebruikers. De gebruiker krijgt dan via een knop toegang tot een aantal geavanceerde opties. In games vorderen gebruikers stapsgewijs doorheen verschillende niveaus (levels).

De 80/20-regel: Gebruikers benutten volgens schatting 80% van de werktijd slechts 20% van de functies in een UI. Bij het ontwerp van een interface moet je je dan ook richten op die 20%. Niet-kritische (niet noodzakelijke) functies die behoren tot de 80%, beperk je tot een minimum of je gooit ze helemaal weg. Ondermeer om die reden zit bij software veel functionaliteit verborgen achter uitklapmenu's.

Waarom hebben interfaces niet altijd het verwachte succes?

Het is niet omdat iets beter "presteert", dat de voorkeur van de gebruiker daarnaar uitgaat. Het succes van een applicatie, product of interface hangt niet alleen af van de eigen eigenschappen. De firma General Magic ontwikkelde in de vroege jaren 1990 een smartphone met toegang tot de cloud, maar kende geen succes. Dezelfde ontwikkelaars stonden later aan de basis van de iPod en iPhone, maar ook van Android en eBay en kenden toen wel succes. Het falen van een dienst of product kan van heel veel factoren afhangen. Goede producten kunnen dus mislukken en omgekeerd kunnen 'slechte' producten heel succesvol worden.

Herkenning gaat boven herinnering

In de eerste computerinterfaces gaf de gebruiker commando's via een tekstinterface. Je moest de instructies of commando's goed onthouden om de computer te vertellen wat je precies wou doen. In een GUI zitten commando's achter knoppen verstopt. Dat maakt het gebruik veel eenvoudiger. De gebruiker "herkent" de commando's en hoeft ze niet meer te onthouden. Houd hier rekening mee wanneer je een interface bouwt.

Wanneer de gebruiker te veel stappen moet onthouden, zal hij sneller geneigd zijn hulp in te roepen. Maak de gebruiker desnoods bij elke stap duidelijk wat de volgende stap is. **Beperk je tot 5, maximaal 7 stappen.**

De in besturingssystemen aanwezige API's maken het voor programmeurs mogelijk om herkenbare dialoogvensters op te roepen vanuit hun eigen software.

Maak zelf ook gebruik van herkenbare symbolen en de plaats waar je ze geeft op het computerscherm. Hoe herkenbaarder, hoe groter het gebruiksgemak.

Belangrijke tip:
Ontwikkel een interface op zo'n manier dat een gebruiker geen handleiding of uitleg meer nodig heeft. Maar voor heel geavanceerde software (zoals Adobe Photoshop) is dit natuurlijk niet altijd mogelijk. In sommige gevallen primeert functionaliteit op eenvoud.

Een voorbeeld van "chunking", waarbij het aantal items beperkt blijft tot 7. Extra nieuwsberichten achter een "meer"-knop.
(Bron: www.demorgen.be)

Mooi werkt beter

Esthetische ('mooie') ontwerpen wekken de indruk dat ze gebruiksvriendelijker zijn, ook al is dit niet automatisch zo. Gebruikers zijn toleranter tegenover ontwerpfouten als het ontwerp er esthetisch goed uitziet. Esthetische ontwerpen worden sneller geaccepteerd, en wekken de indruk dat ze gebruiksvriendelijker zijn. Uiteraard blijft "schoonheid" en "esthetiek" een zeer subjectief gegeven, maar de perceptie is op dit vlak moeilijk te wijzigen. Denk aan het idee dat veel Applegebruikers hebben over de producten van hun favoriete merk. Wie aan Apple denkt, denkt aan mooi design, gebruiksvriendelijkheid, eenvoud in gebruik, stabiliteit... ook al klopt dat niet altijd. Voor concurrenten is het bijzonder moeilijk om die *perceptie* voor de eigen productlijn op te wekken.

Grafische vormgeving

In het hoofdstuk "How to become a smartist" leer je waar je allemaal rekening moet mee houden bij de **opmaak** van een interface. Je kan de leesbaarheid verhogen door het juiste **lettertype** te kiezen. Niet alle **kleuren** passen bij elkaar. Maar ook **uitlijning** van elementen en **symmetrie** zijn van belang. Gebruikers ervaren symmetrische ontwerpen als mooier en makkelijker.

Belangrijke tip:
Een ontwerp moet er goed uitzien. Ontwikkelaars moeten niet neerkijken op het belang van de vormgeving. Anderzijds moet het product of de applicatie voldoende functies bieden. Beide zijn even belangrijk. Het ene kan niet zonder het andere.

Technieken om beter te onthouden

Hoe moest ik dit nu weeral doen? Vaak vergeten gebruikers hoe ze een bepaalde functie in een interface moesten doen of waar ze iets konden terugvinden. Met kleine aanpassingen kan je ervoor zorgen dat informatie beter "bestendigd" (blijft hangen) in het geheugen van de gebruiker.

Een aantal tips:

- Mensen kunnen **5 tot 7 items** makkelijk in één keer memoriseren. Als je een navigatiebalk bouwt, beperk je je best tot maximaal 7 knoppen.
- Gebruikers onthouden items aan het begin van een lijst (een reeks slides in de presentatie, een lijst met knoppen...) het best, die in het midden het minst. Naar het einde van de lijst toe, blijven items weer beter in het geheugen hangen (=**Seriële positie-effect**).
- **Nabijheid en gelijkenis**: Gebruikers ervaren elementen die dicht bij elkaar liggen als gerelateerd, en elementen die elkaar overlappen als "gemeenschappelijk". Knoppen die op elkaar gelijken, lijken eerder bij elkaar te horen. Hoe dicht bepaalde elementen bij elkaar staan, bepaalt dus in sterke mate hoe ze door de gebruiker ervaren worden. Zorg ervoor dat benamingen vlak bij elementen staan en niet in afzonderlijke "legenda" (zoals bijvoorbeeld bij grafieken in MS Excel).
- Splits tekstinhouden op in kleinere onderdelen (uiteraard op een zinvolle manier) en je gebruiker zal de inhouden beter kunnen memoriseren. De techniek om info op te splitsen in kleine eenheden (ook niet te klein), noemt men **chunking**.
- Soms helpt het om informatie in stapjes zichtbaar te maken. **Stapsgewijze informatieverstrekking (progressive disclosure)**: kan de complexiteit van een toepassing aardig verlagen. *Een slideshow, 'lees meer'-knop, uitklapmenu's, leerpaden (in bijvoorbeeld e-learningomgevingen), tabbladen...* verhogen op die manier het leerrendement.
- Welke momenten uit je schooltijd heb je het best onthouden? Niet zo zeer de leerstof, maar vooral de anekdotes. Opvallende dingen onhoud je nu eenmaal beter dan gewone dingen. Dit heet het **"Van Restorff"- effect**. Wil je elementen benadrukken, zorg dan dat ze opvallen. Gebruik eventueel ongewone beelden, speel met het tekstbeeld...

Zeggen beelden meer dan tekst?

"Een beeld zegt meer dan duizend woorden," hoor je vaak. Mensen met een zogenaamd "visueel geheugen", zouden er voordeel uit halen. Maar er is weinig wetenschappelijk bewijs voor het bestaan van iets als een "visueel" of "auditief" geheugen. Meer afbeeldingen leiden echter niet noodzakelijk tot een "betere herinnering". In het ontwerp van user interfaces en presentatie gaat de voorkeur uit naar een **combinatie van tekst en afbeelding**.

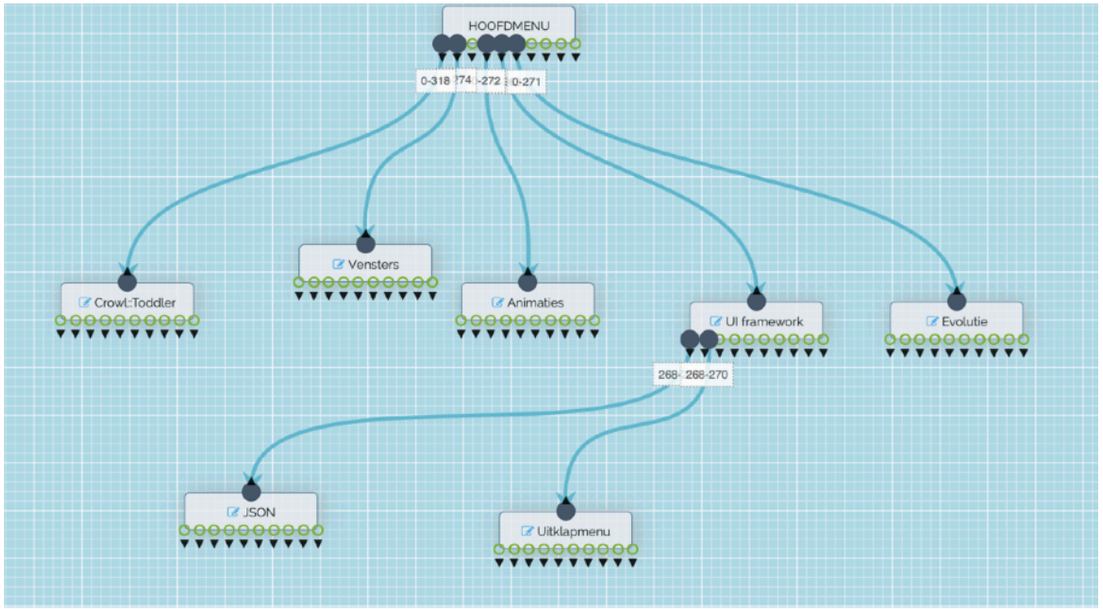
Een aantal tips:

Als je relaties en verbanden tussen diverse elementen wil aantonen, probeer dit dan te visualiseren met **netstructuren, lagen, boomstructuren**... **Grafieken** kunnen helpen bij de visualisatie van cijfergegevens, maar dat is niet altijd zo. Het kan ook leiden tot overdaad.

De meeste GUI's maken veelvuldig gebruik van **pictogrammen**. De metrointerface van Microsoft Windows is daar gedeeltelijk van afgestapt. Het gebruik van schermpictogrammen als *symbolische links naar bepaalde programma's of bestanden* is reeds aanwezig van bij de eerste "graphical user interfaces" (Xerox).

Overdrijf niet...

Houd een interface of voorstelling overzichtelijk. *Overlaad een interface niet met te veel beeld of animatie.* Alles wat de ontvanger te veel afleidt van de eigenlijke boodschap vormt ruis. Een goede interface zorgt voor betere communicatie met de eindgebruiker. Richt je daarom op het "signaal" en beperk de "ruis".



Een hiërarchische boomstructuur voor de opbouw van de navigatiestructuur van een website. Vijf items op het eerste niveau.
(Bron: Crowl CMS, Kris Merckx)

Herkenning

Een deur met een “klink” nodigt uit om aan te trekken, niet om tegen te duwen. Als je de klink vervangt door een vlakke metalen plaat, weet de gebruiker meteen dat hij/zij tegen de deur moet duwen om binnen te komen. Die vorm van “uitnodiging” noemt men **affordance**. Als de affordance van een object om omgeving goed overeenkomt met de bedoelde functie, dan verhogen het gebruiksgemak en de prestaties.

Bij het ontwerp van software-interfaces gebruikt men vaak afbeeldingen van fysieke objecten om de functie van de interface-elementen te verduidelijken. Het programma Garageband van Apple gebruikt bijvoorbeeld herkenbare knoppen van gitaarversterkers om de affordance te verhogen.

Die techniek waarbij men afbeeldingen van bestaande toestellen of onderdelen ervan gebruikt binnen een digitale omgeving, heet **skeuomorfisme**. In wezen kan je heel wat "metaforen" die men bij het ontwerpen van interfaces gebruikt, skeuomorfismen noemen: *de prullenmand, het bureaublad*. De graad waarin een ontwerper de overeenkomst met fysieke objecten doorvoert kan echter meer of minder prominent zijn.

Steve Jobs en Apple-ontwerper Scott Forstall waren grote voorstanders van skeuomorfisch ontwerp. Na de dood van Steve Jobs en het ontslag van Forstall, kreeg John Ive, een groot tegenstander van skeuomorfisme, de leiding over de "look and feel" van iOS7. Tegenstanders van skeuomorfisme argumenteren dat skeuomorfisme enkel een "overgangsfase" was. Vroeger zag de interface van een camera-app er nog uit als de knoppen van een analoog fototoestel. De meeste huidige gebruikers hebben echter nooit een analoog fototoestel gebruikt.

Skeuomorfisme in Garageband.
(Bron: Apple, Garageband)

Metaforen en analogieën

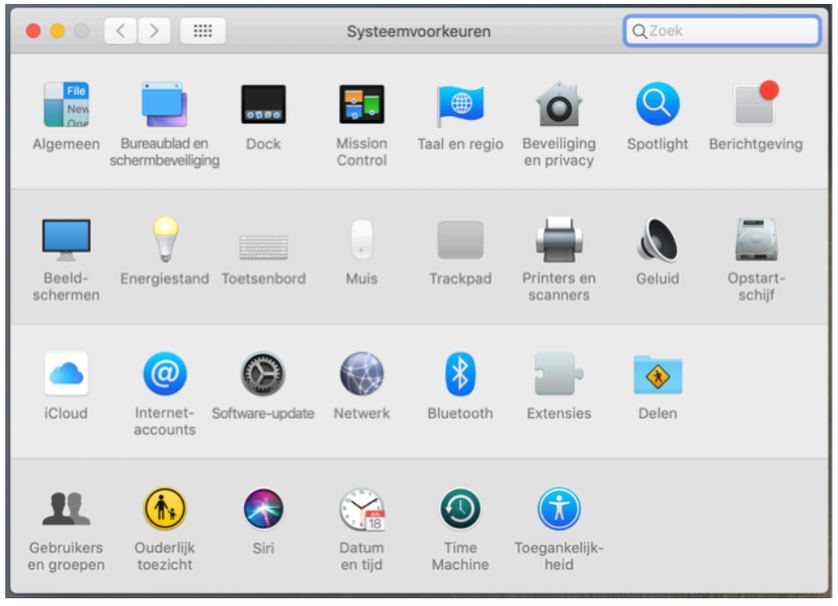
Hoe sterk tegenstanders van skeuomorfisme ook hun best doen, een digitale interface blijft op één of andere manier wel steeds schatplichtig aan zijn analoge voorgangers. We spreken nog steeds van een bureaublad of desktop, ook al ervaart lang niet iedereen dit meer als een digitale versie van een "echt" bureaublad. Een map (Engels: folder) heet nog steeds een map. Een "prullenmand" blijft nog steeds aanwezig in zo goed als elke GUI.

In heel wat applicaties blijven ontwikkelaars zich bedienen van metaforen en analogieën met "analoge voorgangers". Adobe Photoshop gebruikt net zoals een massa andere digitale beeldbewerkingsapplicaties virtuele "lagen" waarmee je als gebruiker composities kan samenstellen, net zoals tekenfilmmakers vroeger elk beeld opbouwden door afzonderlijke figuren op celluloidlagen boven elkaar te plaatsen. De tijdlijn in videomontagesoftware is nog steeds een virtuele weergave van een klassiek filmspoor. Gelijkenissen en metaforen zorgen voor herkenbaarheid en dat is misschien maar goed ook.

Nabootsing of mimicry

In de natuur bootsen wandelende takken een echte "tak" na om zich te verbergen voor hun vijanden. Op dezelfde manier bootsen interface-ontwikkelaars fysische objecten of hun functionaliteit na om de herkenbaarheid te verhogen. Een klassiek telefoontoestel en zelfs een iPhone bootst in zijn toetsenbord het toetsenbord van een rekenmachine na. Die nabootsing kan op drie niveaus gebeuren:

Nabootsing van uiterlijk	Het uiterlijk lijkt op iets anders.	Het pictogram van een map.
Nabootsing van gedrag	Het ontwerp gedraagt zich zoals iets anders.	Een robot die zich beweegt zoals een insect.
Nabootsing van functie	Het ontwerp werkt zoals iets anders.	Toetsenbord van een rekenmachine bij een oude Nokia-GSM.



Herkenbare pictogrammen vergezeld van een label. (Bron: Mac OS X)

Voorkom dat gebruikers fouten maken

Een interface moet voorkomen dat de gebruiker fouten kan maken én bij fouten duidelijke feedback geven.

Een aantal tips:

- **Vraag de gebruiker om te bevestigen:**
  Door bevestiging te vragen, kunnen onopzettelijke en ondoordachte handelingen voorkomen worden. Overdrijf hier echter niet in, want anders krijgen gebruikers het *"ben je hier wel zeker van"*-gevoel. Verificatie is enkel noodzakelijk indien het beslissingen nadien niet meer kunnen aangepast worden. In het bovenstaande NMBS-voorbeeld had de site kunnen vragen: "U staat op het punt tickets te bestellen voor een dag die bijna afgelopen is. Bent u hier zeker van?"

- **Restricties:**
  Door stapsgewijze informatieverstrekking kan het aantal potentiële fouten tot een minimum herleiden. Zulke fysieke restricties (begrenzingen, paden) laten de gebruiker toe bepaalde functies enkel binnen een bepaalde zone uit te voeren.

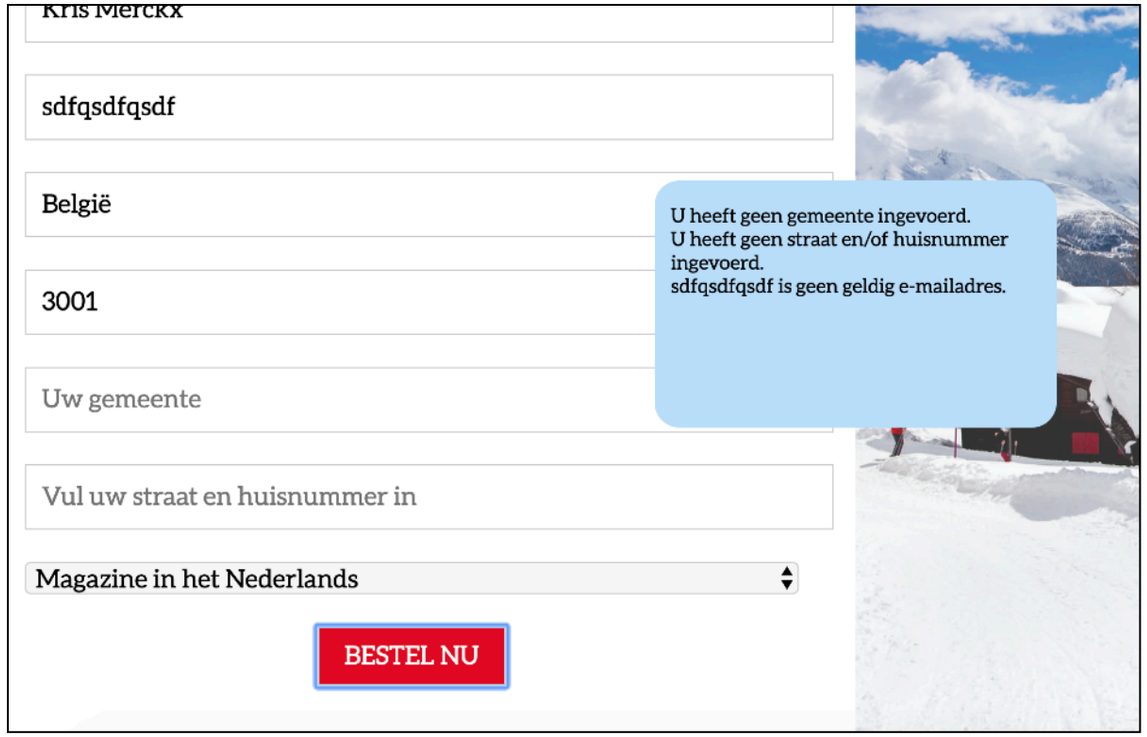
- Daarnaast kan je het foutief gedrag van eindgebruikers voorkomen door een reeks psychologische **restricties**:

Symbolen	Waarschuwing, icoontjes, pictogrammen… kunnen gebruikersgedrag beïnvloeden.
Conventies	rood is gevaar, geel is waarschuwing…
Mapping	Welke handelingen zijn mogelijk? De zichtbaarheid, het uiterlijk, de plaatsing van de interface-elementen… spelen hier een rol.

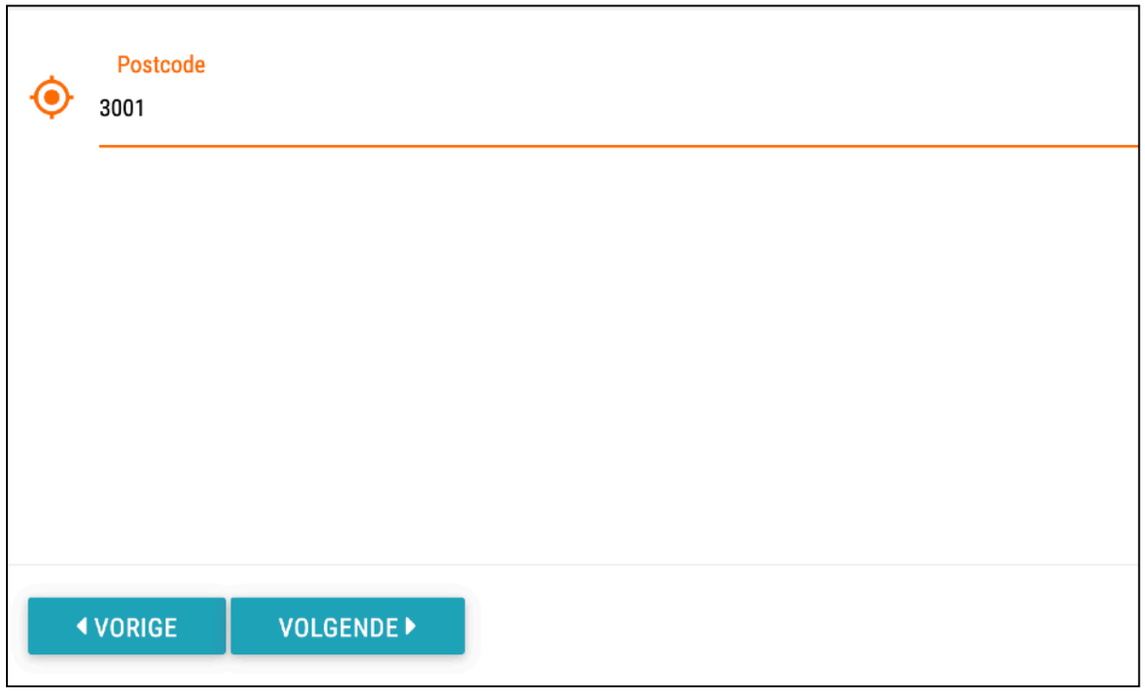
- **Garbage in, garbage out:**
  Als je fouten kan invoeren, krijg je foute output. *Bijvoorbeeld: In een invoerveld voor een leeftijd mag je enkel de mogelijkheid hebben om cijfers in te tikken.* Foutcontrole voorkomt foute output. Als UI-ontwerper moet je voorkomen dat troep wordt ingevoerd.

- **De wet van Fitts en inspanningsbelasting:**
  Hoe verder een doel verwijderd is, hoe langer het duurt om het te bereiken. Hoe sneller de beweging (van de cursor) en hoe kleiner het doel, hoe groter de foutenmarge (door snelheid en accuraatheid). Het menu dat tevoorschijn komt als je op de rechtermuisknop drukt, staat meteen bij het doel. Die beperkt het aantal potentiële fouten bij de invoer of bewerking van gegevens. Hier geldt eveneens de inspanningsbelasting: *hoe meer inspanning vereist om doel te bereiken, hoe meer kans op fouten.*

- **Operante conditionering:**
  Operante conditionering is een techniek om gewenst gedrag positief te bevestigen/belonen en ongewenst gedrag te bestraffen. Het duikt vaak op in leeromgevingen (e-learning) en games.



Fouten beperken door de gebruiker bij te sturen met feedback.
(Bron: www.100snowmagazine.be)



Stapsgewijze informatieverstrekking en "chunking".
(Bron: Crowl CMS Form Builder, Kris Merckx)

Storytelling

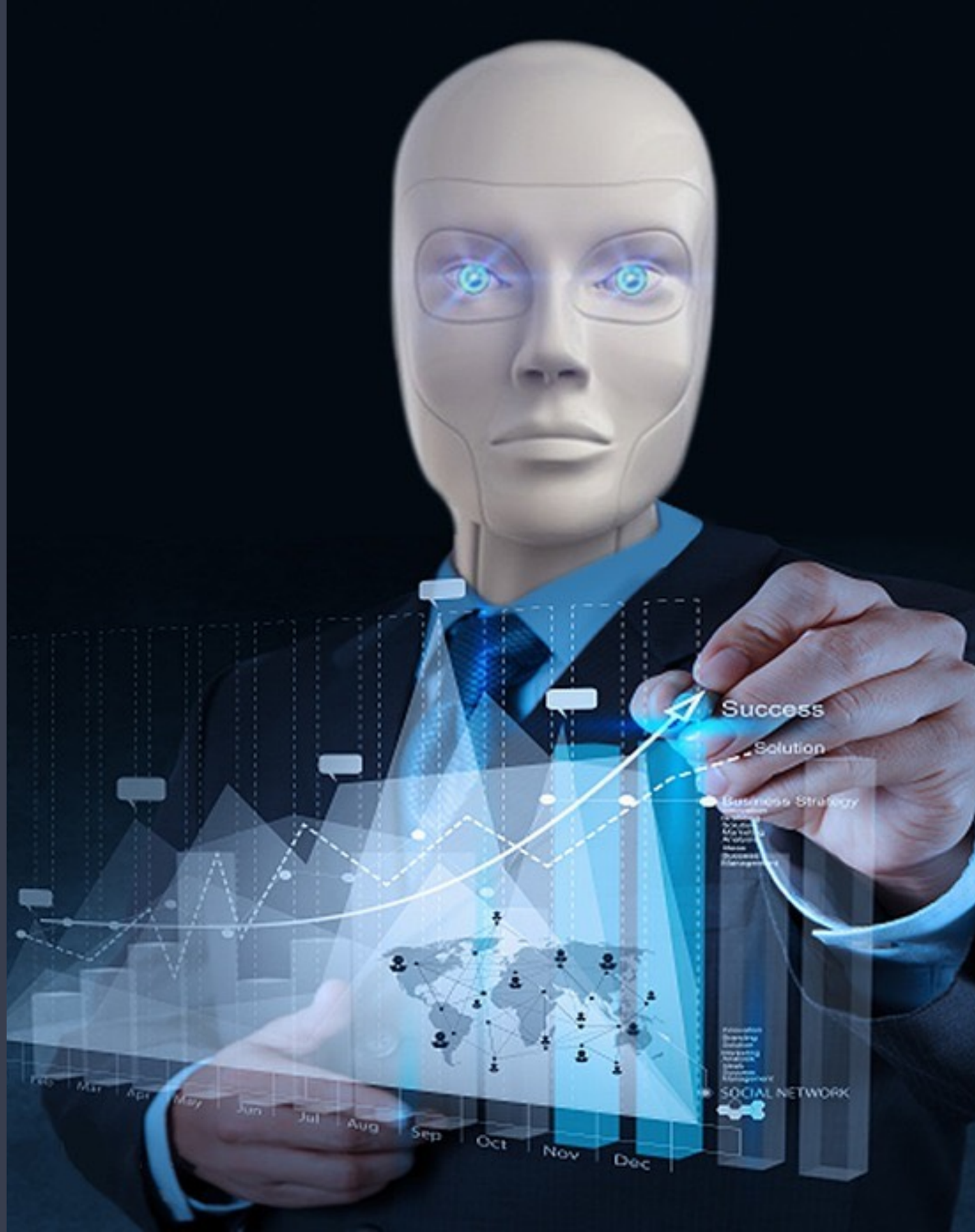
Een goed en herkenbaar verhaal, een pittige anekdote, een pakkend emotioneel verhaal... zorgt voor een gevoel van herkenbaarheid. Bij de presentatie van een product, een film of animatie, een presentatie… trekt een sterk verhaal de aandacht van de gebruiker. Geef de potentiële gebruiker het gevoel dat hij iets mist en dat jouw applicatie of product dit probleem oplost.

User experience

Hoe de eindgebruiker de interface of het product emotioneel aanvoelt, bepaalt mee het succes van het ontwerp. Bij UX-design werken professionals met heel verschillende achtergronden samen aan één project: grafici, UI-designers, programmeurs, financiële experts… Ze moeten rekening houden met grafisch ontwerp, technische mogelijkheden en beperkingen, commerciële overwegingen enz. Het is niet eenvoudig om in zo'n samenwerkingsverband de eindgebruiker centraal te stellen. *Onthoud echter:* de eindgebruiker is de hoofdrolspeler van het UX-ontwerp.

DEEL 6 COMPUTERS WORDEN SLIM

- Wat zijn algoritmes? Waarom zijn ze zo belangrijk in de wereld van vandaag?
- Welke rol spelen computers in de automatisering?
- Hoe werkt kunstmatige intelligentie?
- …

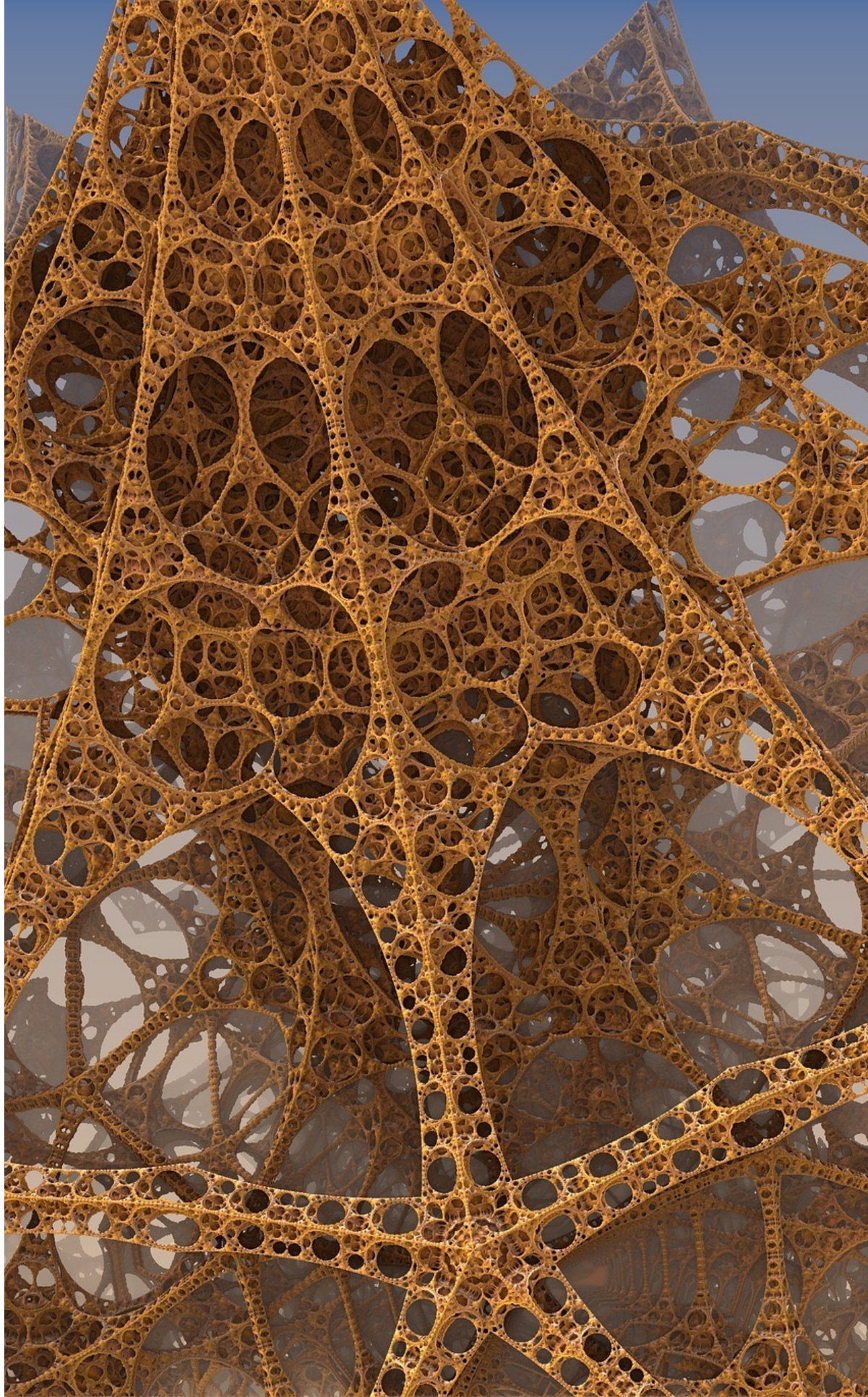


ALGORITMES

Ongetwijfeld heb je het woord al eens gehoord: een algoritme. Het heeft niks met ritmische muziek te maken alhoewel Shazam wel algoritmes gebruikt voor patroonherkenning in muziek. Google gebruikt het PageRank-zoekalgoritme en wanneer je naar een MP3-muziekbestand luistert dan is dit eerst met een compressiealgoritme tot een relatief klein bestand herleid vooraleer het op je mediaspeler is beland. Algoritmes spelen een cruciale rol in de verwerking van big data.

Algoritmes zijn in de digitale wereld alomtegenwoordig, maar wat zijn het eigenlijk?

Algoritmes zijn in dit geval stukjes programmeercode die ervoor zorgen dat een computer bepaalde taken in de correcte volgorde kan uitvoeren. We sommen de belangrijkste software-algoritmes op.



Dijkstra: de kortste weg

Het Dijkstra-algoritme is vooral bekend als het kortste pad-algoritme. Het beschrijft de kortste afstand tussen twee punten.

"Het algoritme van Dijkstra wordt gebruikt door verschillende internetprotocollen, voor het vinden van de kortste route tussen computers of routers. Het algoritme van Dijkstra geeft gegarandeerd het kortste pad, als er überhaupt een pad tussen A en B bestaat, maar kan hier wel lang over doen. De snelheid van het algoritme is afhankelijk van de manier waarop de punten en lijnen opgeslagen zijn in het geheugen van de computer."

(BRON; GUNNINK, M., "Padvinder vindt pad: over Pathfinding", (http://kninnug.nl/padvinder/index.html), Geraadpleegd op 20 oktober 2015.)

Dobbelen

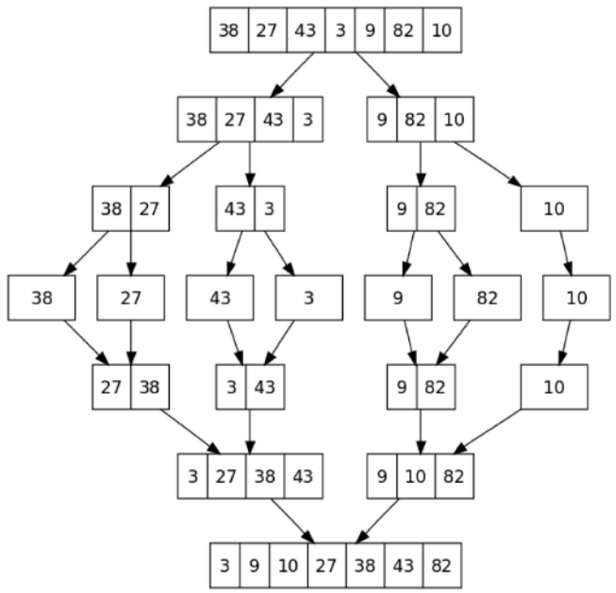
Als je dobbelt met een dobbelsteen of de lotto speelt, speel je letterlijk met "toeval". Een computer kan eveneens een lukraak getal genereren.

In de meeste programmeertalen kan je de functie om een random getal te genereren eenvoudig oproepen. Random-getalgenerators zijn belangrijk in beveiliging en cryptografie, games, AI enz.

Sorteeralgoritmes

Op school leer je woordjes alfabetisch rangschikken. Dat doe je in eerste instantie door naar de eerste letter te kijken. Bevatten veel woorden dezelfde beginletters, dan rangschik je die woorden door naar de tweede letter en vervolgens naar de volgende letters te kijken.

Rangschikking vereist dus een algoritme, een strikt stappenplan om tot het beoogde eindresultaat (een alfabetische volgorde) te komen. Een **sorteeralgoritme** zet elementen van een lijst in een bepaalde volgorde: alfabetische, aflopend, oplopend… Als je in bijvoorbeeld MS Excel getallen naar grootte rangschikt, dan gebruik je in feite een sorteeralgoritme.



Het alfabetisch rangschikken van woorden en het ordenen volgens grootte van getallen lijkt voor ons erg makkelijk. Maar het vraagt wel een stappenplan in ons hoofd, ook al doe je het na een tijdje zonder na te denken. Hoe beter het algoritme, hoe sneller de procedure. Dit soort algoritmes wordt reeds van in de beginjaren van de computer gebruikt in de computerwetenschappen. John von Neumann bedacht het "sort merge"-algoritme in 1945. Het sorteert gegevens door het splitsen, ordenen en weer samenvoegen (merge)van data. Tony Hoare bedacht in 1959 het Quicksort-algoritme, dat zoals de naam al doet vermoeden, sneller werkt dan "sort merge".

In 1964 bedacht J.W.J. Williams het "heapsort"-sorteeralgoritme. Zonder dit soort "sorteeralgoritmes" zouden heel wat moderne computertechnieken zoals data mining, AI, linkanalyse... ondenkbaar zijn.

Linkanalyse

Welke links bovenaan verschijnen in Googles zoekresultaten wordt bepaald door het PageRank-algoritme. Facebook toont bij iedere gebruiker een andere newsfeed gebaseerd op de "vrienden" met wie je het meest contact hebt (op Facebook dan toch), berichten die de meeste "likes" hebben gekregen enz. Wat je te zien krijgt, hangt dus af van een heleboel factoren en wordt bepaald door een linkanalyse-algoritme. Uiteraard verschillen de algoritmen maar de basis voor linkanalyse werd gelegd in 1976 door Gabriel Pinski en Francis Narin.

"Who uses this algorithm? Google in its Page Rank, Facebook when it shows you your news feed (this is the reason why Facebook news feed is not an algorithm but the result of one), Google+ and Facebook friend suggestion, LinkedIn suggestions for jobs and contacts, Netflix and Hulu for movies, YouTube for videos, etc. Each one has a different objective and different parameters, but the math behind each remains the same. Finally, I'd like to say that even thought it seems like Google was the first company to work with this type of algorithms, in 1996 (two years before Google) a little search engine called "RankDex" , founded by Robin Li, was already using this idea for page ranking. Finally Massimo Marchiori, the founder of "HyperSearch", used an algorithm of page rank based on the relations between single pages. (The two founders are mentioned in the patents of Google)."

Bron: OTERO, M., "The real 10 algorithms that dominate our world" (https://medium.com/@_marcos_otero/the-real-10-algorithms-that-dominateour-world-e95fa9f16c04), 2014, Geraadpleegd op 19 oktober 2015.

Suggesties

Wanneer je op twee verschillende computers van twee verschillende "mensen" dezelfde zoekterm invoert in Google krijg je vaak andere zoekresultaten. Googles **PageRank** baseert de zoekresultaten immers ook op uw zoekgeschiedenis en toont op basis daarvan ook andere advertenties via het **Google Adwords-algoritme**.

Wanneer je een zoekterm begint in te voeren toont **Google Suggest** reeds een reeks mogelijkheden waaruit je een optie kan selecteren. Ook iTunes, Amazon en Netflix tonen op basis van uw zoekgeschiedenis en aankopen andere producten die u wellicht interesseren. Eli Pariser noemt dit "informatie determinisme". De gebruiker zit in een soort van "filter bubbel" waar hij zelf voor een deel de controle over verliest. Je kan dus niet zelf meer kiezen wat je te zien krijgt, algoritmes bepalen dit voor u. Vraag is of je, puur uit marketingoogpunt, op die manier geen potentiële kopers van bepaalde producten mist.

Online dating is eveneens een voorbeeld van een veelgebruikte dienst die op basis van suggestie-algoritmes functioneert. Het matching-algoritme van OKCupid (een online datingdienst) werd afgeleverd door de Harvardwiskundige Christian Rudder. Het maakt een "ruwe" match op basis van gemeenschappelijke interesses. Vraag is natuurlijk of iemand met gelijkaardige interesses sowieso de beste partner is. Daarom berekent het algoritme eveneens hoe belangrijk elke vraag is voor de andere partij.

Veiligheid

Steeds meer computergebruikers plakken hun webcam af met een stukje plakband, iedereen weet dat je privacy wel eens kan geschonden worden door al je internetavonturen. Toch rekenen we wel op veiligheid als we online bankieren of wanneer we een reis boeken via Booking.com en onze creditcardgegevens invullen in een App Store of in PayPal. Zonder dit gevoel van zekerheid en veiligheid zou je wel gek moeten zijn om je bankkaartgegevens ergens achter te laten.

Het **RSA-algoritme** (ooit geschreven door de firma RSA) en het **Secure hash-algoritme** zorgen ervoor dat onze data veilig en versleuteld worden uitgewisseld tussen onze thuiscomputer en de servers van banken en online winkels. Een ander belangrijk algoritme in de wereld van de cryptografie is het **Integer factorization-algoritme**.

Foutcorrectie en autotuning

Het klinkt in de eerste plaats al indrukwekkend, het "Proportional Integral Derivative Algorithm" maar wat het doet is al even tot de verbeelding sprekend. Een **PID-controller** berekent constant foutwaardes als het verschil tussen de gemeten signaalwaarde en het gewenste resultaat en werkt vervolgens de fout weg. PIDcontrollers worden in de industrie toegepast, maar ook in meer populaire toepassingen. Zangers zingen in een opnamestudio niet altijd netjes op de toon. Doorheen een zangpartij zitten ze vaak wel eens net boven of net onder de juiste toon. Dat wil nog niet zeggen dat ze ronduit vals zingen, maar ze zitten zelden loepzuiver op de toon.

Autotuning zorgt ervoor dat al die foutjes verdwijnen in de uiteindelijke gemixte opname. Het lied "Believe" van de zangeres Cher zou de eerste popsong zijn die van autotuning gebruik heeft gemaakt. Je merkt meteen het verschil als je naar opnames uit de jaren 1960 of 1970 luistert en die vergelijkt met opnames uit de digitale periode vanaf 1984. Beluister bijvoorbeeld eens het album "Communique" van Dire Straits en het loepzuivere digitale geluid van "Brothers in Arms" van diezelfde groep. Niet dat we Dire Straits en Mark Knopfler "verdenken" van autotuning, maar de digitaliseringsalgoritmes hebben een blijvende "stempel" gedrukt op alle opnames vanaf de komst van de CD.

Conversie en compressie

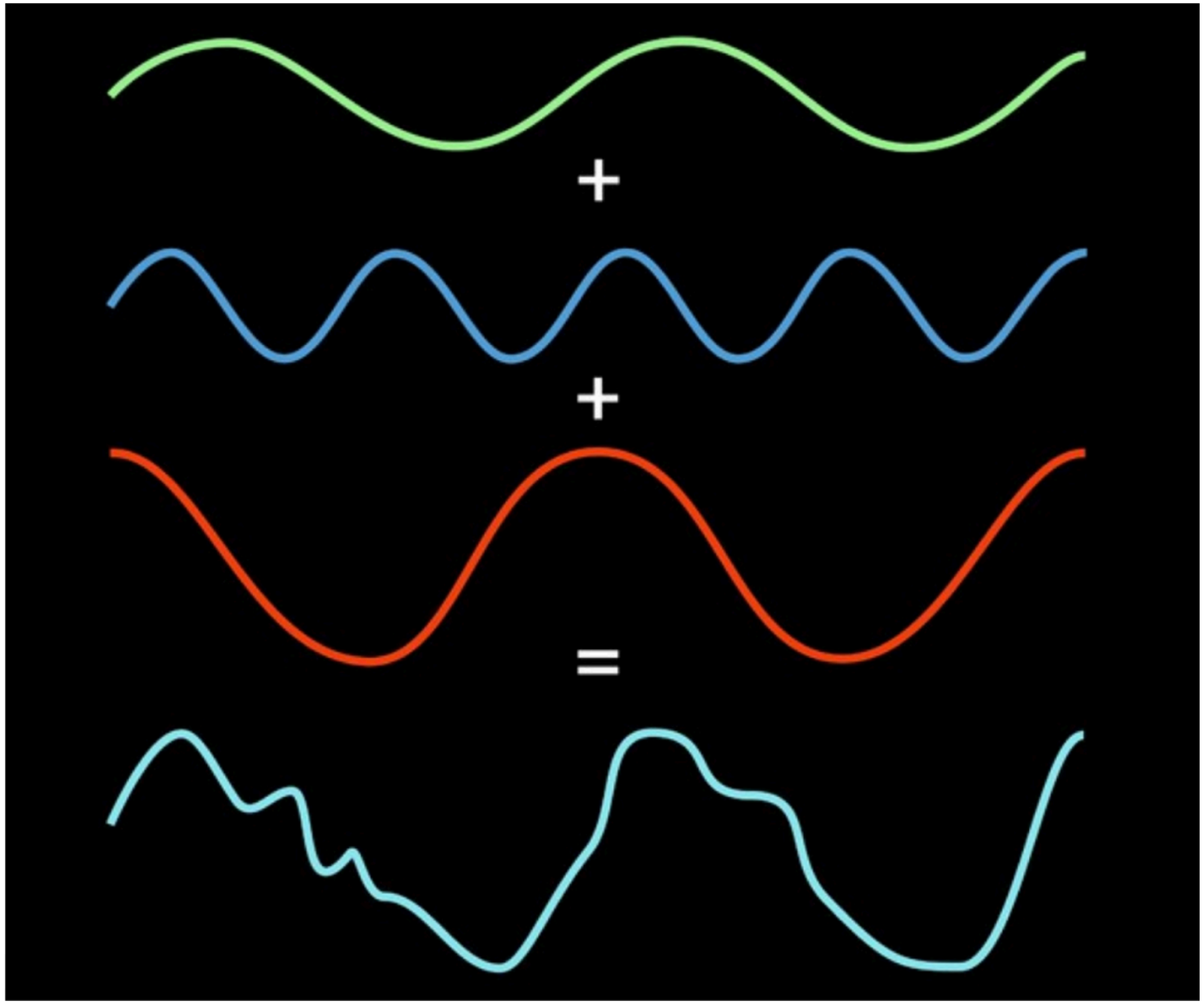
Het Fourier-transformatie-algoritme en de **Fast Fourier-transformatie of het FFT-algoritme** liggen aan de basis van heel wat compressie-algoritmen zoals JPG en MP3. Een **Fourier-tansformatie** zet een geluidssignaal of lichtintensiteit om in een golffunctie.

Het internet, WiFi, een smartphone, computer, router, communicatiesatellieten enz., zo wat alle toestellen waarin een soort computer aanwezig is, gebruikt deze algoritmen. Het meet analoge signalen doorheen de tijd en tekent die uit als een golf op basis van de gemeten frequenties. Door signalen op te zetten in (golf)functies, kan de intensiteit van de golf eveneens getransformeerd (vervormd) worden.

Het FFT-algoritme wordt bijvoorbeeld gebruikt bij afbeeldingsfilters waarbij men de intensiteit van kleuren of belichting 'transformeert'. Bij compressie worden lage intensiteitsverschillen weggegooid waardoor de bestandsgrootte erg verkleind.

"Fourier-transformaties worden gebruikt bij het omzetten van analoge signalen in digitale data en worden onder meer toegepast bij signaalverwerking, maar spelen ook een rol bij de compressie van beelden. Een complex signaal wordt met fft gereduceerd tot een aantal eenvoudig digitaal te representeren componenten. Daarvoor wordt een analoog sample opgedeeld in kleine stukjes, waarna de stukjes worden geanalyseerd en geconverteerd naar een digitaal signaal."

Bron: DE MOOR, W., "Sneller algoritme voor Fourier-transformaties ontwikkeld", (http://tweakers.net/nieuws/79444/sneller-algoritme-voor-fouriertransformaties-ontwikkeld.html), 2012, Geraadpleegd op 19 oktober 2015.



Compressie-algoritmes spelen een bijzonder belangrijke rol. *Ze zorgen ervoor dat data kleiner worden, maar vaak gaat dit gepaard met kwaliteitsverlies.* De algoritmes maken een afweging tussen kwaliteit en kwantiteit.

Controle en voorspelling

In 1948 schreef George Orwell het boek "1984" waarin hij scherpe kritiek leverde aan het adres van de USSR.

Hij schetste *"een onmenselijke dictatoriale eenpartijstaat die in alle opzichten volledig beheerst wordt door de Partij. De alom aanwezige leider van de Partij en het land wordt Big Brother genoemd. Iedere bewoner van het land wordt continu in de gaten gehouden via camera's die zelfs in de huizen zijn geïnstalleerd. Dit gebeurt onder de slogan 'Big Brother is watching you' ('Grote Broer houdt je in de gaten')."*

Ondertussen leven wij in het westen niet in dictatoriale eenpartijstaten, maar Big Brother lijkt heel reëel. Ieder van ons wordt in de gaten gehouden, zij het niet door mensen, maar door algoritmes. Vanuit de USA houdt het National Security Agency (NSA) en zijn internationale partners (Five Eyes: US, Australië, Canada, Nieuw-Zeeland, Groot-Brittannië) wereldwijd miljoenen mensen in de gaten. Ze monitoren telefoongesprekken, SMS-berichten, e-mails, webcambeelden, GPS-locaties enz. De hoeveelheid verzamelde informatie is veel te groot om te laten analyseren en interpreteren door mensen. De analyse gebeurt automatisch met behulp van krachtige algoritmes.

Sommige algoritmes zoals IBM's CRUSH, gaan nog een stukje verder. CRUSH of " Criminal Reduction Utilizing Statistical History" zorgt voor "predictive analysis" en is in hoofdzaak bedoeld om misdaten te voorkomen. De politiediensten van Memphis konden de misdaadcijfers met 30% laten teruglopen dankzij CRUSH. Het aantal gewelddadige misdrijven liep sinds 2006 terug met 15%. Op basis van statistische gegevens, data-aggregatie en algoritmes, toont de software criminele hot spots op een kaart.

Politie-eenheden kunnen op de manier pro-actief ingeschakeld worden en bij wijze van spreken aankomen voor een misdaad plaatsvindt. In de toekomst zullen criminelen heel snel opgespoord kunnen worden dankzij internetactiviteit, GPS en biosignaturen, verdacht gedrag.

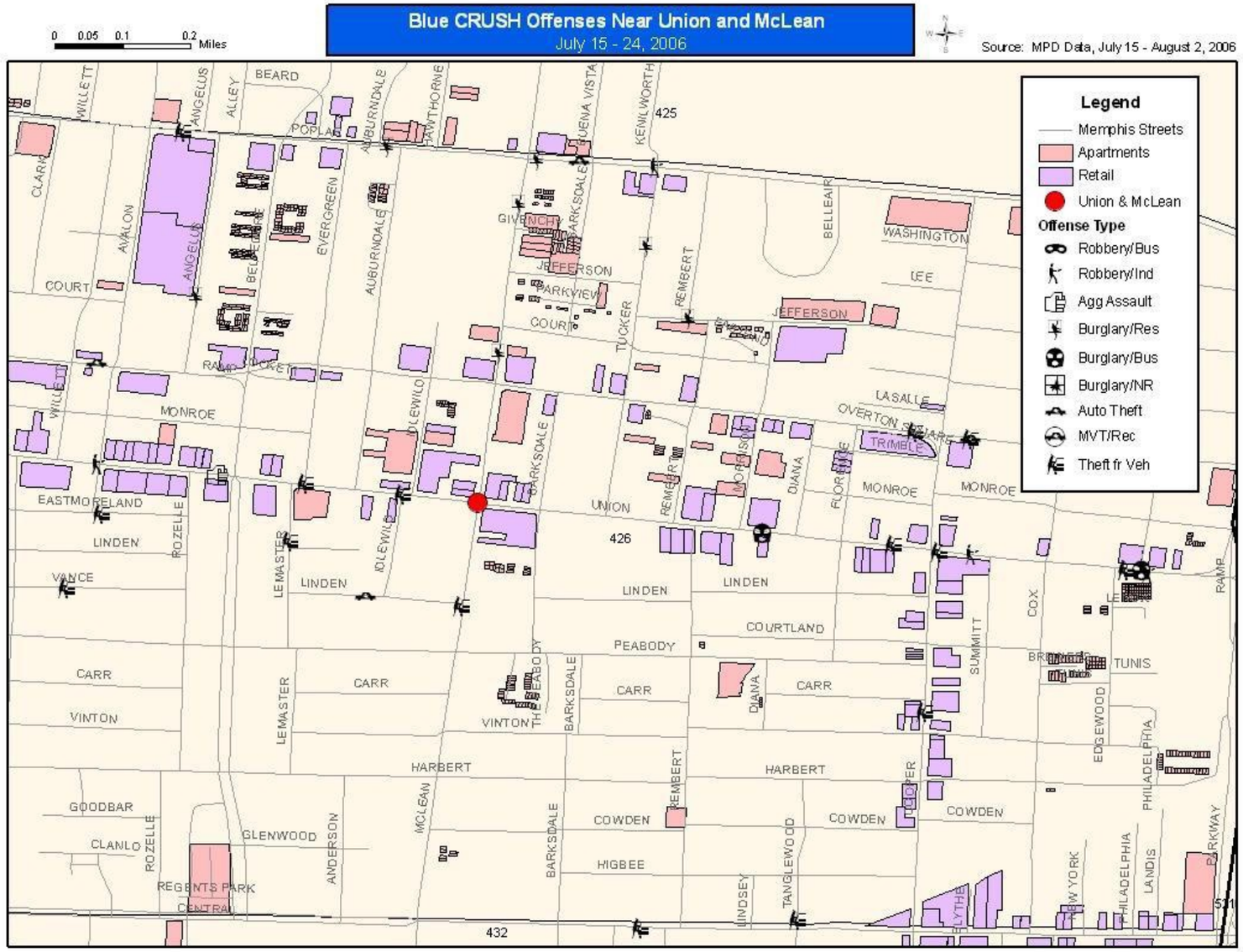
"In the future, these systems will largely take over the work of analysts. Criminals will be tracked by sophisticated algorithms that monitor internet activity, GPS, personal digital assistants, biosignatures, and all communications in real time. Unmanned aerial vehicles will increasingly be used to track potential offenders to predict intent through their body movements and other visual clues."

Bron: DVORSKY, G., "The 10 algorithms that dominate our world", (http://io9.com/the-10-algorithms-that-dominate-our-world-1580110464), 2014, Geraadpleegd op 20 oktober 2015.

De film Minority Report van Steven Spielberg uit 2002 toont al een glimp van waartoe dit in de toekomst kan leiden.

Algoritmes voor predictieve analyse zijn reeds lang in gebruik in de beurswereld. De analyse van razendsnel voorbijvliegende transacties gebeurt met behulp van slimme algoritmes. Soms kloppen de predicties niet, zoals bij de "Flash Crash" van 2010.

De opkomst van AI en slimme camera's, maakt het volgen en controleren van burgers nog makkelijker.



In China is Big Brother niet langer fictie: *"This fictional scenario is now a daily reality in the People's Republic of China, thanks to a massive police surveillance apparatus powered by Big Data and artificial intelligence. For example, trains now require national ID's to buy tickets, which allows the government to block human rights activists or anti-corruption journalists from traveling. In Xinjiang province, home of China's Uighur Moslem minority, the government uses AI-sifted Big Data to scrutinize anyone entering a mosque or even a shopping mall thanks to thousands of checkpoints requiring a national ID check-in—and which can collate real-time data with other personal information on everything from bank accounts to family planning."*

Bron: HERMAN, A. "China's Brave New World Of AI", (https://www.forbes.com/sites/arthurherman/2018/08/30/chinas-brave-new-world-of-ai/), 2018, Geraadpleegd op 5 april 2019

HET DOEL EN DE EVOLUTIE VAN COMPUTERS EN AUTOMATISERING

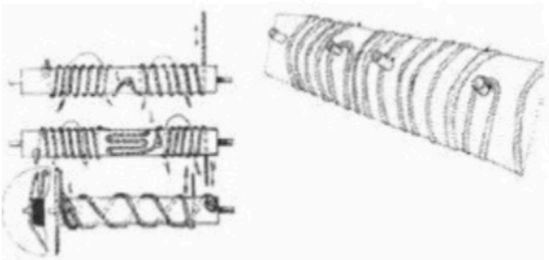
Heel wat mensen vrezen dat de opkomst van kunstmatige intelligentie zal leiden tot een gigantisch banenverlies. Wat moeten onze kinderen en kleinkinderen nog gaan doen om een inkomen te vergaren? Waarom maken mensen machines en technologie die hen overbodig maakt?

Welke evolutie kennen machines? In eerste instantie hebben mensen gereedschappen ontwikkeld om het werk lichter te maken, zodat ze minder spierkracht moesten gebruiken. Het is nu eenmaal eenvoudiger om een spijker met een hamer in een muur te slaan dan met je blote vuist.

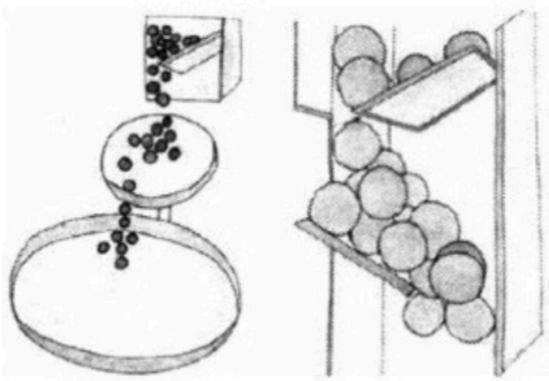
De mens heeft m.a.w. gereedschappen ontwikkeld die het werk, maar ook het denkwerk of onthoudwerk lichter te maken. De uitvinding van het schrift bijvoorbeeld zorgde er voor dat je niet langer alles moest onthouden. Door het gebruik van trekdieren, maar ook windmolens en watermolens, kon heel wat menselijke spierkracht uitgespaard worden.

Maar het aansturen van taken diende over het algemeen nog door mensen te gebeuren. Hoe je een bepaalde taak tot een goed einde moest brengen, hoe je gereedschap moest bedienen… dat diende nog aangeleerd te worden. Mensen moesten de diverse stappen van een taak, job of functie nog aanleren.

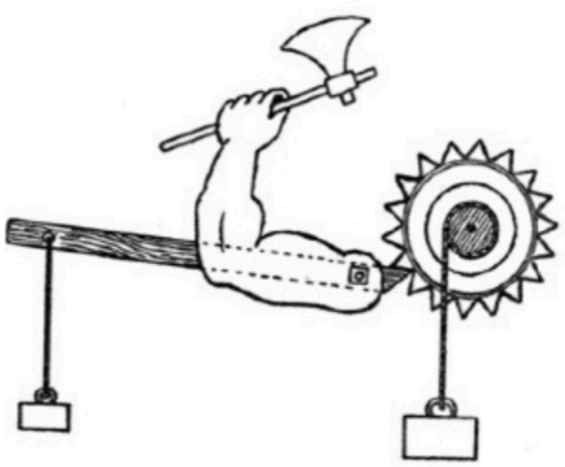
Elke taak bestaat immers uit een aantal te volgen stappen. Ze volgt een soort van recept. In de wiskunde en informatica noemt men zo'n proces of stappenplan een **algoritme**. De mens ontwikkelt al meer dan 2000 jaar automaten waarin zo'n algoritmes konden worden geprogrammeerd. Een klok is daar een mooi voorbeeld van. Een klok bevat een ingewikkeld mechanisme waarmee de tijd kan worden berekend. Tot de meest bekende voorbeelden van automatisering, behoort het assemblageproces van auto's aan een "lopende band" in autofabrieken.

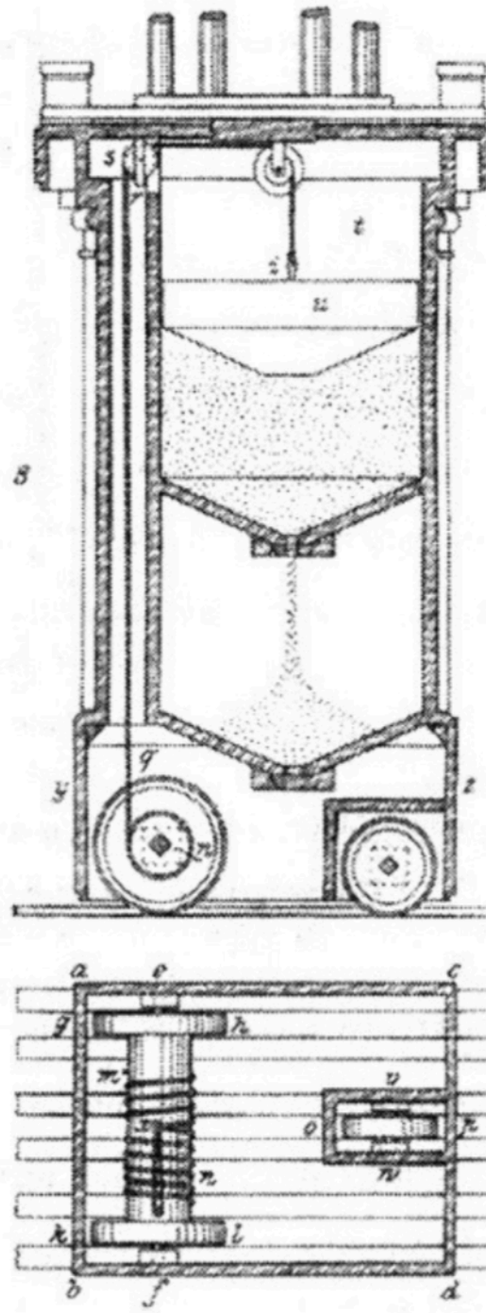


De touwen en de pinnen bepalen de beweging van de assen. (Bron: Manuela Rausch, *Die Programmgesteuerten Automaten Des Heron Von Alexandria*, 2012)



Geluidseffecten in het automatische theater. (Bron: Manuela Rausch, *Die Programmgesteuerten Automaten Des Heron Von Alexandria*, 2012)





(Bron: Beck T., *Herons des älteren Automatentheater*, Jahrbuch des Vereines deutscher Ingenieure, 1909)

De eerste golven van automatisering vanaf midden 19e eeuw kostten het werk aan veel arbeiders. Zeker vanaf het moment dat ook robots in het productieproces werden ingezet. Een robot is zodanig afgesteld en geprogrammeerd dat hij een bepaalde taak telkens stapsgewijs kan uitvoeren, zonder moe of ziek te worden of zonder vakantie te eisen. Robots voeren m.a.w. voorgeprogrammeerde algoritmes uit.

De volgende stap in de evolutie van technologie gaat nog een stuk verder. We bouwen machines die zichzelf kunnen programmeren. Meer nog, slimme algoritmes zijn in staat om zelf taken te leren. Shazam kan bijvoorbeeld liedjes herkennen en wordt daar steeds beter in. Google Translate leert net als een kind stapsgewijs alle talen ter wereld.

Doemsnecario's duiken op waarbij toekomstige software in staat zal zijn elke denkbare taak aan te leren om in sneltempo slimmer te worden dan de mens.

Programmeerbare automaat, 1e eeuw na Chr. Ingenieur: Heron van Alexandrië

WORDEN COMPUTERS SLIM(MER DAN DE MENS)?

Kunstmatige intelligentie. Het klinkt velen angstaanjagend in de oren. Zullen computers en robots ons leven gaan domineren? Is er nog wel een toekomst weggelegd voor ons? Heeft het nog wel zin om te studeren als computers en robots ons werk overnemen? Zal AI straks slimmer zijn dan de mens of is dit nu al het geval?

In dit hoofdstuk komt je te weten hoe ver het staat met de ontwikkeling van kunstmatige intelligentie. Hoe werkt AI nu eigenlijk? Kan je kunstmatige intelligentie nu eigenlijk wel vergelijken met menselijke intelligentie? Werken die beide vormen van intelligentie op dezelfde manier?

Zijn er terreinen waarop AI de mens nooit zal voorbijsteken?

Hoe leren mensen en computers?

Computers volgen stappenplannen (algoritmes) om invoer tot de gewenste uitvoer te verwerken. Hoe slim die stappenplannen soms ook wel niet zijn, toch kunnen ze niet tippen aan menselijke intelligentie. Bovendien zijn al die algoritmes eerst door mensen bedacht. Computers slagen er namelijk niet in om zelf stappenplannen te bedenken of om de best mogelijke oplossingen te bedenken voor problemen.

Mensen kunnen dit wel. Ook al maak je in je leven behoorlijk wat foute keuzes, toch leer je in veel gevallen uit je fouten. We leren uiteraard niet alles door eerst fouten te maken. De meeste kennis en vaardigheden leren we door ervaring of door naar voorbeelden te kijken. Herhaling helpt daarbij, dat weten we allemaal.

Kunstmatig leren

Kunstmatige intelligentie betekent dat computersystemen zelf leren tot oplossingen te komen. We programmeren software om zelf te leren door ervaring of aan de hand van voorbeelden, net zoals mensen en zelfs dieren dit doen.

Computerwetenschappers proberen hiervoor de werking van neuronen in de menselijke hersenen te simuleren met software. We spreken in dit geval van een **neuraal netwerk**.

Een paar voorbeelden:

Als je twijfelt of je de fiets of de auto zou nemen om naar je werk te gaan, dan weeg je heel wat factoren af: het weer, de afstand, vermoeidheid, kostprijs, of je nog naar de winkel moet… De ene dag neem je de auto, de andere dag weer niet. Je uiteindelijke beslissing (output) hangt van heel veel inputs af. **De neuronen in je hersenen maken heel wat (statistische) afwegingen.** Misschien neemt je collega wel de fiets, terwijl jij de auto neemt. Hij/zij heeft misschien een andere afweging gemaakt. Als je echter de ene dag strontnat werd door een fikse regenbui, leer je in de toekomst eerst het weerbericht te bekijken voor je op je fiets stapt.

Stel dat je een kat als huisdier hebt en je zegt telkens tegen je zoontje van twee: “Dit is de kat”. Je zoontje heeft echter nog nooit een hond gezien. De kans bestaat dat hij bij zijn eerste ontmoeting met een hond zegt: “Kijk mama, kat!” Jij zal dan zeggen: “Nee hoor, dat is een hond”.

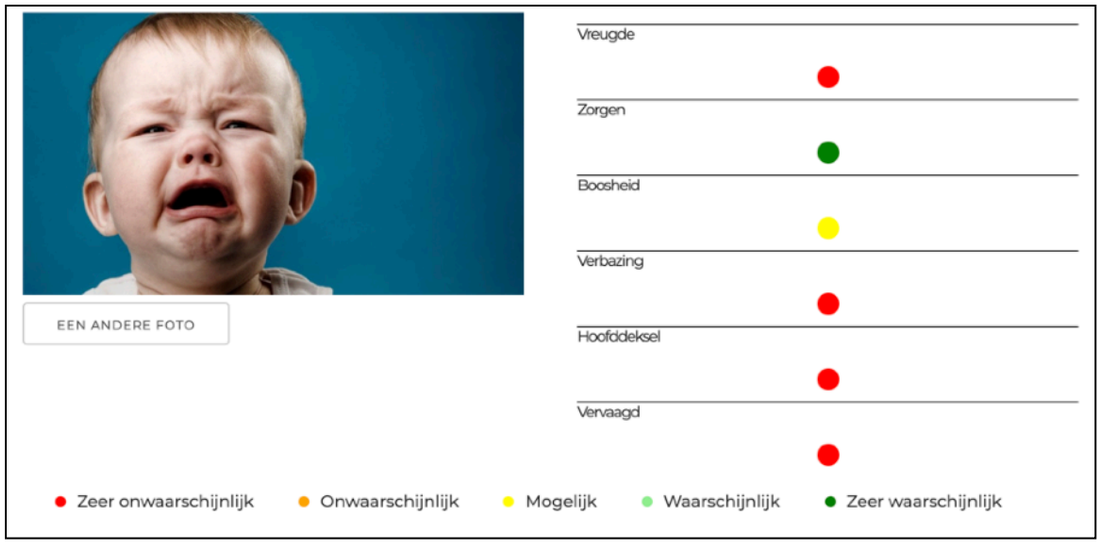
Door dit vaak te herhalen, zal je zoontje snel het verschil leren tussen een hond en een kat. Hij zal geleidelijk beide leren onderscheiden en even zeer begrijpen dat de schapen van de buren al evenmin honden of katten zijn. Kinderen (ook op school) leren door "**supervised learning**". *Ze leren uit hun fouten en uit voorbeelden omdat iemand hen daar op wijst.*

Door neurale netwerken in software te programmeren, kunnen computers zelf leren (**machine learning**). We voeren de AI-software eerst met **trainingssets**. Dat kunnen bijvoorbeeld een reeks foto's van honden zijn. Door een massa foto's van honden als trainingssets aan de software te voeren, kan het neuraal netwerk stilaan zelf honden op andere foto's leren herkennen.

Een spamfilter is een bekend voorbeeld van AI-software op basis van een neuraal netwerk. Heel wat mails die het woord "casino" en "penisverlenging" bevatten, zullen spamberichten zijn. Maar stel dat je vriend je een mail stuurt met de melding "Ga je vanavond mee naar het casino?" dan wil je niet dat dit bericht in de spam terechtkomt. Of als je echt problemen hebt en je huisdokter stuurt je een mailtje over je voortplantingsorgaan met daarin het zinnetje "we gaan je ziekteverlof verlengen", dan wil je niet dat de combinatie van "verlengen" en "penis" in hetzelfde bericht tot "spam" leidt. Het neuraal netwerk moet heel wat afwegingen maken om te besluiten of een bericht al dan niet als spam gemarkeerd moet worden. Zo'n spamfilter bouw je door een reeks van verdachte woorden samen te stellen. Berichten waarin een of meer van die woorden voorkomen, moeten al dan niet als spam worden herkend. Die verdachte woorden krijgen in het ene bericht de classificatie "true negative" (geen spam), in het andere "true positive" (wel degelijk spam).

Andere termen kunnen leiden tot de classificatie "false positive" of "false negative". Dit gebeurt wanneer een spambericht onverwacht toch in je gewone mailbox terechtkomt of een goed bericht in de spam. In dat geval kan je een bericht *zelf* classificeren. *Op dat moment leert de spamfilter dat hij een fout heeft gemaakt.* Daar zal hij in de toekomst rekening mee houden. Spamfilters worden op die manier steeds beter en maken minder en minder fouten.

Daarom kan je zeggen dat AI-software letterlijk **voorspellingen** doet op basis van wat de software eerder is geleerd. Die voorspellingen kunnen ok zijn (true positive, true negative), maar even zeer fout (false positive of true negative). In dit laatste geval moet het algoritme bijgestuurd worden.



Een AI-algoritme op basis van de Google Vision API herkent de emoties. Het doet dit met "procentuele" waardes, in dit geval uitgedrukt met kleurwaardes. (Bron: www.schoolvoorbeeld.be/nodig/cloudvision - Kris Merckx)

Een neuraal netwerk berekent statistische kansen.
(Bron: www.schoolvoorbeeld.be/nodig/cloudvision - Kris Merckx)

AI is beperkt

Toch moet elk stukje AI-software, elk neuraal netwerk nog door mensen worden geprogrammeerd. We moeten ze nog voeden met trainingssets: **supervised learning**. Een spamfilter is sterk in het herkennen van spam, maar kan geen zelfrijzende auto besturen. De software van zo'n auto kan op tijd stoppen voor een overstekend ree, maar kan je niet gebruiken voor een vertaalopdracht. **Elk stukje AI-software heeft namelijk nog zijn eigen specialiteit.**

Daarom kan je gerust stellen dat de meeste beperkt is, en bestempelt men de meeste AI-algoritmes als "**Narrow AI**".

De veroveringstocht van AI

Toch is AI aan een niet te stuiten opmars begonnen. Amazon, Netflix en Spotify geven hun gebruikers suggesties (**recommendations**) op basis van het gedrag van andere bezoekers. Zelfrijdende auto's voorspellen het rijgedrag van de auto's in hun buurt om ongelukken te voorkomen. Weervoorspellingen op basis van AI zijn accurater dan ooit tevoren. Digitale assistenten zoals Google Home en Amazon Alexa begrijpen gewone vragen en proberen daarop zinvol te antwoorden. Google Translate vertaalt elke dag een beetje beter.

Algemene kunstmatige intelligentie

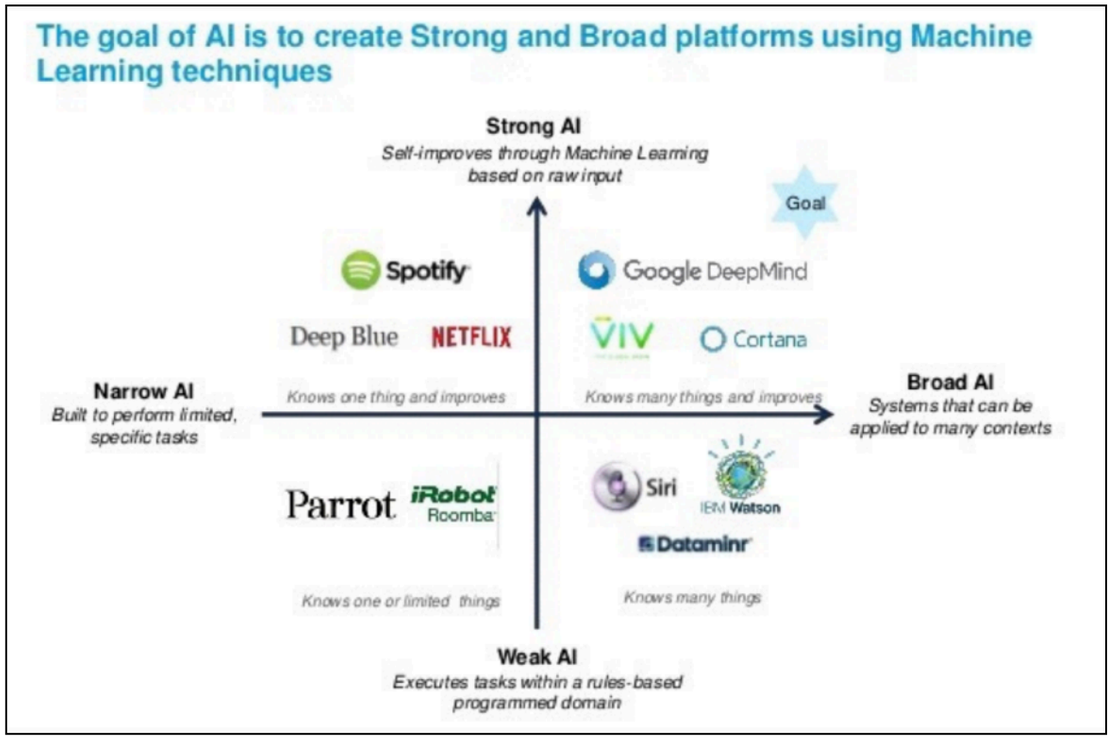
Wie bang is dat AI de mens is voorbijgestoken, kunnen we (voorlopig) gerust stellen. Ondanks het buitensporig enthousiasme en de talloze persberichten, zijn we nog een eind verwijderd van een AI-algoritme dat zonder toezicht **(unsupervised learning)** elk probleem kan oplossen (**broad AI of Algemene Kunstmatige Intelligentie**).

Toch vordert de ontwikkeling van unsupervised learning en **deep learning**: algoritmes die zelf problemen leren oplossen, zonder hiervoor eerst trainingssets te doorspitten. Een van de rijzende sterren achter deep learning is **Demis Hassabis** en zijn bedrijf **Deepmind** dat werd ingelijfd door Google. Deepmind scheerde hoge toppen toen het AI-programma **AlphaGo** de grootmeester in het Chinese Go-spel versloeg.

Om je een idee te geven: bij Go kan een speler meer zetten doen dan er atomen in het heelal zijn. De overwinnende zet die AlphaGo deed, verbaasde Go-spelers wereldwijd. Geen mens zou ooit op zo'n creatief bedachte zet zijn gekomen.

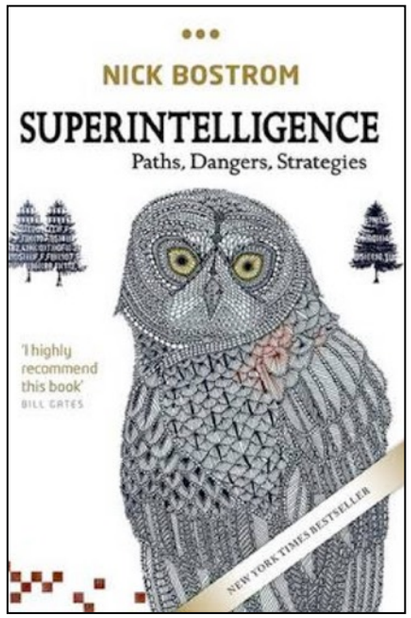
Wordt de mens overbodig?

Automatisering maakte in het verleden handenarbeid overbodig. AI kan taken waar voordien denkwerk aan te pas kwam, automatiseren. Sowieso zal AI-automatisering heel wat jobs en opleidingen overbodig maken. Over welke jobs het eerst zullen verdwijnen, hoor je tegenstrijdige berichten. Over het algemeen gelooft men dat jobs die meer expertise of creativiteit vragen, langer zullen overleven, evenals gespecialiseerde handenarbeid.



(Bron: AMP new ventures)

Het boek "Superintelligence" van de Britse computerwetenschapper Nick Bostrom



Februari 2018. Volgens The Economic Times van India vertelde Dr. David Hanson van Hanson Robotics, op het World Congress on Information Technology in Hyderabad, dat robots "would be alive and have full consciousness in five years".

De singulariteit

Wanneer een computer menselijke intelligentie bereikt of het niveau waartoe mensen na miljoenen jaren biologische evolutie zijn aanbeland, dan stopt het voor AI niet. Nick Bostrom, Ray Kurzweil, Bill Gates, Elon Musk... waarschuwen voor dit schakelmoment: de **singulariteit**.

Kunstmatige intelligentie zal zich, volgens sommigen, vanaf dat moment in razend tempo blijven verder ontwikkelen. Menselijke intelligentie is volgens hen geen eindpunt, maar een kantelmoment waarna menselijke intelligentie snel overbodig lijkt te worden, als we hen mogen geloven.

Voor Ray Kurzweill is die "singulariteit" het moment waarop mens en machine in elkaar zullen opgaan en er hybride overgangsvormen zoals humanoïden of cyborgs zullen ontstaan.

Bostrom, Gates en Musk roepen overheden op om goed na te denken en nu al "regels" en "beperkingen" in te stellen. Eén van de eerste resultaten van die oproep is de "internationale" bereidheid om een halt toe te roepen aan de ontwikkeling van autonome robots die op het "slagveld" en bij oorlogvoering zullen ingeschakeld worden.

DEEL 7 VERBONDEN WERELD

- Hoe werkt het world wide web?
- Hoe kan je foto's en teksten door een draadje sturen?
- ...

WERELDWIJDE NETWERKEN

Leerstof tweede en derde jaar (nog niet beschikbaar)

Netwerktechnologie, websites… enz.

WERELDWIJD WEB & HTML

Websites en html

Leerstof tweede en derde jaar (nog niet beschikbaar)

DEEL 8 HOW TO BECOME A SMARTIST?

- Hoe gebruik je grafische software?
- Hoe maak je animatiefilm?
- Hoe werkt 3D?
- Werken met film en geluid.
- …

GRAFISCHE VORMGEVING

Grafische vormgeving heeft op zich niets met computers te maken. Maar de opkomst van computers met WYSIWYG-interfaces en printers, heeft de deur geopend voor digitale grafische vormgeving. Nadeel is dat heel wat mensen zonder grafische voorkennis zich met vormgeving zijn gaan bezighouden. Iedereen waant zichzelf fotograaf en grafisch vormgever.

Enige voorkennis en vooral basiskennis is belangrijk om echt een verschil te kunnen maken. Niet zo zeer de software die je gebruikt, maar wat je maakt, de ideeën die je hebt en hoe je ze uitwerkt, zijn van belang.

Een grafisch vormgever hanteert bepaalde stijlen naargelang de opdracht. Een stijl is vaak gebaseerd op een bepaalde kunststroming, gebruikt een afgemeten kleurenpalet en typografie.

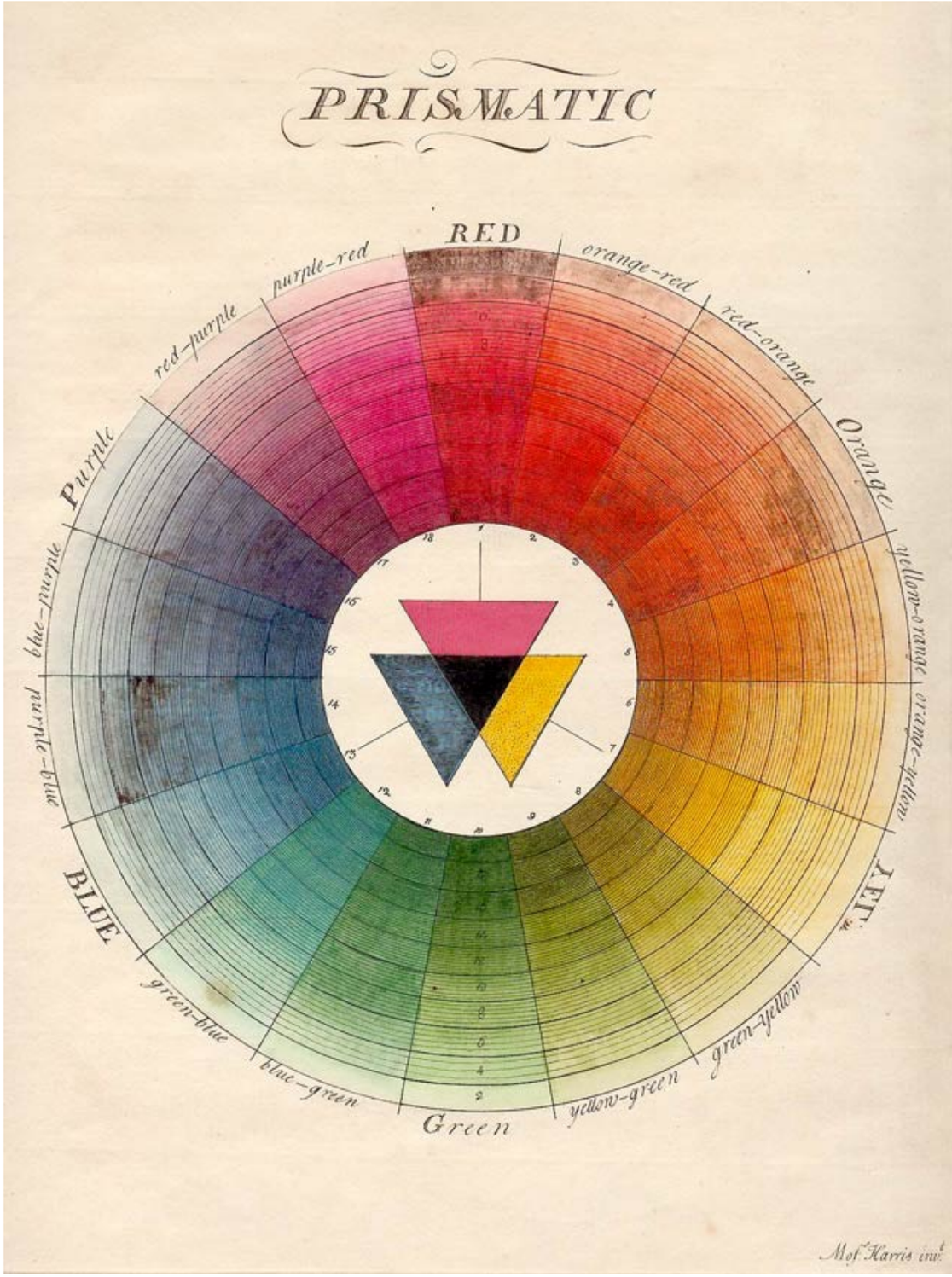
Platenhoezen zijn een mooi voorbeeld van graphic design waarbij de stijl meteen wat meer vertelt over het muziekgenre. Het kleurenpalet, de typografie en de illustraties of foto's bepalen de stijl.

Huisstijl voor restaurant en bar, duidelijk gebaseerd op de Art Deco-kunststroming (aanvang 20e eeuw)

Huisstijl en grafische stijlen

Net zoals jijzelf herkenbaar bent aan je uiterlijk, maar ook je kledingstijl en je haartooi, beschikken veel bedrijven en organisaties over een huisstijl. Het bezorgt hun bedrijf en eventuele producten en merken een identiteit (**corporate identity**). Hiervoor stelt het bedrijf in samenspraak met een grafisch bureau of reclamebureau een **stijlgids** of stijlbijbel op. Hierin bepalen ze welke kleuren, lettertypes, illustraties, logo's enz. gebruikt moeten worden in alle bedrijfscommunicatie. Bij een merknaam zorgt het voor een betere "herkenning" **(brand recognition**).

Originaliteit is een gevaarlijk begrip. Over het algemeen laat je je altijd inspireren door bepaalde stijlen. Daar is ook niks mis mee. Belangrijk is dat de diverse elementen van jouw stijl bij elkaar passen. Daarom kan je best naar voorbeelden uit bepaalde **grafische stijlen** of **kunststromingen** kijken. *De Art Deco-stijl leent zich bijvoorbeeld goed voor zaken of producten die een kwaliteitsvolle uitstraling hebben: parfum, chique restaurants enz.* Als grafisch vormgever is kennis van de kunststromingen van de negentiende en twintigste eeuw, en vooral van de grafische stijlen daar binnen, niet enkel belangrijk, maar zelfs essentieel. Een graficus of graphic designer gebruikt een stijl voor het maken van affiches,



Het kleurenwiel van Moses Harris in zijn boek The Natural System of Colours (1776). Tegenoverliggende kleuren noemt men complementaire kleuren. Ze 'passen' goed bij elkaar.

posters, de opmaak van boeken en magazines, visitekaartjes… Hij/zij combineert tekst met afbeeldingen en grafische elementen.

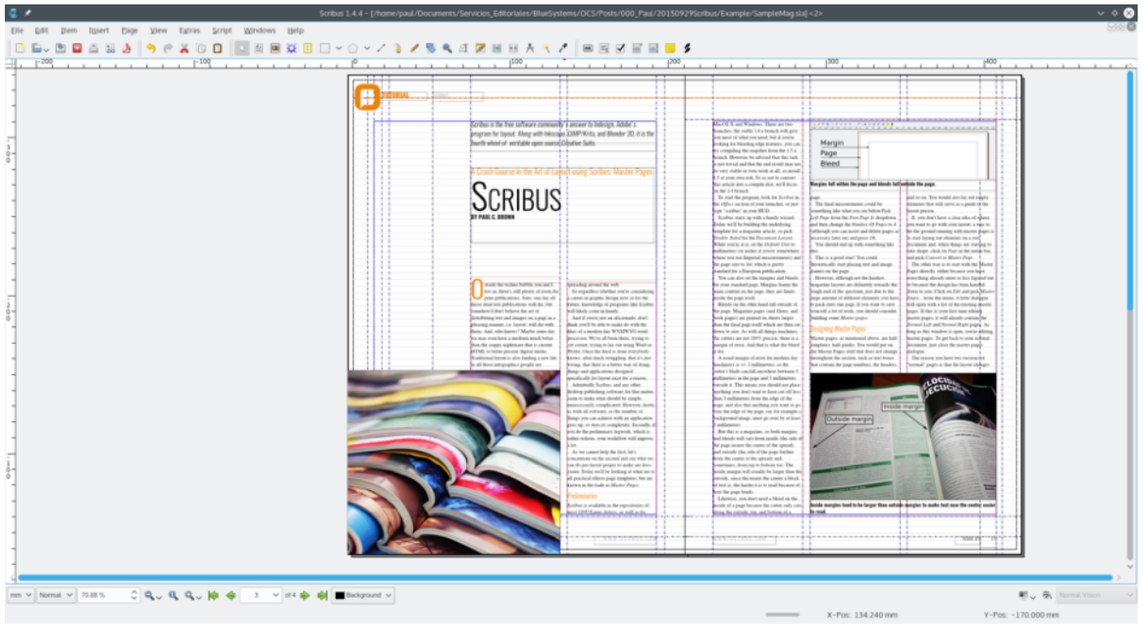
Kleuren en kleurenpalet

Over het verschil tussen RGB- en CMYK-kleuren hebben we het al uitgebreid gehad. Bij een huisstijl hoort een **kleurenpalet:** een reeks kleuren (of beter gezegd "kleurcodes") die de graficus gebruikt bij de opmaak van bedrijfscommunicatie. Als je ontwerpt voor drukwerk, gebruik je zoals eerder aangehaald CMYK-kleuren, bij weergave op een scherm RGB-kleuren.

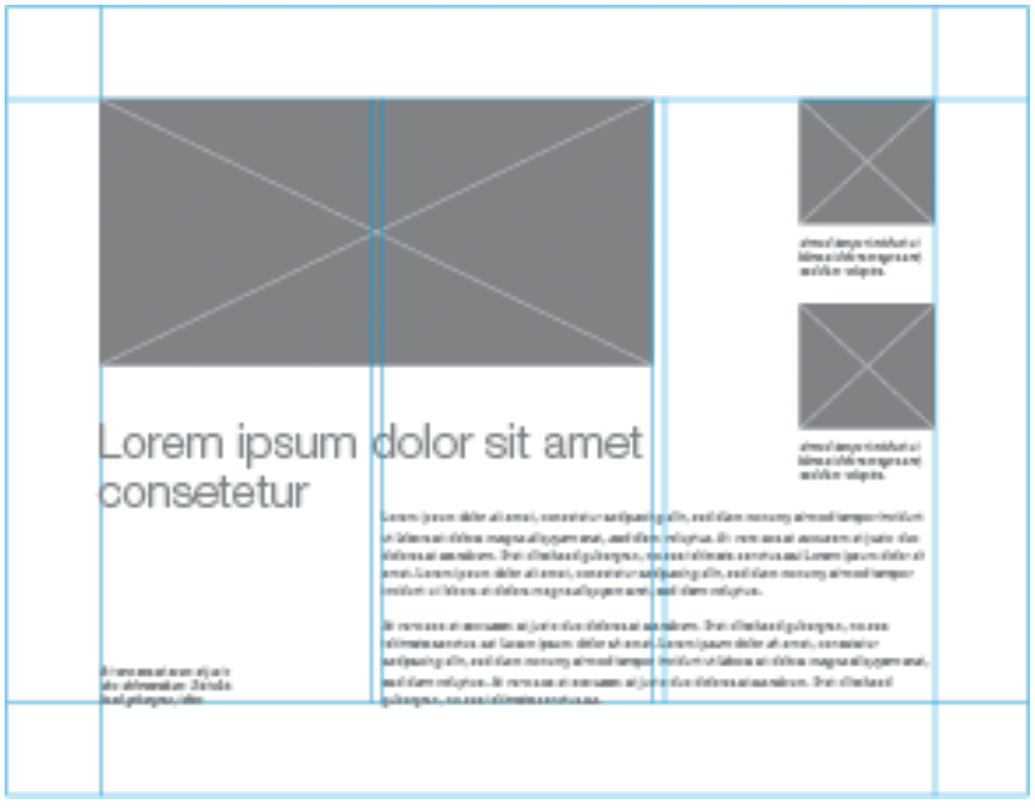
Een kleurenwiel is een handige techniek om een passend kleurenpalet samen te stellen. Uiteraard bestaan hier ook softwaretools voor.

Over kleuren en smaken valt niet te discussiëren hoor je vaak. Maar dit klopt niet helemaal. Je kan dus perfect, met behulp van een kleurenwiel, een passend palet samenstellen. Tegenoverliggende kleuren noemt men **complementaire kleuren**. Ze passen goed bij elkaar.

Houd er eveneens rekening mee dat kleuren vaak een **symbolische connotatie** hebben.



Een grid met hulplijnen in het open source DTP-programma Scribus.



Zowel bij DTP als bij webdesign maken grafici gebruik van "grids", rasters van hulplijnen waar tegen alle pagina-onderdelen worden uitgelijnd.

Uitlijnen op een grid

Om de positionering van elementen zoals afbeeldingen, titels en teksten te vergemakkelijken, gebruiken grafici een grid. Dit is een in het eindresultaat onzichtbaar raster (een soort "basisstramien") van hulplijnen waarop je alle grafische elementen (inhouden, grafieken, illustraties...) kan positioneren.

Typografie

Ook de keuze van een geschikt lettertype is belangrijk. Lettertypes kan je op basis van hun vorm indelen in een aantal groepen. **Serif-lettertypes (schreef-lettertypes)** hebben dunne streepjes aan de uiteinden van de letters. **Sans-serif-lettertypes (of schreefloze lettertypes)** hebben dit niet.

Serif-lettertypes hebben een iets klassiekere uitstraling en zouden moeilijker leesbaar zijn voor ondermeer dyslectici. Toch zijn heel wat romans en boeken nog opgemaakt met serif-lettertypes.

Bij **monospace**-lettertypes neemt elke letter even veel ruimte in in de breedte. Een i zal dus even veel breedte in beslag nemen als een l.

Serif

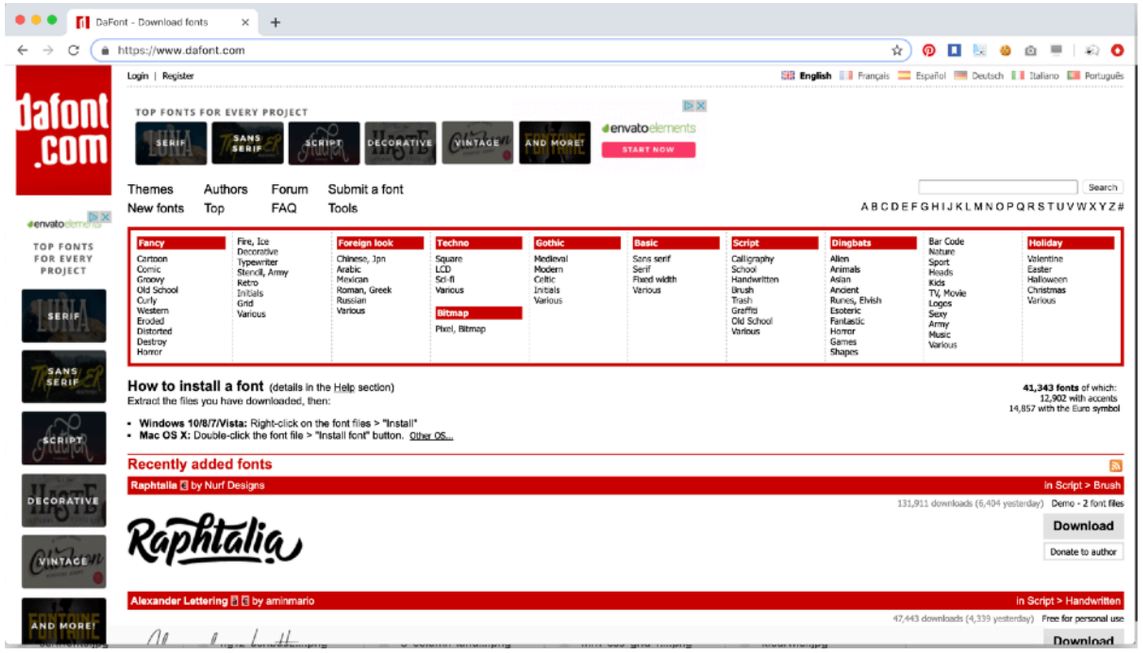
A serif is a small decorative flourish on the end of the strokes that make up letters and symbols.

Sans Serif

Sans Serif fonts do not have any flourishes at the end of strokes.

Monospaced

Monospaced fonts, letters, and characters each occupy the same amount of horizontal space.



dafont.com, een site met honderden lettertypes. Je kan de TTF- en/of OTF-bestanden downloaden en installeren.

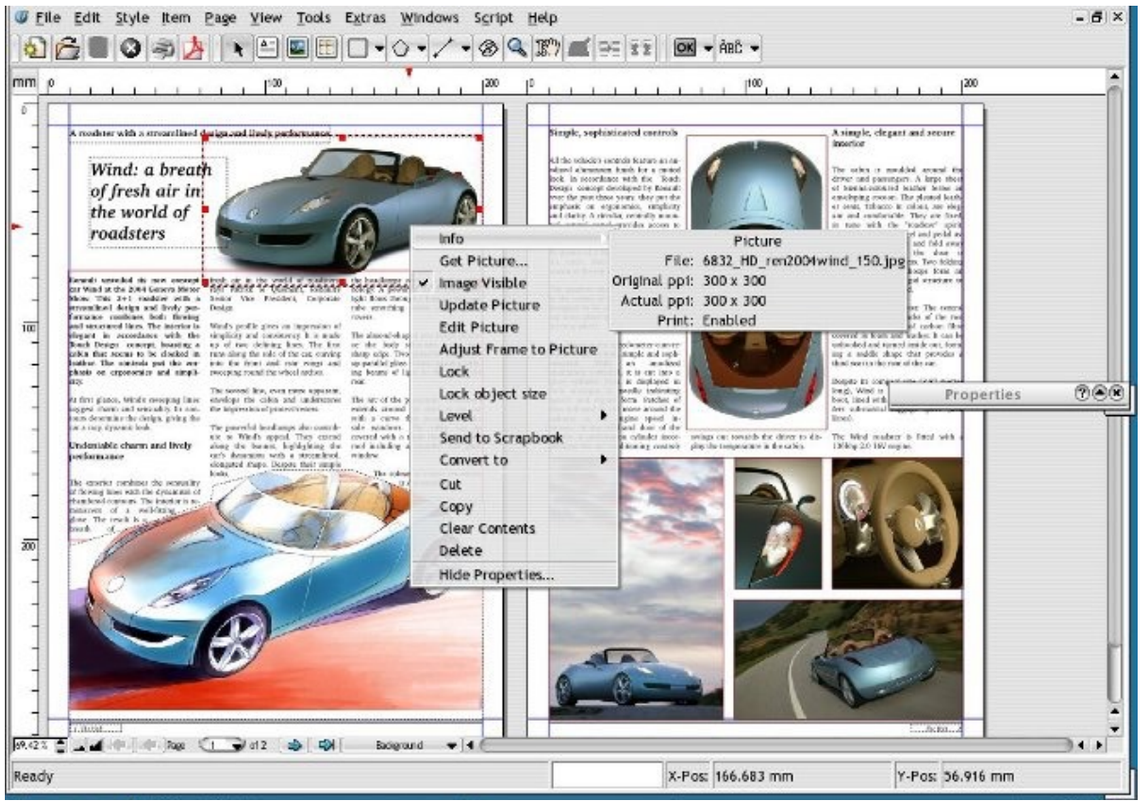
Dit wil niet zeggen dat je een vette "i" krijgt, maar links en rechts wordt dan meer witruimte voorzien.

Er bestaan honderden, zo niet duizenden lettertypes. Naast de genoemde soorten bestaan er ondermeer **handgeschreven (script) lettertypes, fantasy-lettertypes** en lettertypes waarbij elke letter vervangen wordt door een symbool, de zogenaamde **dingbat-lettertypes** (webdings, wingdings…)

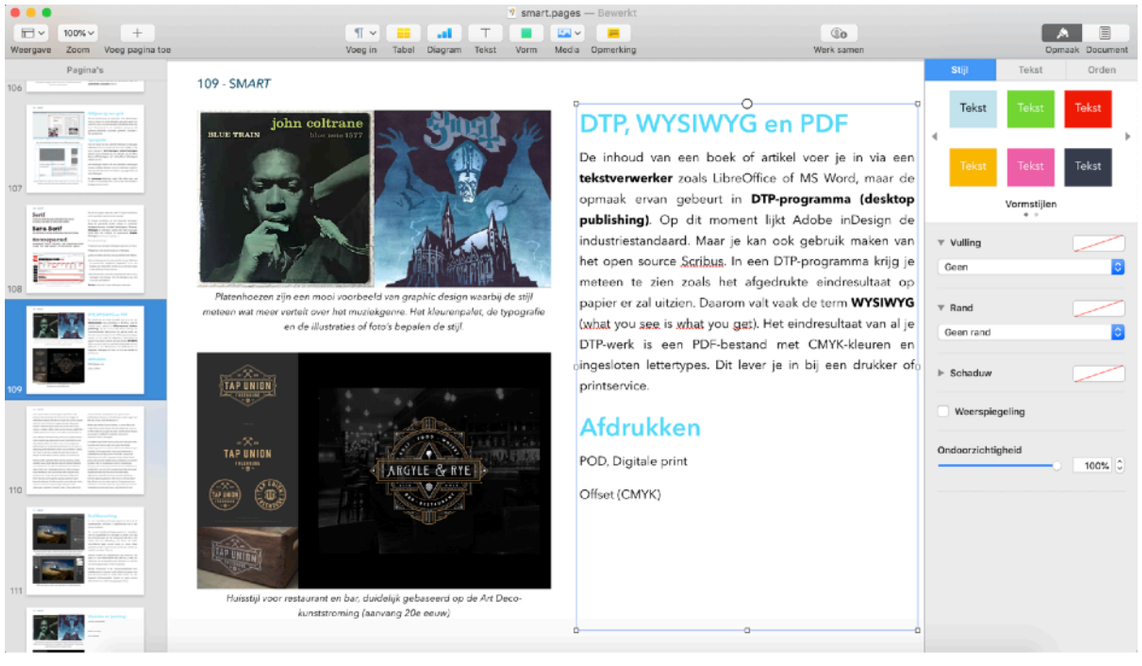
Een paar gouden tips:

- Gebruik nooit standaard lettertypes zoals Arial of Times.
- Beperk je in één stijl tot maximum 2 lettertypes.
- Vervorm letters niet (door ze bijvoorbeeld uit te rekken).
- Houd rekening met de auteursrechten op een lettertype Is commercieel hergebruik toegestaan? Is er een kostprijs aan verbonden? Anders zou je wel eens zwaar in de problemen kunnen komen.
- Als je bestanden uitwisselt, is het belangrijk dat je ook je lettertypes mee bezorgt. Niet elk lettertype is op elke computer geïnstalleerd.

Tot slot: je kan ook je eigen lettertypes ontwerpen.



DTP in het open source Scribus



Apple Pages heeft meer kenmerken van een DTP-programma dan van een klassieke tekstverwerker. Je ziet meteen het eindresultaat (WYSIWYG).

DTP, WYSIWYG en PDF

De inhoud van een boek of artikel voer je in via een **tekstverwerker** zoals LibreOffice of MS Word, maar de opmaak ervan gebeurt in een **DTP-programma (desktop publishing)**. Op dit moment lijkt Adobe inDesign de industriestandaard. Maar je kan ook gebruik maken van het open source Scribus. In een DTP-programma krijg je meteen te zien zoals het afgedrukte eindresultaat op papier er zal uitzien. Daarom valt vaak de term **WYSIWYG** (what you see is what you get). Het eindresultaat van al je DTP-werk is een PDF-bestand met CMYK-kleuren en ingesloten lettertypes. Dit lever je in bij een drukker of printservice.

Offset, digitale print en POD

Ruwweg bestaan er in onze tijd twee belangrijke druktechnieken: offsetdruk en digitale druk.

Bij **offsetdruk** maakt de drukker voor elke basiskleur (C, M, Y en K) een aluminiumplaat waarop de pagina is 'afgebeeld' . Die platen worden nat gemaakt. De zones op die plaat waar geen beeld is op aangebracht, trekken water aan. Daarna brengt de drukker een vettige inkt aan.

De gedeelten die water hebben opgenomen, stoten de inkt af. De gedeelten die droog zijn gebleven, nemen de inkt op. De inkten worden nu op een rubberen doek gedrukt. Dit rubberen doek drukt het beeld op papier. Voor elke kleur (C, M, Y en K) is er een rubberen doek. Het papier gaat achtereenvolgens langs die vier rubberen doeken waardoor de kleuren gemengd raken op het papier.

Digitale print daarentegen kan je vergelijken met een grote laserprinter. Dit toestel gebruikt geen inkt, maar toners. De kleuren worden op het papier gebakken.

De opstartkosten van digitale druk zijn veel lager omdat je geen aluminiumplaten moet maken waarop het beeld (tekst, foto's, opmaak...) wordt aangebracht. Bij offsetdruk betaal je dus altijd een opstartkost. De stukprijs zal dalen naargelang je een grotere hoeveelheid laat afdrukken. Een minimum aantal is vaak noodzakelijk om de opstartkosten rendabel te maken. Digitale druk zal duurder zijn in grote hoeveelheden omdat de stukprijs niet evenredig daalt.

POD (print on demand) of "short run digital printing" wordt steeds meer toegepast voor het drukken van kleine hoeveelheden boeken (tot 300 exemplaren). In tegenstelling tot offset-boeken, wordt het papier niet gevouwen en "genaaid", maar gelijmd. De kwaliteit van de lijm is echter dusdanig verbeterd dat de POD-boeken intensief gebruik doorstaan.

Offset

- Middelgrote tot grote oplages.
- Full colour drukwerk met Pantone- of PMS-kleuren.
- Drukwerk waar kleurechtheid erg belangrijk is.

Digitaal

- Kleinere oplages, vanaf 1 stuk.
- Grootformaat drukwerk.
- Spoedopdrachten, drukwerk dat je zeer snel moet hebben.
- Drukwerk dat je per stuk gepersonaliseerd wil hebben (nummering, adresgegevens, variabele data…).

Een bestand met drie lagen in Adobe Photoshop. Door lagen boven elkaar te plaatsen kan de graficus een samengesteld beeld maken (compositing).

GIMP is een gratis en open source alternatief voor Adobe Photoshop.

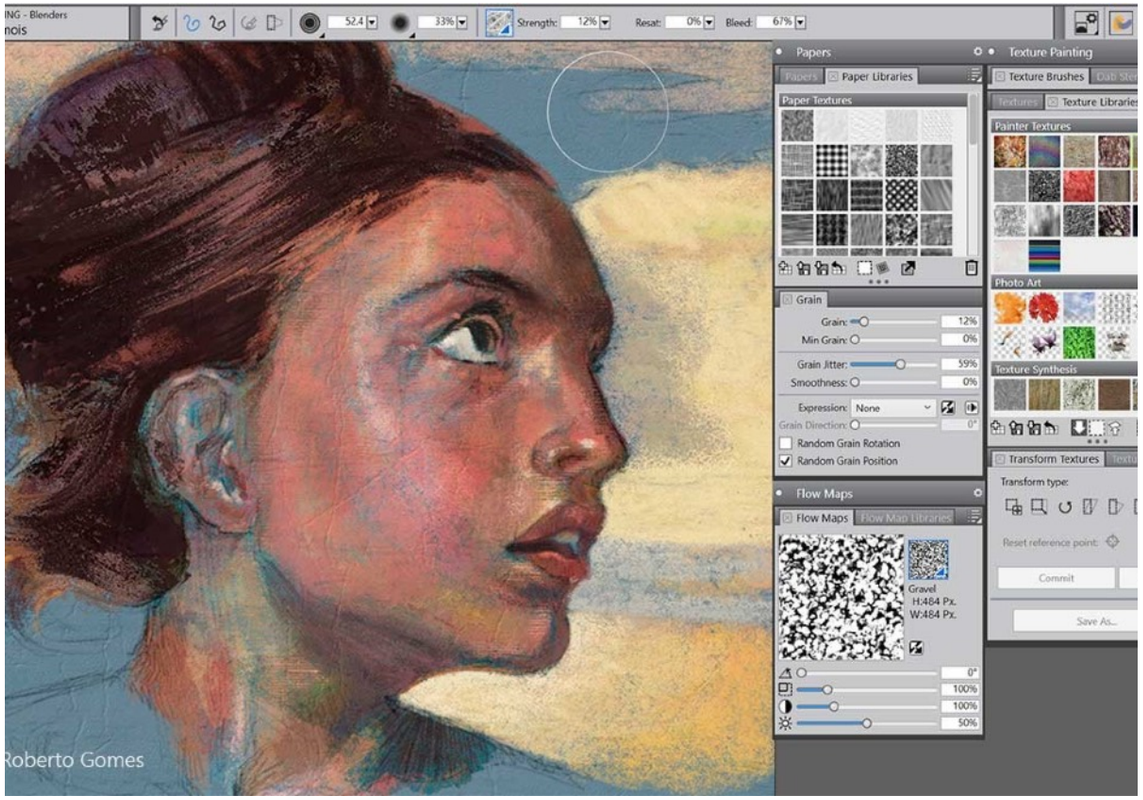
Beeldbewerking

In een beeldbewerkingsprogramma bewerk je pixelbestanden. Dit kwam al uitgebreid aan bod in een eerder hoofdstuk.

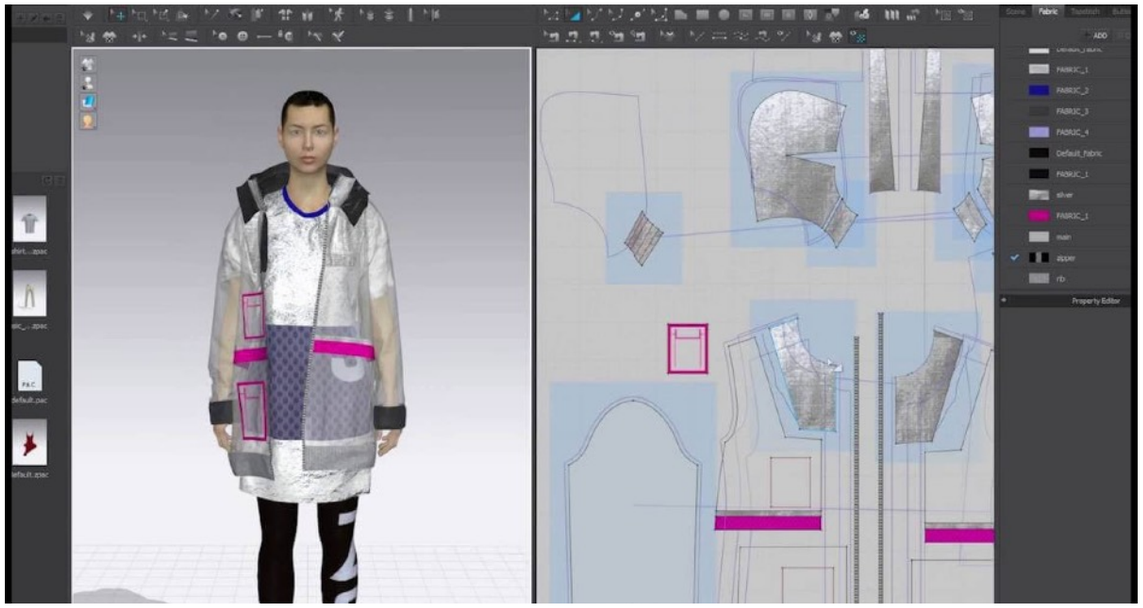
De meeste beeldbewerkingsprogramma's beschikken over de mogelijkheid om met lagen te werken. Een laag kan je beschouwen als een transparant vlak dat je kan vullen met een afbeelding, een kleur, een tekst. Verschillende lagen kunnen onder en boven elkaar geplaatst worden. Lagen kunnen verschoven worden ten opzichte van elkaar.

Filters en effecten breiden de mogelijkheden nog verder uit. Net zoals je in een tekstverwerker een deel van je tekst kan selecteren, kan je bepaalde pixels selecteren en enkel op die pixels aanpassingen of filters toepassen.

Adobe Photoshop is de industriestandaard voor beeldbewerking. Een goed grafisch vormgever, mag zich echter niet met handen en voeten laten binden aan een bepaald softwarepakket. Goede en open source alternatieven voor Photoshop zijn ondermeer Paint.net en Gimp.



In Corel Painter maak je levensechte schilderijen zonder verf.



CLO 3D, gespecialiseerde software voor het ontwerpen van mode en kleren.

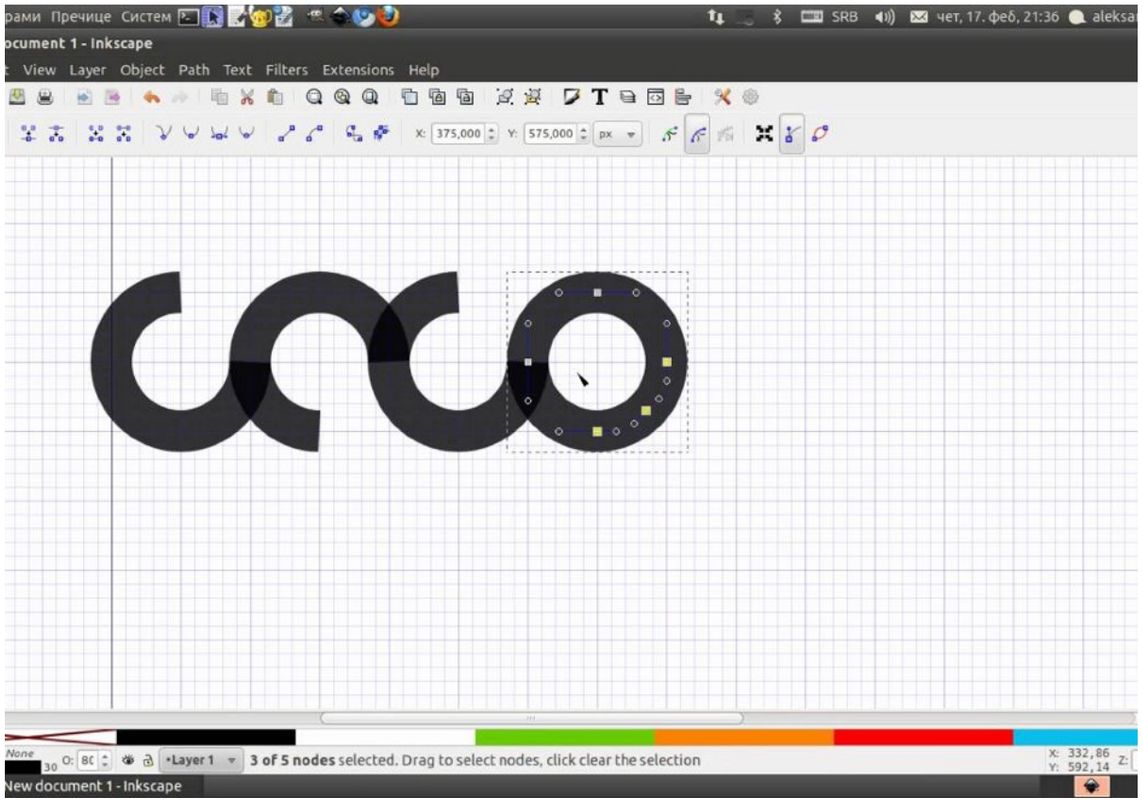
Paintingsoftware

Paintprogramma's vervangen de klassieke schildersezel. De programma's beschikken doorgaans over de meeste mogelijkheden van compositingprogramma's, maar breiden het gamma uit met digitale borstels, potloden, stiften en soorten verf… De grootste speler in deze markt is **Corel Painter**.

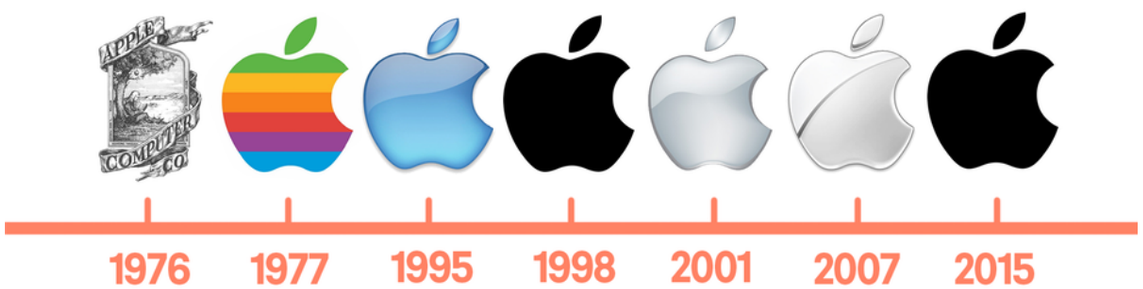
Deze programma's worden meestal gebruikt in combinatie met een drukgevoelig (hoe harder je drukt, hoe dikker de getekende lijn) grafisch tekentablet ter vervanging (of uitbreiding) van de klassieke muis.

Illustraties

Beeldbewerkingsprogramma's en paintingsoftware werken doorgaans met pixels. Programma's zoals CorelDraw, Adobe Illustrator en Inkscape werken met vectoren. Mode-ontwerpers, illustratoren of tekenaars die hun tekeningen en illustraties maken of inkleuren op de computer, kiezen de software die het best bij de opdracht past of die hun voorkeur wegdraagt.



Logo-ontwerp in het open source Inkscape



Logo's evolueren met de trends, maar behouden hun herkenbaarheid.

Logo-ontwerp

In het onderdeel "Vectorafbeeldingen" hebben we vermeld dat logo's worden getekend in vectortekenprogramma's als Adobe Illustrator, CorelDraw of Inkscape. Je werkt in dit geval niet met pixels. Een **logo** moet je immers zonder kwaliteitsverlies op een gevel, vrachtwagen en visitekaartje kunnen afdrukken.

Waar moet je op letten bij het ontwerpen van een logo? We geven je een aantal tips voor logo-ontwerp:

- Less is more. Keep It Simple, Stupid! (K.I.S.S.)
- Een logo hoeft niet perse duidelijk te maken wat je doet.
- Vermijd clichés. Het logo van een opticien moet geen bril bevatten.
- Een symbool moet niet, een "woordmerk" kan even sterk zijn.
- Past de uitstraling van het logo bij je business?
- Overdrijf niet met het aantal kleuren.
- Wees niet te trendy. Je logo moet een tijd meegaan. *Leave trends to the fashion industry*
- Het logo moet altijd en overal even herkenbaar zijn.
- Er moet een verhaal achter zitten. Waarom net dit logo?
- Zorg ervoor dat je logo veelzijdig is. Het moet herkenbaar blijven als je het verkleint of vergroot, als je het wijzigt van kleur of het woord weglaat.
- Vermijd standaard lettertypes.

Kleuren en hun symbolische connotatie in de westerse wereld. Dit geldt niet altijd in andere culturen.
(Bron: https://thelogocompany.net/blog/infographics/psychology-color-logo-design/)

BEWEGENDE BEELDEN: FILM & SPECIAL EFFECTS

Film is de illusie van bewegend beeld. Maar sinds de eerste filmvertoning in 1895 schept het de ongelooflijke mogelijkheid om herinneringen levensecht en voor altijd vast te leggen.

Een filmcamera maakt met een hoge snelheid foto's. Een film bestaat dus eigenlijk uit een reeks foto's die heel snel achter elkaar worden afgespeeld. Naast een filmspoor bevat de meeste film eveneens een audiospoor.

Dankzij de digitale revolutie is films maken en delen (vertonen) bijzonder makkelij. Bovendien kan je zonder al te veel voorkennis films bewerken, monteren en zelfs speciale effecten toevoegen.

Wil je echter professioneel aan de slag met film, dan moet je nog heel wat stappen verder gaan: een goed verhaal, cameratechniek, degelijk geluid, professionele montage…

Beeldsnelheid, timecode en FPS

De beeldsnelheid van een film drukt men uit in **frames per seconde (fps)** m.a.w. hoeveel beelden er per seconde aan het menselijk oog voorbijflitsen (=**framerate**) om de illusie van beweging te creëren.

Daarom tonen videoprogramma's naast het aantal uren, minuten en seconden, eveneens het aantal verstreken frames. De tijdcode (**timecode**) van een video ziet er als volgt uit:

uren:minuten:seconden:frames

Bijvoorbeeld 01:24:15:14 staat voor 1 uur, 24 minuten, 15 seconden, het veertiende frame...

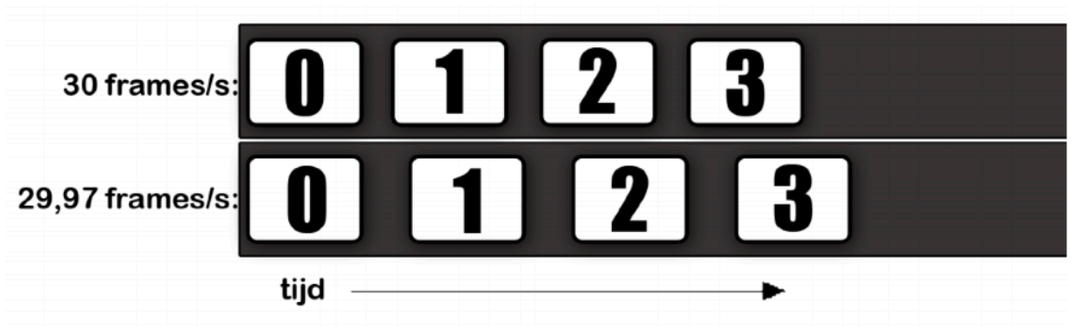
Bij het begin van elke seconde start de framerate terug van bij 0. Bij tijdcode 00:00:00:00 wordt het eerste beeldframe getoond. Een frame later staat de tijdcode op 00:00:00:01 enz.

00:00:00.00
00:00:00.01
00:00:00.02
...
00:00:01.00
00:00:01.01

Waarom is het belangrijk om dit te weten? Wanneer je een reclamespot maakt en **zendtijd** moet betalen op TV, dan betaal je per tijdseenheid. Daarom is het belangrijk om op een frame nauwkeurig te werken.

Hoeveel frames per seconde?

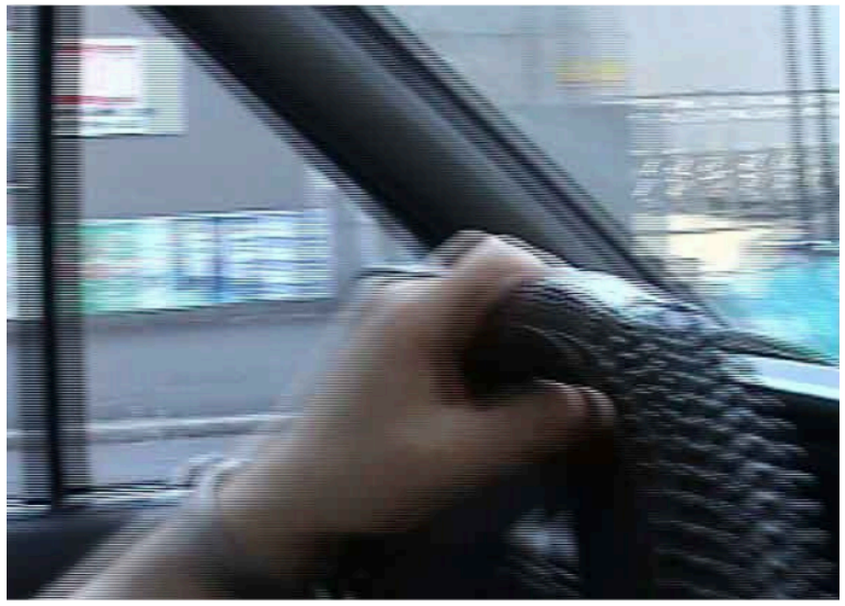
In de Verenigde Staten koos men voor de weergave van 2997 frames op 100 seconden. Waarom dit zo is, hoef je niet te weten. De uitleg is behoorlijk technisch en heeft te maken met het elektriciteitsnet dat wisselstroom met een frequentie van 60 Hz levert. Omgerekend betekent dit dat een Amerikaanse **NTSC-film 29,97 beelden per seconden** toont. In Europa (wisselstroom van 50 Hz) gebruikt het **PAL-formaat 25 fps**.



Interlacing, scan lines en fields

Een PAL-film telt 25 frames per seconde. Toch is het in werkelijkheid niet zo eenvoudig.

Een klassieke (PAL-)camera filmt 50 beelden per seconden en mixt ze tot 25 frames. De camera verdeelt het beeld in horizontale lijnen (**scan lines**). Om het wat begrijpelijker te maken, kan je die lijnen best zien als een genummerde lijst. In het ene beeld registreert hij alle even lijnen, in het andere beeld alle oneven lijnen.



Beide verzamelingen noemen we een veld (**field**). Vervolgens mixt de camera het veld van de oneven lijnen met het veld van de even lijnen. Het eindresultaat is één beeldframe. Het vermengen van beide velden noemen we **interlacing** (25 interlaced frames, 25 fps).

Omdat er een minimaal tijdsverschil is tussen de opname van beide velden, zit er ook een minimale verschuiving tussen de beelden in beide velden. Op een beeldbuistelevisietoestel zorgen de interlaced/interweaved velden voor een vloeiender beweging. Op een computerscherm (bijvoorbeeld wanner je film monteert in videomontagesoftware) zorgen interlaced frames voor vervelende lijnen in het beeld.

Toen televisie aan zijn opmars begon, bleek analoge technologie niet in staat om aan die snelheid te filmen. Televisie was niet in staat om beelden met zo'n snelheid weer te geven. Interlacing is een oude, maar niettemin zeer slimme compressiemethode die de benodigde bandbreedte halveert.

Progressive scan

De huidige generatie camera's en camerachips (in smartphones) filmen vaak in **progressive mode**. Ze maken geen gebruik van scan lines, maar filmen maar 12,5 frames per seconde! Bij weergave duurt elk beeld 2 frames. Zo kom je aan 25 fps!

Interlaced frames en progressive frames. De progressive scantechniek wordt gebruikt bij LCD-, Plasma-, DLP-, OLED flat panel- TV's en computermonitoren. (Bron: Samsung, https://www.lifewire.com/why-ntsc-and-pal-still-matter-1847856)

Motion blur wordt ook toegepast in games (GTA5). (Bron: https://www.youtube.com/watch?v=Y16jJflxcHc)

Meer frames zijn niet altijd beter

'Mijn smartphone kan films maken met 60 frames per seconde', hoor ik soms wel eens iemand bluffen. *So what?*

Bioscoopfilms tonen 24 non-interlaced/progressive frames per seconde. Dat is minder dan PAL en NTSC... en toch lijken de beelden vloeiend en helder. Zoals we eerder reeds zeiden, is film niet meer dan de illusie van beweging. Die illusie hangt dus niet noodzakelijk samen met de hoeveelheid frames per seconde.

Onze hersenen ervaren vervaging in het beeld als vloeiend. Hoe scherper de beelden, hoe schokkeriger het aanvoelt. Wanneer je een dicht bewolkte hemel filmt met vrijwel stil hangende wolken, zal de kijker dit zelfs bij een lage beeldsnelheid als vloeiend ervaren. Oorzaak is dat er vrijwel geen scherpe randen in voorkomen en de verschillen tussen de opeenvolgende frames minimaal zijn. Wanneer je een potlood snel heen en weer beweegt voor je ogen, zie je die niet meer scherp, maar vaag. Onze hersenen ervaren het niettemin als een vloeiende beweging. Een film met 50 scherpe *progressive* beelden zou je eerder als schokkerig ervaren dan een 24 fps-bioscoopfilm met wat **blur** (vervaging) op de beelden!

Flikkerende beelden

Onze ogen nemen de werkelijkheid vloeiend waar en delen die niet echt op in frames per seconde. Dit verklaart meteen waarom een filmbeeld en zeker een televisiebeeld kan flikkeren. Wanneer het ene frame het andere vervangt, is het scherm gedurende een fractie van een seconde even zwart. Aan 24 fps lijkt de film dan wel vloeiend (smooth), maar de 24 'verversingen' zorgen voor heel wat geflikker. De oplossing voor dit probleem lijkt op het eerste zicht nogal vreemd: de projector in de bioscoop toont/ververst elk frame 3 keer.

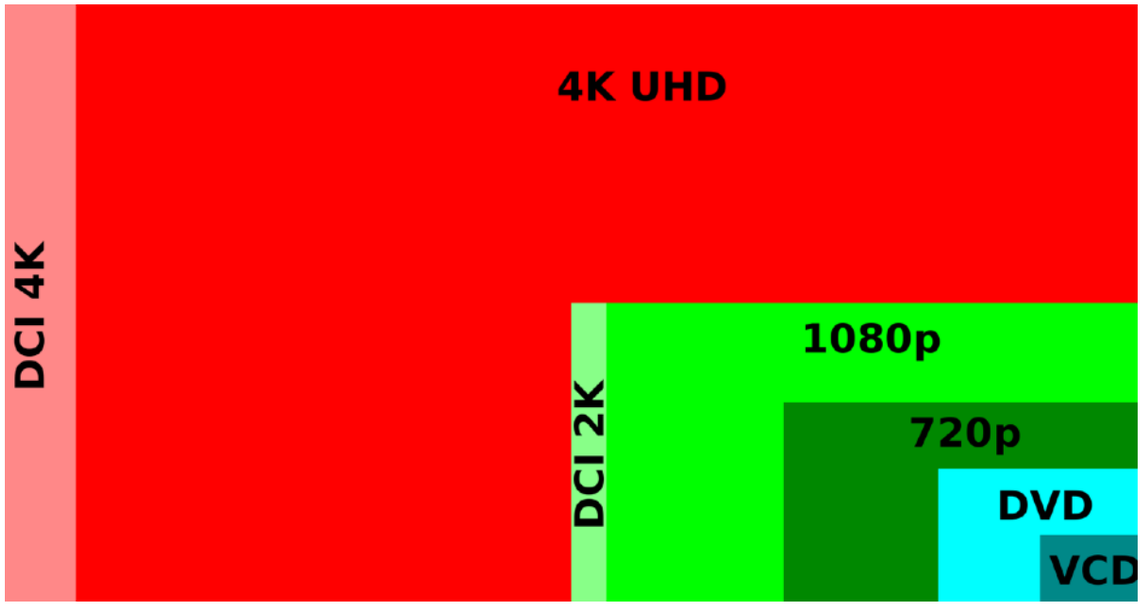
Hierdoor worden de zwarte overgangen iets korter en frequenter. Op die manier zie je in de bioscoopzaal films aan 72 fps... ook al zijn er in werkelijkheid maar 24 progressieve afbeeldingen. Een TV-scherm past diezelfde techniek toe. Hoe hoger de Hz of verversingsgraad (**refresh rate**), hoe minder beeldflikker! Een TV van 100 Hz ververst zijn beeld 100 keer per seconde waardoor het geflikker nog nauwelijks zichtbaar is. Bovendien blijft een helder beeld een paar seconden op ons netvlies bestaan (**afterimage**). Kijk maar eens even in een lamp en je ervaart meteen wat we bedoelen! Door die nabeelden winnen heldere beelden het van de zwarte beelden tussen de frames.

Onderzoek heeft uitgewezen dat onze ogen een lichtflits van 1/220e van een seconde kunnen registreren. Wanneer je een tv-scherm van opzij bekijkt, valt het geflikker zelfs bij een hoge verversingsgraad nog op. Om een scherm volledig flikkervrij te maken moet het waarschijnlijk pulseren aan een 500e deel van een seconde.

Meer frames helpen enkel bij hogere verversingsgraad

Games hebben er alle baat bij een monitor met een hoge verversingsgraad te gebruiken. Enkel dan zijn hoge framerates (hoge fps) zinvol. Een gaming-monitor haalt tegenwoordig 240 Hz.

Bij standaard computergebruik volstaat 60 Hz. Wat betekent dit precies? Het zorgt ervoor dat je scherm 60 keer per seconde ververst. Indien dit niet zo was, zou je niet merken dat je je muiscursor verplaatst of zou je niet zie, dat er letters verschijnen als je een tekst intikt in een tekstverwerker.



Filmresoluties: verschil tussen DVD/PAL, HD en 4K.
(Bron: https://en.wikipedia.org/wiki/4K_resolution)

Als je een DVD afspeelt op een 4K-scherm, wordt de film uitgerekt, maar de resolutie blijft laag (meestal 720x576).
(Bron: https://www.komando.com/tips/421949/when-to-buy-sd-hd-or-uhd-streaming-movies-and-tv-shows)

Kwaliteit en grootte

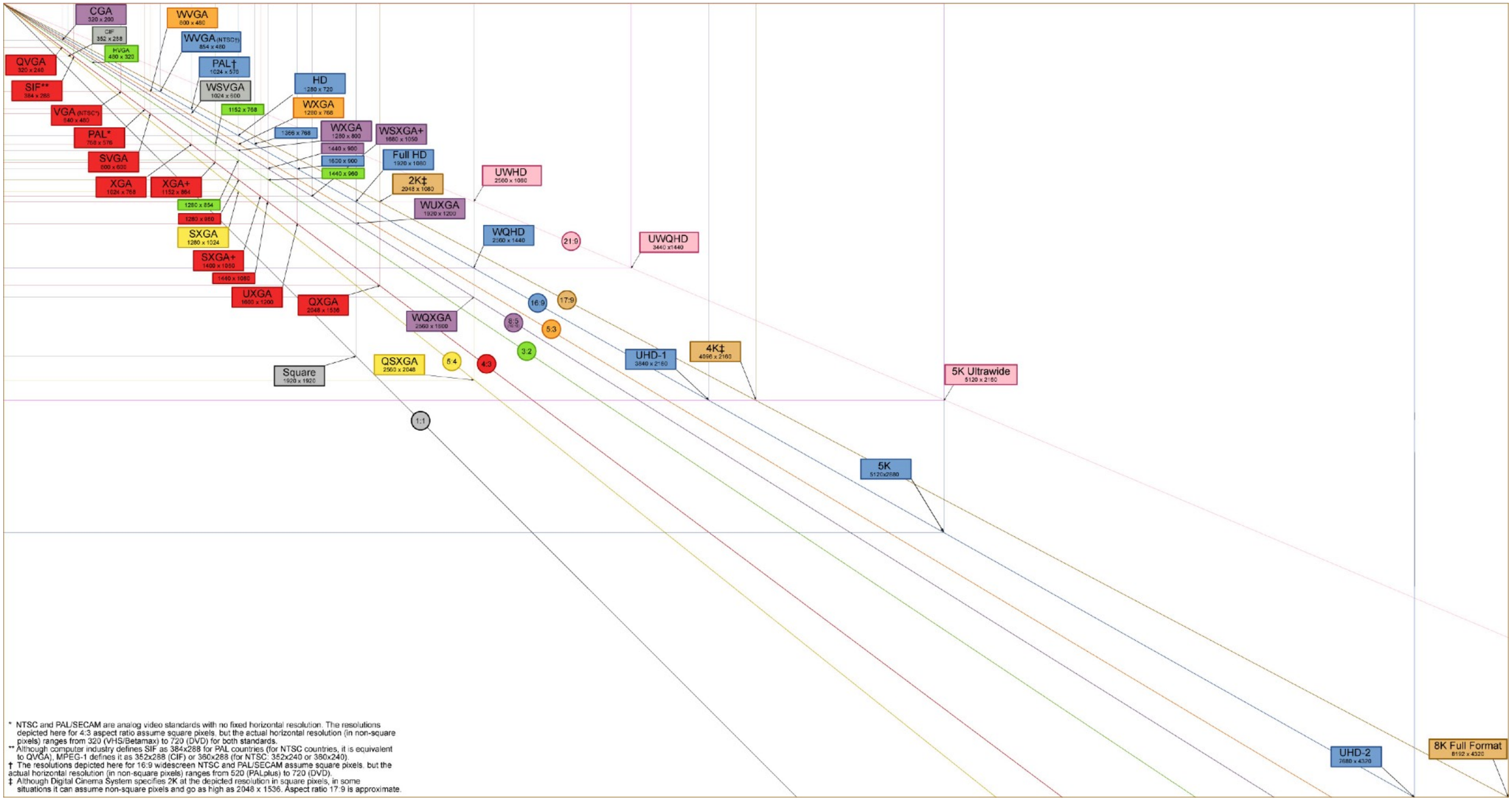
Hoe groter de weergave (of scherm), hoe groter de hoeveelheid benodigde pixels of resolutie.

Analoge film en vroege digitale film:

Pixels / resolutie	Type film / medium
352×240	Video CD
333×480	VHS, Video8, Umatic
350×480	Betamax
420×480	Super Betamax, Betacam

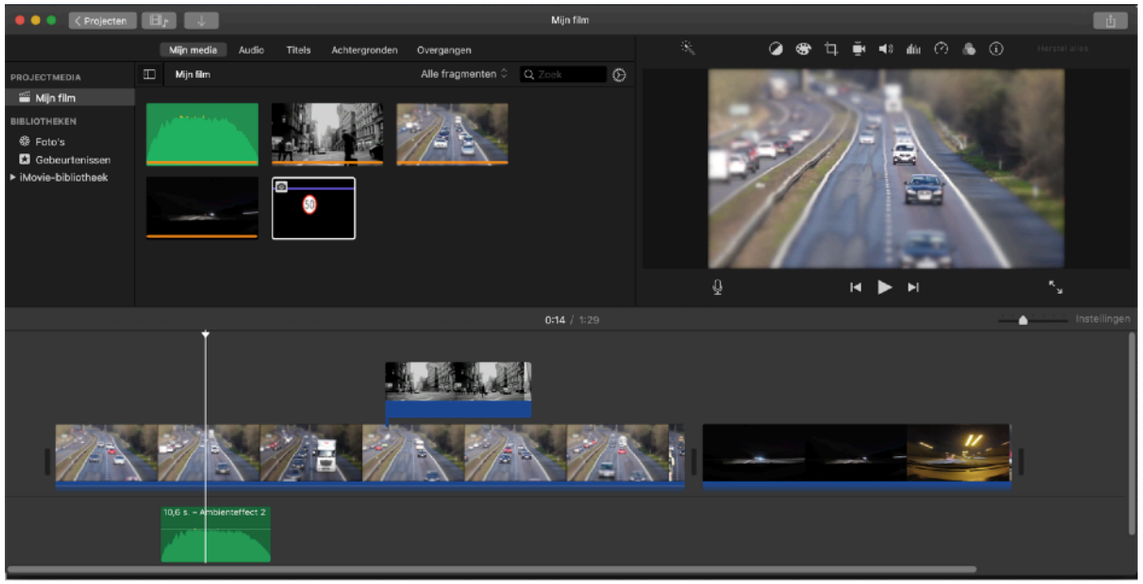
Digitale film:

Pixels / resolutie	Type film / medium
500×480	Digital8
720×480	D-VHS, DVD, miniDV, Digital Betacam (NTSC)
720×480	Widescreen DVD (anamorphic) (NTSC)
720×576	D-VHS, DVD, miniDV, Digital8, Digital Betacam (PAL/SECAM)
720×576	Widescreen DVD (anamorphic) (PAL/SECAM)
1280×720	D-VHS, HD DVD, Blu-ray, HDV (miniDV)
1440×1080	HDV (miniDV)
1920×1080	HDV (miniDV), AVCHD, HD DVD, Blu-ray, HDCAM SR
1998×1080	2K Flat (1.85:1)
2048×1080	2K Digital Cinema
3840×2160	4K UHDTV, Ultra HD Blu-ray
4096×2160	4K Digital Cinema
7680×4320	8K UHDTV
15360×8640	16K Digital Cinema
61440×34560	64K Digital Cinema



Filmresoluties: verschil in verhoudingen en resoluties bij filmformaten.
(Bron: https://en.wikipedia.org/wiki/4K_resolution)

Je koopt een 4K-beeldscherm voor in je woonkamer, maar je gaat gewoon DVD of TV kijken? Tja, dan ben je er aan voor de moeite (en voor je investering). Als je een PAL-film kijkt, gaat de kwaliteit niet plots omhoog omdat je een groter scherm hebt gekocht. Neen, het scherm rekt je film uit. Dat merk je meteen aan de kwaliteit. Hetzelfde geldt voor het bekijken van Ultra HD in Netflix. Enkel de duurste accounts beschikken over die mogelijkheid!



Film monteren in het instapprogramma iMovie van Apple.

Meerdere film- en audiofragmenten én lagen in Final Cut Pro X.
(Bron: https://nofilmschool.com/2011/04/full-resolution-screenshots-of-final-cut-pro-x-and-the-importance-of-avoiding-lock-in)

Film monteren

De meeste camera's in onze tijd leveren digitale bestanden. Het probleem van de conversie van analoog filmmateriaal naar digitaal, is dus veel minder groot dan pakweg 10 jaar geleden. Dit betekent dat je na het filmen snel aan de slag kan met **montage**.

Monteren betekent dat je filmbestanden, maar ook audio en afbeeldingen aan een **filmproject** kan toevoegen. Vervolgens kan je stukken een een filmbestand selecteren (knippen) en aan een **tijdlijn** toevoegen.

Belangrijk: verzamel al je beeldmateriaal en audio in één map (eventueel met submappen). Montagesoftware haalt immers het benodigde bronmateriaal steeds op van je harde schijf. Als je je montagebestand nadien wil aanpassen en de software vindt een bepaald bestand niet meer, kan dit voor aardig wat problemen zorgen.

Je kan titels, transities, audio… toevoegen en animeren. De meeste software laat toe om filters op je beelden of audio toe te passen of correcties (volume, kleuren) uit te voeren.

Chroma-key met green screen in Star Wars.
(Bron: https://wallpapersafari.com/star-wars-green-screen-backgrounds/)

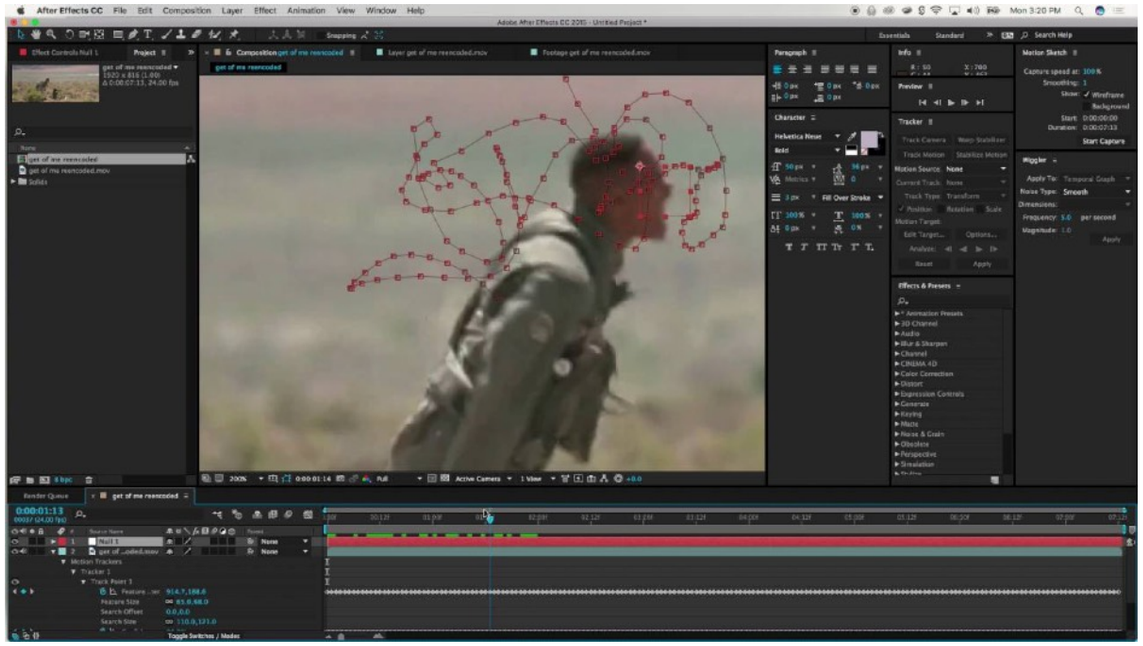
Splitscreen of pip in Final Cut Pro.
(Bron: https://www.youtube.com/watch?v=Qj5-u07ccGY)

Keying, 'greenscreen', overlay…

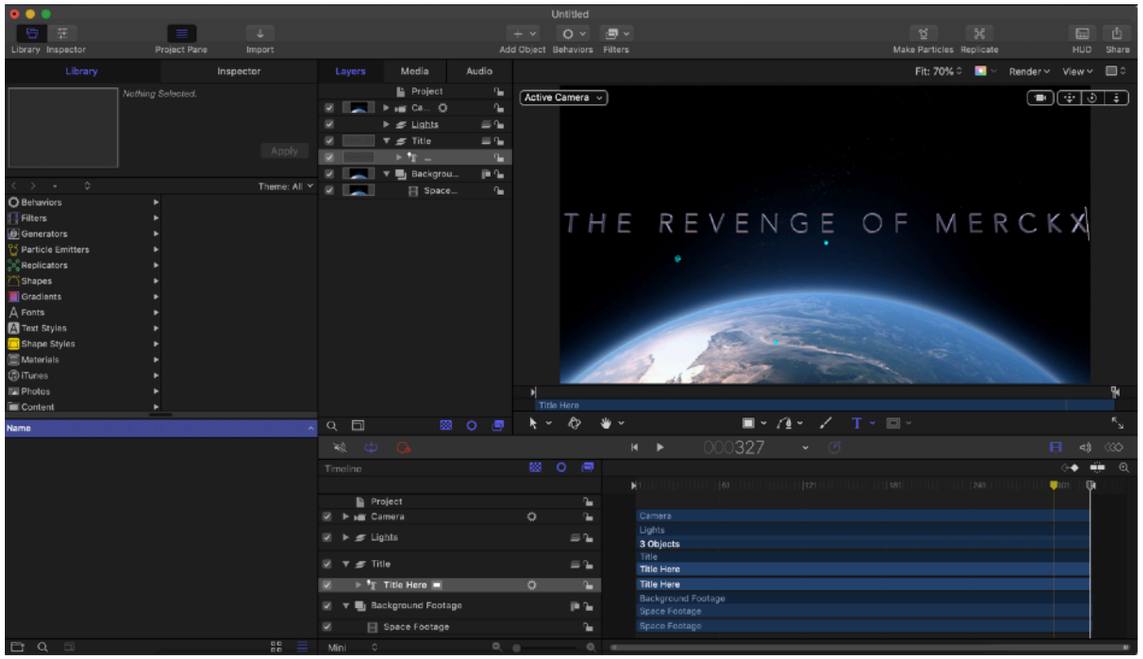
In professionele montageprogramma's zoals Adobe Premiere of Final Cut Pro, kan je meerdere lagen films boven elkaar in je **tijdlijn** toevoegen. Dit schept heel wat mogelijkheden.

Keying is een bijzondere compositietechniek. Hierbij kan je bepaalde delen van een videolaag transparant maken. Dit kan op basis van de belichting of een bepaalde kleur (=**chroma-keying**). Heel bekend is **greenscreen/ bluescreen**. Hierbij film je een bepaalde scene voor een groen of blauw scherm. De software filtert dan het groen of blauw weg waardoor de onderliggende laag zichtbaar wordt. Deze techniek past men heel vaak toe in de professionele filmwereld. Groen of blauw zijn het meest in gebruik omdat ze nauwelijks voorkomen in het menselijk lichaam.

De mogelijkheid om met lagen te werken, laat je toe om titels toe te voegen of **PIP**-effecten te voorzien **(Picture in picture)** zoals je wel eens in nieuwsuitzendingen ziet of een **splitscreen**. Maar je kan ook **overlay**-effecten toepassen waardoor je een bepaalde kleurfilter of belichtingseffect plakt op een onderliggende laag.



Motion tracking in Adobe AfterEffects.
(Bron: https://www.youtube.com/watch?v=5fZ2KkuS9Gk)



Animaties in Apple Motion.

Special effects

Montagesoftware als Adobe Premiere en Final Cut Pro X bevat al heel wat toeters en bellen. Wil je nog meer, heb je behoefte aan **speciale effecten (SFX)**, uitgebreide **animatie**mogelijkheden, **motion tracking**… dan moet je uitwijken naar Adobe AfterEffects of Apple Motion…

Bij motion tracking kan je bijvoorbeeld een bepaald object in een filmbeeld volgen. Zo zou je een rijdende auto in een filmbeeld kunnen "volgen" en vervangen door een transparante afbeelding (bijvoorbeeld een PNG) van een ufo…

Daarnaast kan je animaties, **particle systems** (voor het produceren van effecten als regen, bliksem…) toevoegen.

Filmformaten: codecs en containers

Doorheen de jaren zijn er diverse manieren bedacht om video digitaal te coderen. Een **codec** is een softwarematige manier om video en audio in bits en bytes (nullen en enen) om te zetten. Een codec kan die bits omgekeerd ook weer omzetten in beelden en audio.

De face2face-software maakt realtime reenactment mogelijk.
(Bron: https://www.semanticscholar.org/paper/Face2Face%3A-real-time-face-capture-and-reenactment-Thies-Zollh246fer/

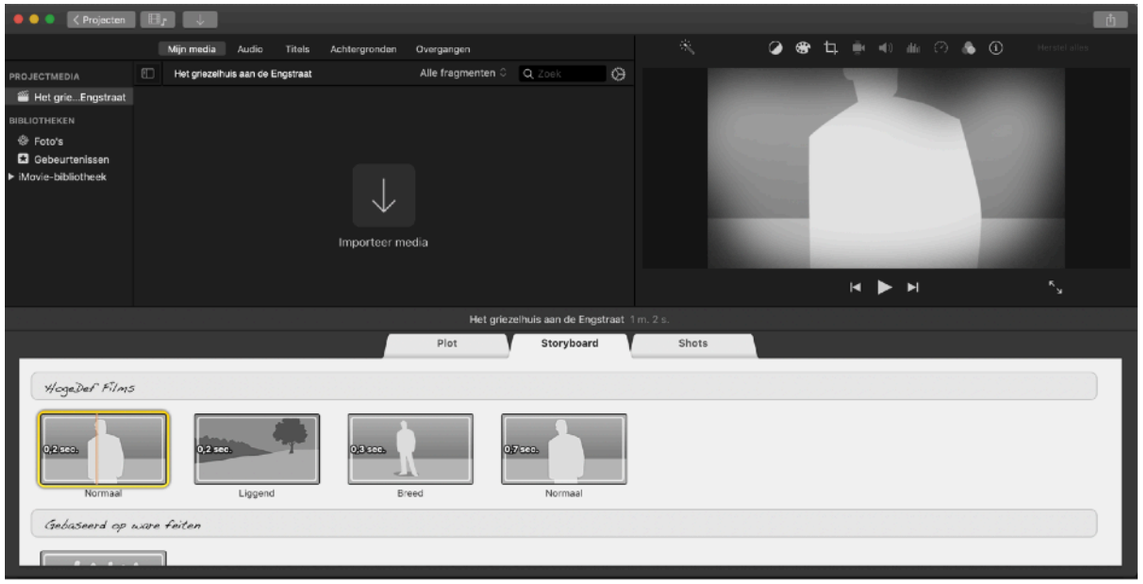
Interactive Dynamic Video (Abe Davis)
(Bron: https://www.ted.com/talks/abe_davis_new_video_technology_that_reveals_an_object_s_hidden_properties)

Codec staat immers voor COderen/DECoderen. Een **container** bewaart het videospoor en één of meerdere audiosporen in één 'verpakking'. Sommige containers kunnen eveneens ondertitels, menu-indelingen, hoofdstukken,... bewaren. Aan de bestandsextensie herkent u welke container voor de digitale video is gebruikt, al weet u daarmee nog lang niet welke codec voor de audio en/of video is gebruikt. De meest gebruikte codec in onze tijd is **H.264 / MPEG-4 AVC**.

De toekomst van film

Omdat film niet meer is dan digitale informatie opent het ongekende mogelijkheden. De face2face-software laat **realtime reenactment** toe. Hiermee kan de gezichtsexpressie van een acteur of persoon in een bestaand videofragment vervangen worden door de gezichtsexpressie van een andere (live) acteur.

Stel dat je een kamer filmt, maar je wil een storm simuleren waardoor de gordijnen aan een openstaand raam heel erg wapperen. Computerwetenschapper Abe Davis ontwikkelde een techniek waardoor je minuscule vibraties van objecten met een doordeweekse camera kan vastleggen. Vervolgens kan je die elementen met je computermuis animeren (**interactive dynamic video**).



iMovie biedt je een aantal sjablonen voor trailers. Daarin zitten standaard een aantal storyboards. Ze helpen je om een professionele trailer te maken. Als je dat wil, kan iMovie zelf op zoek gaan naar de meeste geschikte beelden in je filmmateriaal.

Het camerastandpunt en de begrenzing van het beeld, verschillen naargelang het verloop van je plot.
(Bron: https://www.independent.co.uk/arts-entertainment/films/halloween-2015-can-you-guess-the-horror-film-from-a-still-a6715366.html)

Filmtips

Als je zelf aan de slag wil gaan met film, is het geen overbodige luxe om wat cameratechnieken aan te leren, maar ook belichting en geluidsopname.

Daarnaast blijft het verhaal belangrijker dan de filmtechniek zelf. Begin dus steeds met een goed **plot** en een **storyboard.**

- Gebruik geen transities.
- Vermijd in- en uitzoomen tijdens het filmen.
- Kijk naar andere films. Laat je inspireren. Ook hier geldt wat voor grafische vormgeving van toepassing is: haal inspiratie uit de filmtraditie.
- Vertel een verhaal.
- Start met een storyboard.
- Heb oog voor detail.
- Bouw je verhaal op met een spanningsboog.
- Wissel af in beeldweergave. Een close-up kan je bijvoorbeeld gebruiken op een spannend moment.

…

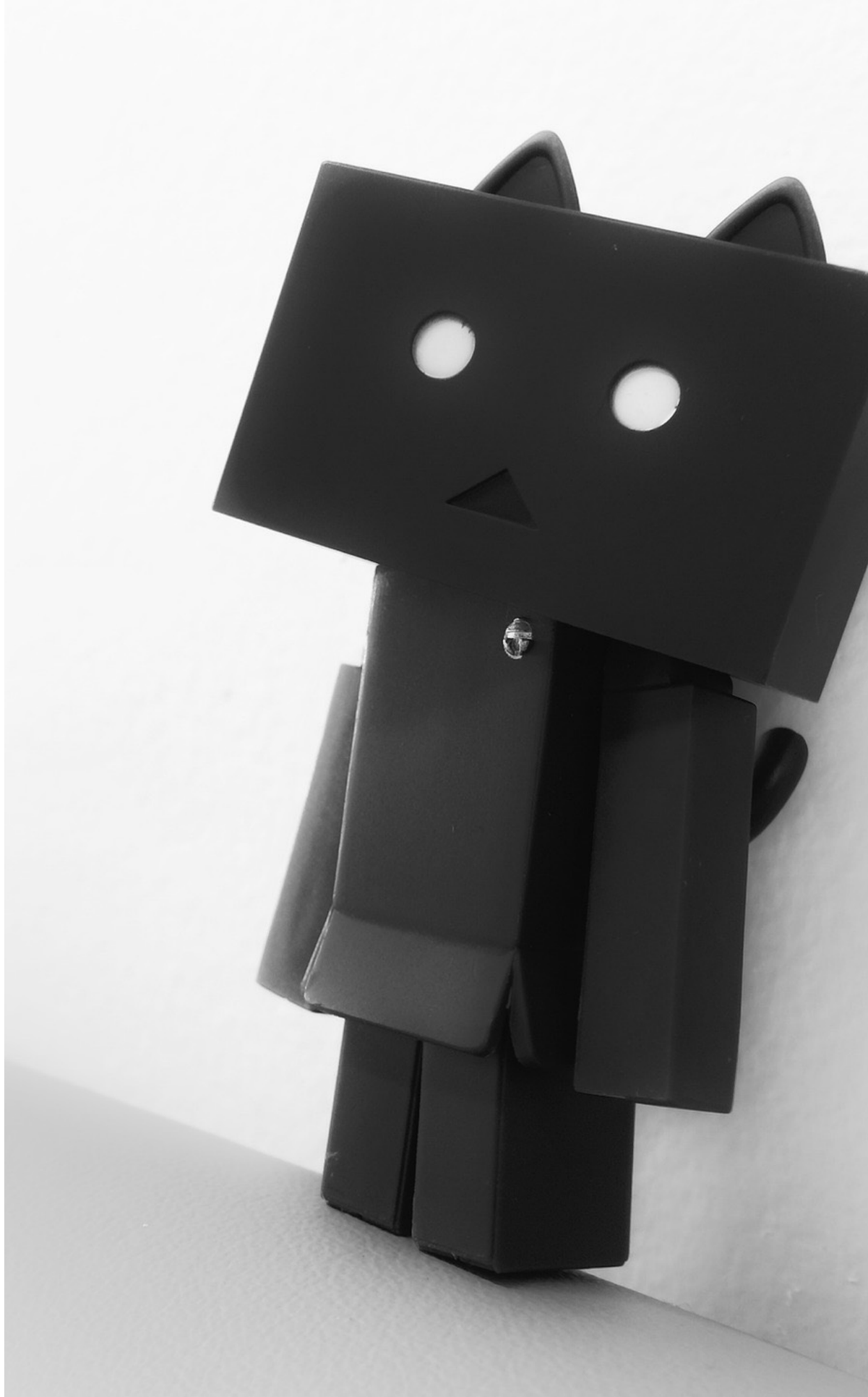
BEELDEN MET EEN ZIEL: ANIMATIE

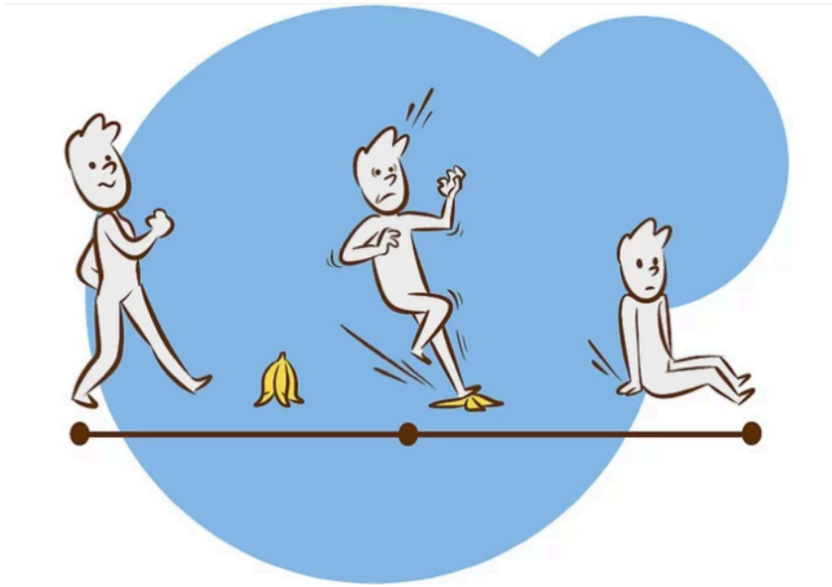
Met wat fantasie en enige kennis van filmtechnieken bouw je snel professioneel ogende animaties. Animatie geeft uw films letterlijk een eigen ziel.

Welke diverse animatietechnieken bestaan er? Hoe kan je ze inzetten in je filmprojecten, voor advertenties, social mediacampagnes…?

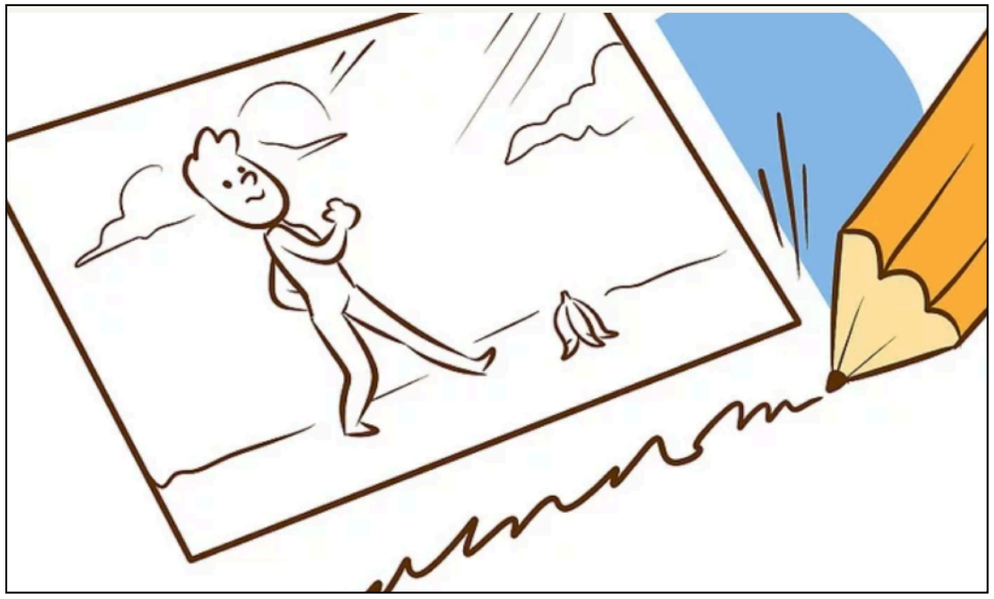
Het begrip animatiefilm is zeer ruim en dekt verschillende ladingen. Het kan gaan om een klassieke tekenfilm (full animation, limited animation), stop motion, claymation, pixilation...

Afhankelijk van de gewenste techniek zal je veel of weinig met de computer en software aan de slag moeten. De nabewerking, montage en opslag zal echter wel met behulp van een computer gebeuren. Maar één ding moet duidelijk zijn: animeren is arbeidsintensief!





De belangrijkste gebeurtenissen op een tijdlijn.
(Bron: https://nl.wikihow.com/Een-storyboard-maken)



Een storyboard bevat meerdere "cellen" (kadertjes).
(Bron: https://nl.wikihow.com/Een-storyboard-maken)

Storyboard

Welke techniek je ook wil gebruiken: animatie begint altijd met een goed verhaal. Je begint niet zo maar zonder meer aan een animatie.

Je bedenkt eerst een leuk verhaaltje, zelfs al moet je enkel en alleen maar een advertentie maken. Daarna zet je je verhaal om in een storyboard.

Een **storyboard** ziet er uit als een soort ruw stripverhaal waarin je tekent welke beelden (**shots**) achtereenvolgens zullen getoond worden.

1. Maak een lijst met de belangrijkste gebeurtenissen en bepaal hiervoor een **tijdlijn**. Wat wordt er achtereenvolgens getoond?

2. Bepaal de belangrijkste scenes in het verhaal.
   Opgelet: het is niet de bedoeling dat je een soort "flipbook" maakt met daarin alle frames. Je maakt eerder een soort stripverhaal.

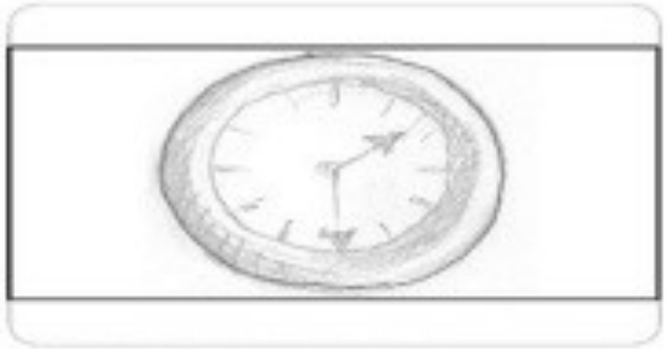
3. Naast elke tekening schrijf je wat er te zien is, welke beeldtechniek gebruikt wordt (close-up, long shot). Je vermeldt eveneens welk geluid er te horen is (tekst, muziek…). Vermeld dus ook de dialogen.

ST. STEPHEN'S HIGH SCHOOL
HIGH SCHOOL DEPARTMENT

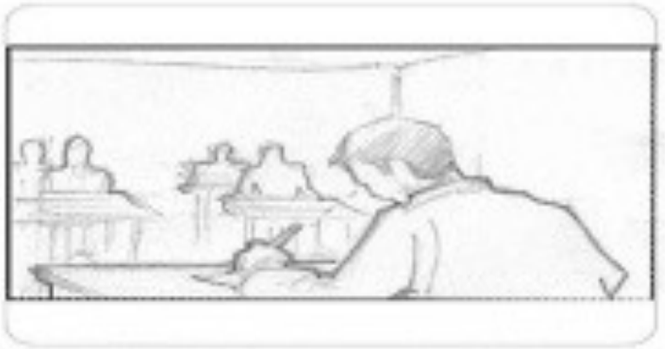
SHS Media and Information Literacy (MIL)
Performance Task No. 4 – Storyboard for the Video Project

Production Company: **Production Staff:**	**Grade and Section:**	**Teacher:**
	Working Title:	

SHOT 01-WS of whole classroom. Professor telling students that they have ten minutes left. as he looks up at the clock behind him.



SHOT 02-CU of clock in the classroom



SHOT 03(A)-MS of professor's back in foreground. students in background. He is looking at his paperwork, then at his watch...

SHOT 03 (B)-CU of professor as he is distracted by the taping of his feet. He then looks down at his shoes.

SHOT 04-POV of professor looking down at his shoes

SHOT 05-MS of professor's back as he begins rolling his ankles under his desk.

(Source: http://bluevalecareers.weebly.com/video-project.html)

Een voorbeeld van een storyboard.

(Bron:https://www.slideshare.net/arnielping/video-project-storyboard-example)

Aan de hand van een storyboard bouwt een animator elke scene stap voor stap op. Elke in de scene aanwezige figuur wordt apart als een cyclus getekend. In het voorbeeld van de beer krijgen we zo de volgende tekeningen:

1. Een achtergrondafbeelding.
2. Een tekening van het emmertje.
3. Een cyclus van 7 tekeningen waarin de beer zucht (lijf gaat zachtjes op en neer).
4. Een cyclus van 19 tekeningen voor het hoofd.
5. Een cyclus van 5 tekeningen waarin de wolk van klein naar groot gaat.
6. Een cyclus van 12 tekeningen voor een bewegende gedachtewolk.

= totaal van 45 tekeningen voor een scene van 5 seconden. Hierbij is niet inbegrepen: de inhoud van de gedachtewolk.
De gedachtewolkcyclus kan in de animatie herhaald worden.

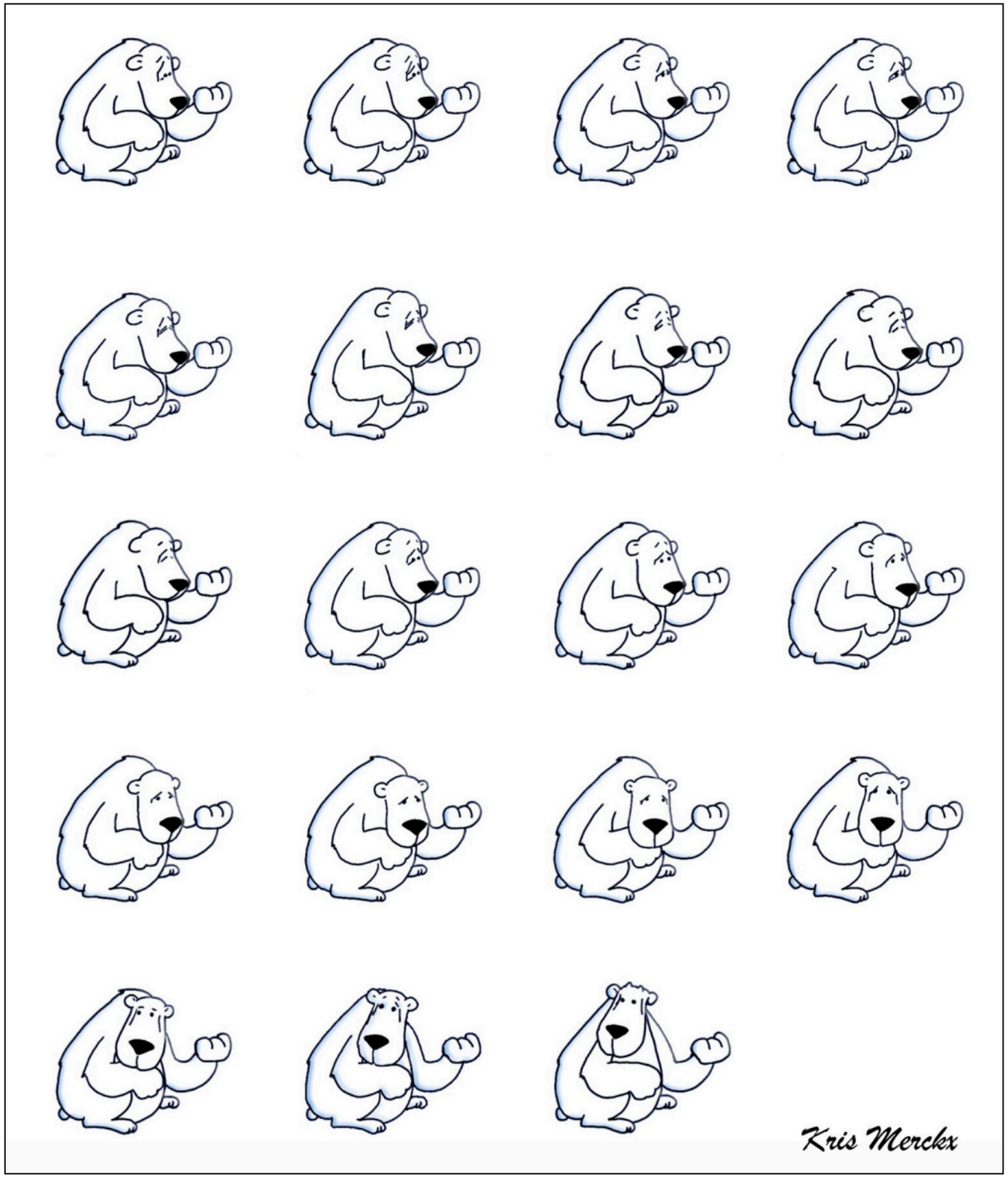
Tekenfilm gebruikt lagen

Vervolgens zet de animator het storyboard om in een echte animatie.

Net zoals een film, bestaat tekenfilm of animatiefilm uit een reeks beelden per seconde. Maar je kan het natuurlijk niet zo maar filmen met een camera.

Echte tekenfilm maken, gebeurt nog steeds met de hand. Dit betekent echter niet dat een animator slechts 25 verschillende tekeningen maakt voor elke seconde animatiefilm.

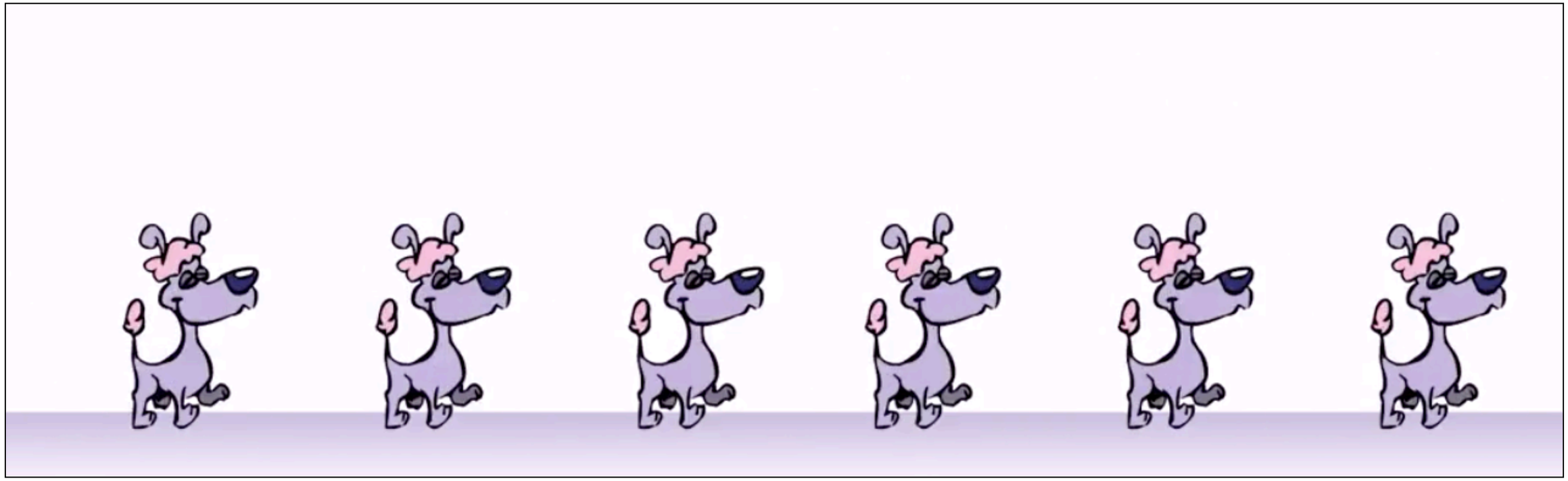
Een tekenaar maakt al evenmin volledige tekeningen. Hij/zij tekent de achtergronden en de figuren afzonderlijk. Dit maakt het mogelijk om bepaalde elementen opnieuw te gebruiken. Bij de eindmontage zal de animator de verschillende tekeningen op elkaar stapelen, als verschillende **lagen**. *Als je een ijsbeer tekent die enkel zijn hoofd draait, dan kan je één tekening maken voor het lijf van de beer, en meerdere tekeningen waarop enkel de beweging van het hoofd wordt getoond.*



Een tekenaar tekent niet de volledige frames. Als enkel het hoofd van een figuur beweegt, moet je niet het hele lichaam tekenen en al zeker niet de hele scene inclusief achtergrond.

De diverse tekeningen worden gescand en in een animatieprogramma in lagen geplaatst en gemonteerd. Van groot belang bij het scannen is dat je de tekeningen met de perforatiegaten in een peg bar (een speciaal metalen latje) op de **scanner** legt. Zo blijven alle tekeningen exact op dezelfde positie liggen. Jammer genoeg kan de meeste scansoftware (van goedkopere scanners of all-in-one-toestellen) geen vaste scanpositie onthouden. Je hebt immers niet altijd het hele blad nodig, want uw tekenfiguren nemen vaak slechts een fractie van uw papier in.

Vaak gebruikt de animator software voor het **inkleuren** van de gescande tekeningen. Ook het op elkaar **stapelen** van de diverse lagen en de **eindmontage** gebeurt over het algemeen met software.



Deze hondjes huppelen over het scherm. De animatiecyclus voor één hondje wordt slechts één keer getekend.
Met animatiesoftware wordt de animatiecyclus 6 keer herhaald.
(Bron: Wim Tilkin, Kristiaan Erlingen, Kris Merckx)

Heel vaak bestaat tekenfilm dus uit meerdere lagen tekeningen die op elkaar worden gestapeld. Een tekening kan ook meerdere frames worden aangehouden. Een PAL-film met 25 fps, zal dus niet noodzakelijk 25 afzonderlijke beelden per seconde bevatten. Is dit wel het geval, dan spreken we van **full animation**. Als beelden meerdere frames worden aangehouden, spreek je van **limited animation**.

Het vraagt behoorlijk wat oefening om te bepalen hoe lang elk frame en/of elke tekening moet aangehouden worden. Een film moet de juiste snelheid hebben, maar het vraagt tijd en vooral ervaring.

Stop motion met speelgoed.
(Bron: Joosfien Merckx)

De animatiestudio Aardman, is wereldvermaard.
(Bron: https://www.instructables.com/id/Make-A-Stop-Motion-Animation-For-Beginners/)

Stop motion

Als je beeld per beeld maakt met een fototoestel, spreken we doorgaans van **stop motion**.

Ruwweg kan je bij stop motion 2 technieken onderscheiden. "Cutout"-animatie (figuren worden uitgeknipt) lijkt qua resultaat op 2D-tekenfilm. Maar je kan ook stop motion maken met poppen of objecten. Dit lijkt qua opzet dus meer op 3D-animatie (volgend hoofdstuk).

De onderstaande lijst is zeker niet volledig:

Poppenanimatie
Een vorm van stop motion waarbij poppen en modellen worden gebruikt. In elke pop zit een *armatuur* (metalen skelet) waardoor de figuren gemakkelijk kunnen gemodelleerd worden. Elk frame wordt in volgorde van de animatie gefotografeerd. Belangrijk is ook te zorgen voor een egale constante belichting. Ray Harryhausen is ongetwijfeld het eerste grote talent op dit terrein. *Hij integreerde poppenanimatie in echte film. Deze vorm van poppenanimatie noemt men* ***model animation***.

Puppetoon
Poppenanimatie waarbij voor elk frame een nieuwe pop wordt gebruikt.

Clay animation (claymation)

Stop motion-animatie of poppenanimatie waarbij de figuren zijn gemaakt van een soort klei of plasticine rond de armaturen. Bekende voorbeelden zijn o.a. Wallace and Gromit en Bob de Bouwer. Zelf kan je gebruik maken van ijzerdraad en plasticine.

Cutout animatie

Stop motion-animatie waarbij je gebruik maakt van uitgeknipte figuren (van papier, stof...). Een tekening of foto van een persoon verknip je in al zijn onderdelen: romp, hoofd, bovenarmen, onderarmen, handen, bovenbenen, onderbenen, voeten... Met draadjes of klemmen maak je de onderdelen weer aan elkaar vast. Leg ze op een plat oppervlak met een egale kleur.

Objectanimation

Stop motion met voorwerpen. Heel populair zijn de zogenaamde *brickfilms* gemaakt met LEGO-blokjes en figuurtjes.

Pixilation

Het meest bekende voorbeeld is ongetwijfeld de videoclip bij Sledgehammer van Peter Gabriel. Bij pixilation kunnen foto's of film van mensen gecombineerd worden met voorwerpen. Bij de manuele vorm wordt een gezicht onder een glasplaat gestopt en de voorwerpen op de glasplaat. De voorwerpen worden verschoven bij elk gefotografeerd frame. Uiteraard kan je dit rechtstreeks op de computer met behulp van software.

Silhouet- of schaduwanimatie

De figuren en de onderdelen van de cutout-animatie zijn enkel als zwarte silhouetten te zien.

Andere technieken

Rotoscoping

Live-actie beeld per beeld traceren (natekenen, doortekenen). Wordt ook wel gebruikt als basis of inspiratiebron voor karakteranimatie.

Live-action animation

Een animatie wordt bovenop een echte film gelegd: als bijvoorbeeld een tekenfilmfiguur meespeelt tussen echte acteurs. Deze techniek wordt ook bij 3D-animatie veelvuldig gebruikt. De live-actionfilms van de smurfen en Alvin and The Chipmunks zijn hiervan bekende voorbeelde. Maar ook in films als Star Wars wordt hier veelvuldig gebruik van gemaakt.

Animatie is niet nieuw. Al een paar eeuwen geleden bedacht men animatietechnieken om te gebruiken in combinatie met een toverlantaarn (de voorloper van de beamer). Dit toestel bevat 2 glazen plaatjes. Op het ene plaatje staat een landschap met windmolen. Op het bovenliggende plaatje staan enkel de wieken. Als je aan de hendel draait, roteren de wieken.
(Bron:collectie Kris Merckx)

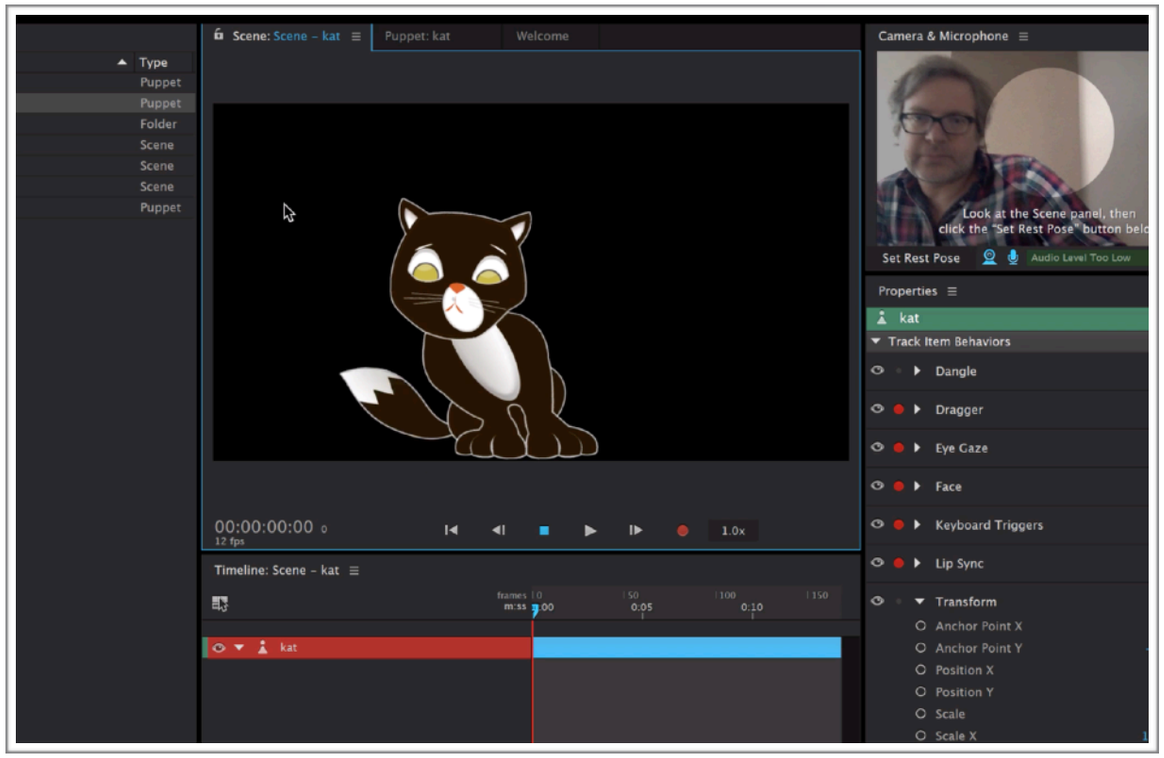
Cutoutanimatie via software, videoclip voor de groep 'Snakes in Exile'.
(Bron: Kris Merckx)

Graphic animation

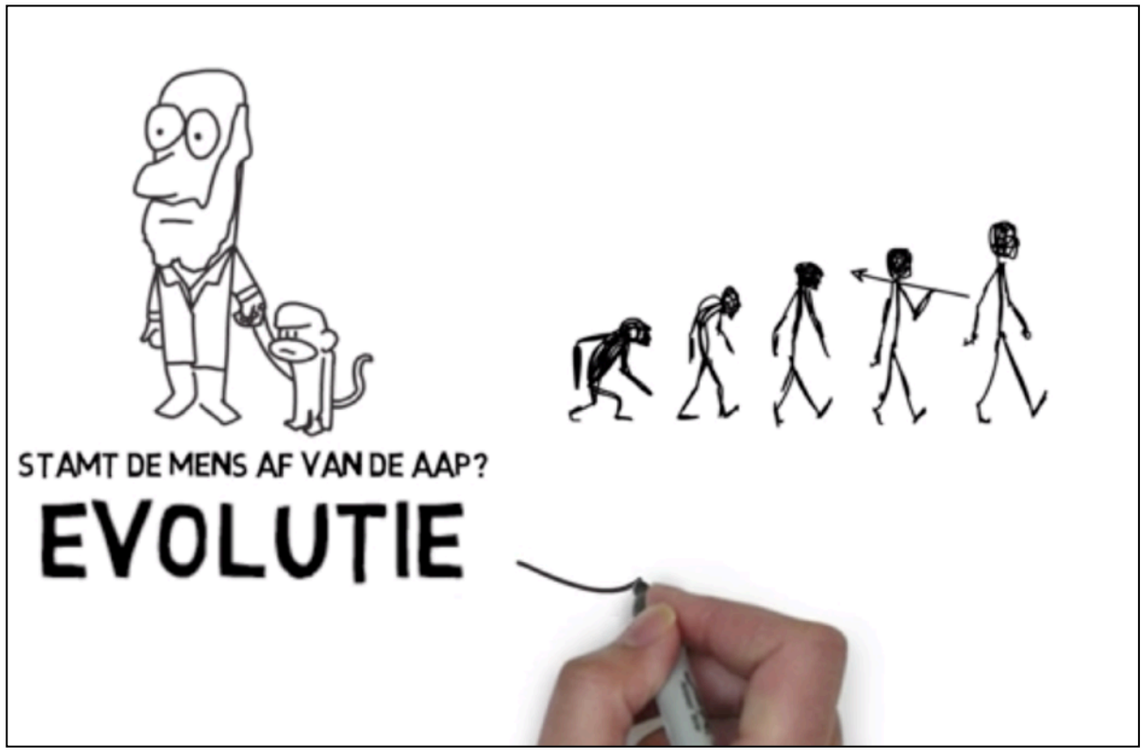
Graphic animation is niet meteen één specifieke techniek. Foto's, teksten, tekeningen… worden frame per frame geanimeerd. In animatiesoftware kan je dit soort animatie tegenwoordig heel snel maken. Anders dan bij stop motion moet je niet telkens een foto maken van elk frame. Per laag biedt de software je een **tijdlijn**. Op die tijdlijn kan je dan bijvoorbeeld een begin- en een eindpunt voor een beweging aanduiden. Je kan bijvoorbeeld zeggen: *in frame 10 start de auto. In frame 30 stopt de auto.* Beide frames zijn zogenaamde **keyframes**. De animator moet aan de software enkel tonen waar de auto in beide frames moet staan. De software zelf zorgt dan voor de rest.

Geavanceerde animatiesoftware biedt eveneens de mogelijkheid om elementen of lagen te groeperen. Op die manier kan je de ene animatie de "parent" van de andere ("child") maken. Bijvoorbeeld: als je een auto verplaatst, zal ook het geanimeerde mannetje in die auto, mee opschuiven.

Moeilijk is het als er ook **lipsynchronisatie** aan te pas komt. Dit betekent dat de mond van de figuren moet mee bewegen met de gesproken tekst.



Animatie van een gelaagd bestand op basis van de webcam met Adobe Character Animator.



Een voorbeeld van whiteboardanimatie.

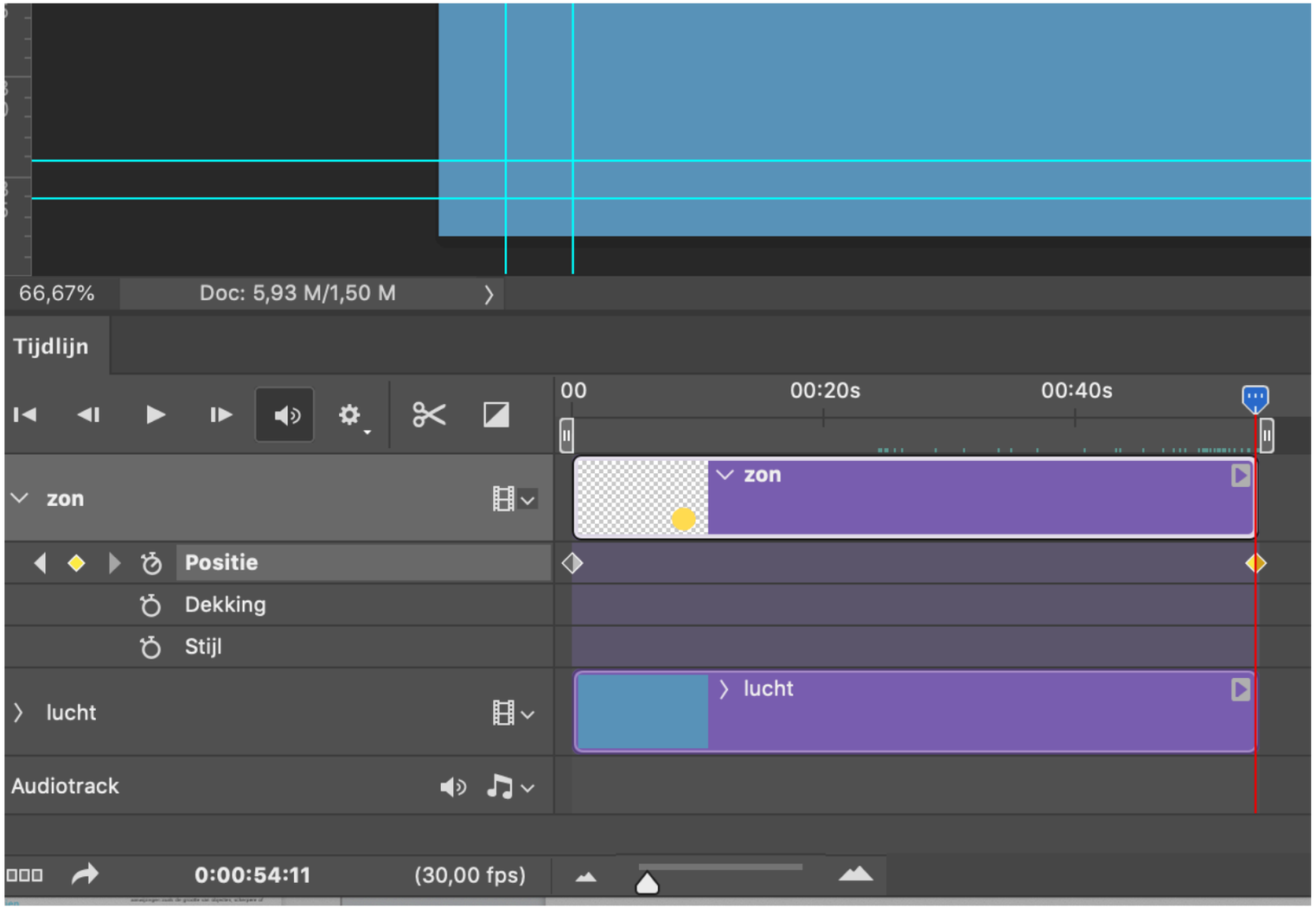
Nieuwe animatietools

Omdat er steeds steeds meer nood is aan korte animaties voor onder meer advertenties, pogen softwarefirma's als Adobe het bouwen van animaties te vergemakkelijken met animatietools zoals Adobe Animate.

Heel wat animaties op internet zijn gebouwd met "code", waardoor ze eveneens interactieve mogelijkheden bieden (zoals doorklikken of surfgedrag meten).

Echte 2D-animatie met ondermeer bewegende figuurtjes blijft echter moeilijk zonder met een potlood en pen aan de slag te gaan. **Adobe Character Animator** is een tool waarmee je relatief eenvoudig figuurtjes (over het algemeen in vooraanzicht) kan animeren. De software capteert je webcambeeld en brengt de beweging van jouw hoofd, gezichtsexpressie, lippen en bovenlichaam over op een getekende figuur. Het levert heel 'smooth' animaties op.

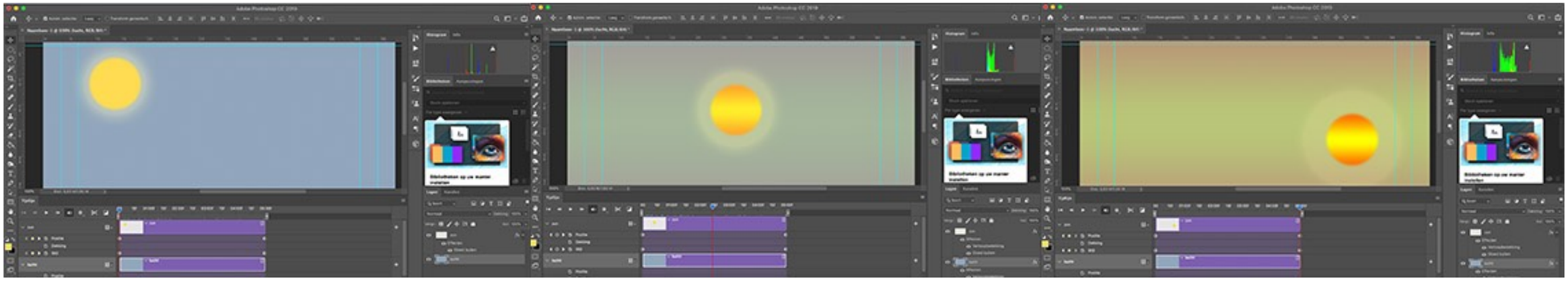
Heel bekend, maar vaak wel trendy, zijn zogenaamde **whiteboardanimaties** waarbij je een hand een animatie ziet tekenen. De software gebruikt hiervoor een tekening van een hand om vectortekeningen geanimeerd tevoorschijn te laten komen.



Elke laag krijgt in Adobe Photoshop, maar ook in Adobe AfterEffects, een eigen tijdlijn. ***Met keyframes kan je aanduiden waar een bepaald animatie-effect moet beginnen en eindigen.***

Je kan bijvoorbeeld de start- en eindpositie van een object in een laag aangeven. De software rendert (ontwikkelt) zelf de tussenliggende frames.

In Adobe Photoshop kan je naast de (x- en y-)positie van een object, ook de laagstijl en de transparantie van een laag animeren.

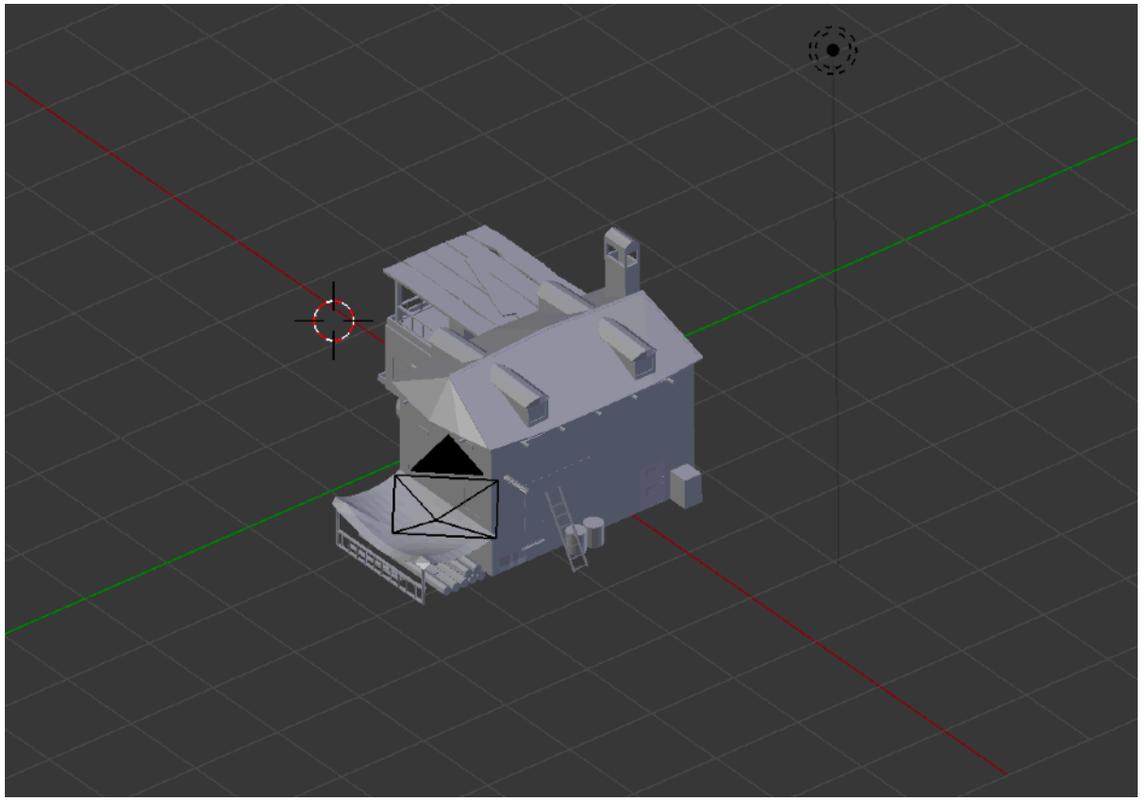


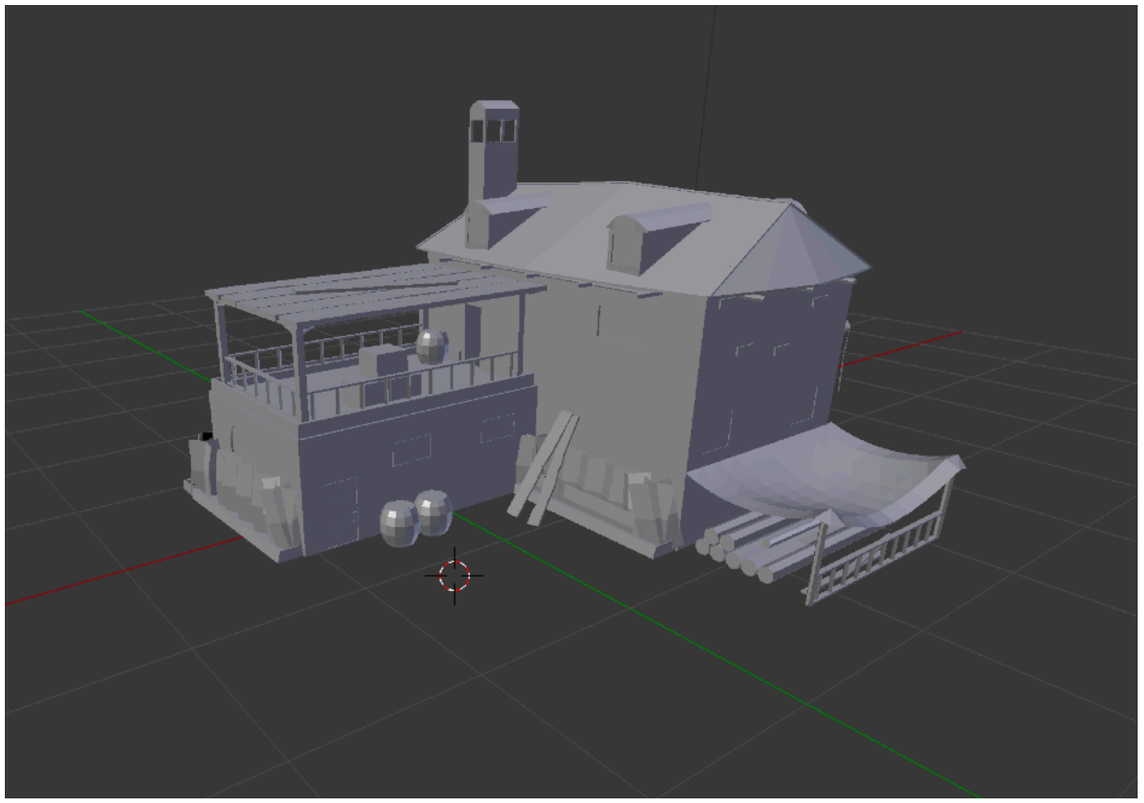
3D: BEELDEN IN DRIE DIMENSIES

De werkelijkheid om ons heen telt 3, sorry, 4 dimensies. Hoogte, breedte, diepte … en tijd. Je kan immers *doorheen de tijd* langs een stoel wandelen. Je ziet de stoel dan langs alle kanten.

Als je een foto maakt van een stoel, dan zie je enkel één kant van die stoel. Wanneer je die foto opent in bijvoorbeeld Adobe Photoshop, dan kan je niet de andere kant van de stoel bekijken. Net zo bij een foto van een mens. Als je iemands rug fotografeert, dan kan je Photoshop niet vragen om zijn gezicht te tonen.

Met 3D-software kan je 3D-modellen tekenen voor ontwerp, machinebouw, architectuur of film. Maar eenvoudig is het niet. Bovendien betekent 3D-software nog niet dat het eindresultaat van je werk meteen een 3D-object is. Vreemd? Lezen dan maar.





Soorten 3D

De term 3D dekt meerdere ladingen.

Een computerscherm of televisie geeft alle beelden weer in **2D: enkel met breedte en hoogte**. Ook een film toont de 3D-werkelijkheid op een plat 2D-vlak. In de bioscoop heb je niet het gevoel echt *in de film* te zitten. Je kan niet in de filmscene binnenwandelen en de acteurs eens langs een andere kant gaan bekijken.

Met een 3D-bril krijg je het gevoel dat de voorwerpen of de scenes waar je naar kijkt, wel degelijk "**tastbare diepte**" hebben. Zowel 3D-brillen in de bioscoop als virtual reality-headsets leveren zo'n resultaat.

Als je een foto of een film zou zien als een "update" van een klassiek schilderij, dan is 3D-software een update van "beeldhouwen". In 3D-software kan je bijvoorbeeld een huis tekenen en dit met je muis roteren om het langs alle kanten aan te passen of van deuren en ramen te voorzien.

Architecten, productontwerpers, maar ook filmmakers gebruiken 3D-software. Heel wat scenes uit films zoals Star Wars en Shrek zijn in 3D-software gebouwd. Maar als je het eindresultaat bekijkt op een gewone TV, dan zie je dit in 2D. Tenzij je een 3D-bril draagt dus…

3D zien

Waarom zien mensen in 3D? Waarom weten we dat bepaalde dingen zich verder of net dichterbij bevinden?

Een eerste belangrijke reden hiervoor is dat we onze beide ogen op hetzelfde voorwerp kunnen richten. Omdat onze beide ogen zich op ongeveer 6 cm van elkaar bevinden, ziet elk oog hetzelfde beeld vanuit een verschillend standpunt. Onze hersenen gebruiken die verschillen om de diepte te kunnen inschatten. In de biologie noemt men dit **binoculair zicht**.

Op een foto of schilderij krijgen we een **gevoel van diepte-afstand** door het **perspectief** of door dichterbij gelegen objecten met meer detail weer te geven.

Dieren zoals paarden zonder binoculair zicht, kunnen de afstand (of diepte) bepalen door het **parallax**-effect: Objecten dichter lijken sneller te bewegen dan objecten die verderaf zijn gelegen als je ze "passeert". Op een autosnelweg zullen de lantaarnpalen voorbijflitsen, maar een berg in de verte of de maan aan de hemel zal minder snel “bewegen".

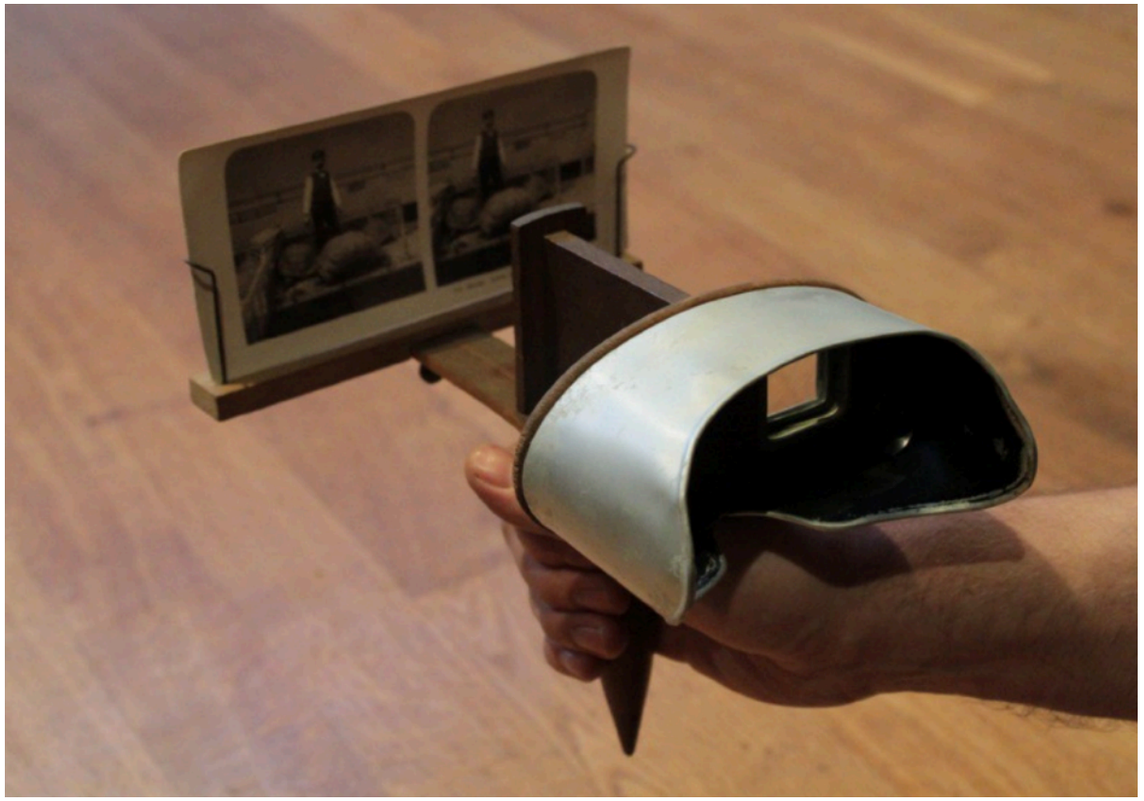
Onze hersenen kunnen diepte eveneens detecteren door aanwijzingen zoals de grootte van objecten, scherpere of vagere "texturen", zichtbare details.

3D en diepte simuleren

Doorheen de tijd zijn diverse technieken bedacht om 3D-effecten en **dieptezicht** te simuleren bij 2D-beelden of op platte oppervlakken. De bekendste technieken zijn **stereoscopie, anaglypische 3D en lensvormige 3D.**

Stereoscopie

Bij stereoscopie simuleert men de werking van het menselijk zicht door simultaan 2 beelden van dezelfde locatie te tonen met daartussen een verschuiving van een aantal centimeter. De standaard is 65mm omdat dit de gemiddelde afstand is tussen de pupillen van beide ogen. Onze ogen doen in principe hetzelfde: er zit een "klein verschil" in afstand tussen beide ogen, waardoor we het gevoel van diepte ervaren. Voor het waarnemen van stereoscopische '3D' moet de gebruiker door een speciale bril of toestel kijken. De komst van de fotografie tekende mee het succes van de **stereoscoop**. *In 1849 bracht David Brewster (1781 – 1868) de draagbare stereoscoop op de markt. Wij kennen het toestel vooral onder de latere merknaam “Viewmaster”.*



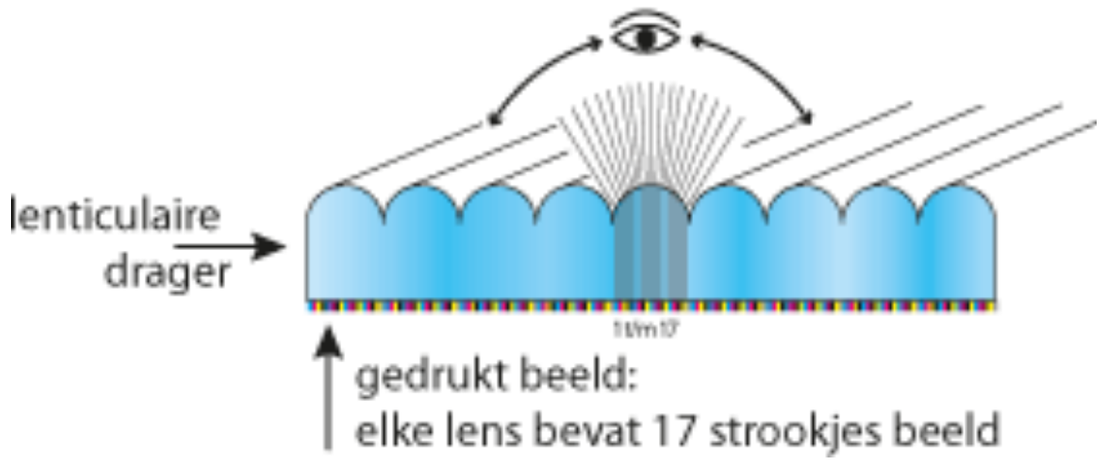
Een stereoscopisch beeld bestaat over het algemeen uit 2 afbeeldingen (een **stereogram**): één voor elk oog. Elke afbeelding toont dezelfde scene maar vanuit een licht afwijkend perspectief. Deze **side-by-side-techniek** zorgt voor een 'natuurlijk' binoculair zicht. Stereoscopie is relatief goedkoop omdat de afbeeldingen niet moeten worden bewerkt. Er bestaan fototoestellen en filmcamera's die stereoscopische beelden maken. Een nadeel van side-by-side stereoscopie is dat de gebruiker door een 'viewer' of 'bril' naar de beelden moet kijken.

Anaglypische 3D

Een **anaglyph** splitst een beeld op in 2 kleuren: een rood beeld voor het linkeroog en een cyaan (combinatie van groen en blauw) beeld voor het rechteroog. Het alom bekende rood-cyaan-brilletje laat enkel rood licht door bij het linkeroog en cyaan bij het rechteroog.

Een anaglyph kan zowel bekeken worden op een scherm als op papier, maar zonder bril levert het moeilijk te bekijken/herkennen beelden op.

Afbeelding bovenaan:19e eeuwse stereoscoop.
Afbeelding onderaan: Een analytische afbeelding.



Lensvormige 3D

Lensvormig drukwerk (lenticular print) heb je ongetwijfeld al wel eens gezien. De afdruk zit onder een soort geribbelde kunststof die nog het meest lijkt op transparant ribfluweel. Als je geen bril nodig hebt, zoals bij lenticular print, spreekt men ook van **autostereoscopie**.

Lenticular print levert niet noodzakelijk een 3D-beeld op, maar twee verschillende beelden om eventueel een gevoel van animatie te geven aan de kijker. Maar op de Nintendo 3DS zet men de techniek in om stereoscopische beelden (een stereogram) te genereren.

Een aantal **autostereoscopische** displays zoals de Nintendo 3DS slagen er zelfs in om het effect van **bewegingsparallax** te simuleren.

Afbeelding bovenaan: De werking van lenticular print.
Afbeelding onderaan: Een voorbeeld van lenticular print. Je kijkhoek bepaalt welk beeld je ziet.

3D-objecten en scenes modelleren

In **3D-modeling software** kan je virtueel 3D-modellen bouwen. Een architect kan huizen bouwen, een autofabrikant een 3D-model voor een nieuwe auto.

In 3D-modeling software kan je wiskundige 3D-modellen zoals kubussen, cilinders, bollen enz. (zogenaamde "**primitives**") tekenen, samenvoegen of de ene vorm uit de andere knippen. Op die manier kan je elk gewenst bestaand of fictief object in een 3D-model omzetten. De 3D-artist of -ontwerper kan een model roteren, er op in- of uitzoomen.

Eens een model gebouwd, kan hij het vullen met een kleur of **textuur**. Een textuur is in veel gevallen een foto uit de werkelijkheid. Zo kan je een 3D-model van de aarde tekenen door op een bol een foto van de wereldbol uit de ruimte te plakken. Als je een 3D-model van een huis tekent, kan je de muren voorzien van foto's van echte bakstenen.

Vervolgens kan de ontwerper **belichtingseffecten** toevoegen waardoor ook schaduwen ontstaan om een realistisch effect te creëren.

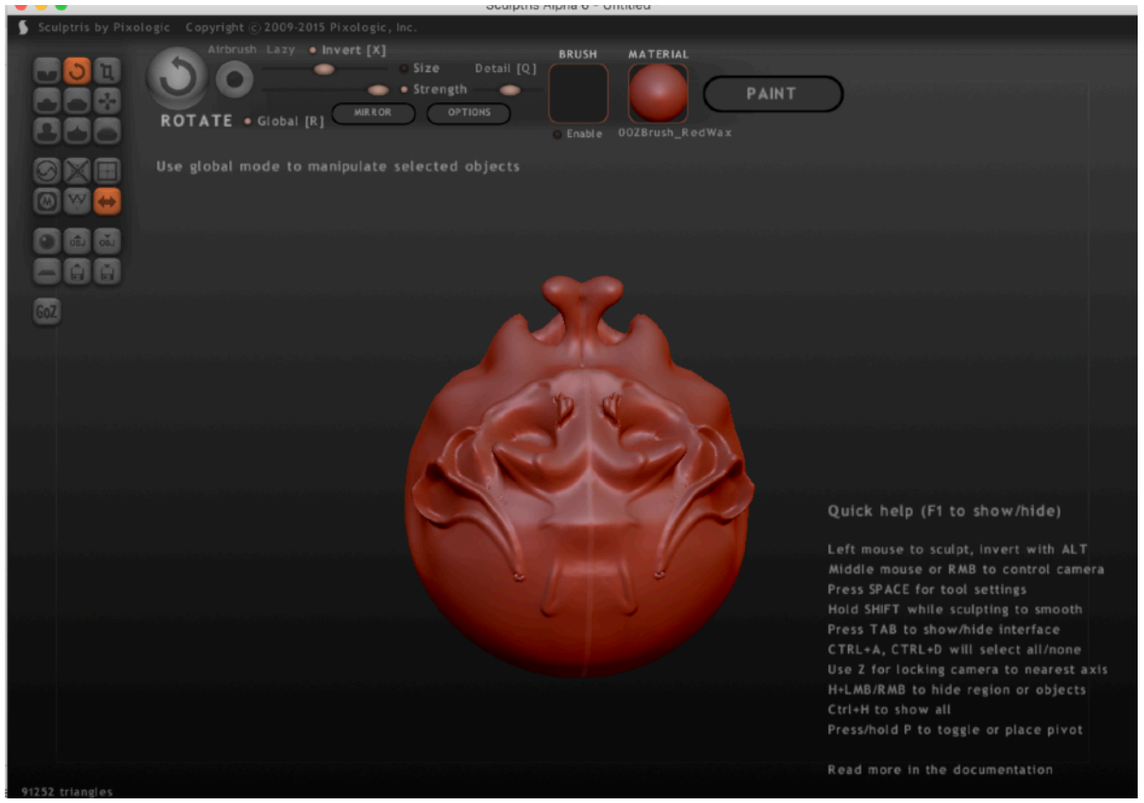
In **3D-animatiesoftware** kan je **virtuele camera's** in een 3D-scene plaatsen en "doorheen" de tijd de camera bewegen. Zo krijg je de indruk dat de camera doorheen echte beelden beweegt.

In de film- en animatiewereld wordt 3D-animatiesoftware veelvuldig ingezet. Terwijl de eerste generatie Star Wars-films nog veelvuldig gebruik maakte van maquettes, zitten de laatste films boordevol 3D-animatie.

Naast het werken met "primitives" kan je in 3D-modelleersoftware vaak ook **boetseren**. Als je menselijke figuren of dieren wil tekenen, is het niet altijd even makkelijk om enkel geometrische vormen te gebruiken. Je kan de figuren dan virtueel boetseren. Vervolgens kan je de modellen voorzien van een **"skelet" (bones)** dat je kan animeren. Via **motion capture** is het mogelijk om de bewegingen van een echte acteur over te brengen op een 3D-model.

Rendering

Om de 3D-scenes en animatie om te zetten in levensechte beelden met texturen, belichting, schaduwen… moet de computer de beelden "**renderen**". Dit vergt heel veel rekenkracht. In de filmwereld gebruikt men hiervoor in netwerk gekoppelde computers (een **renderfarm**).



Landschapsgeneratoren

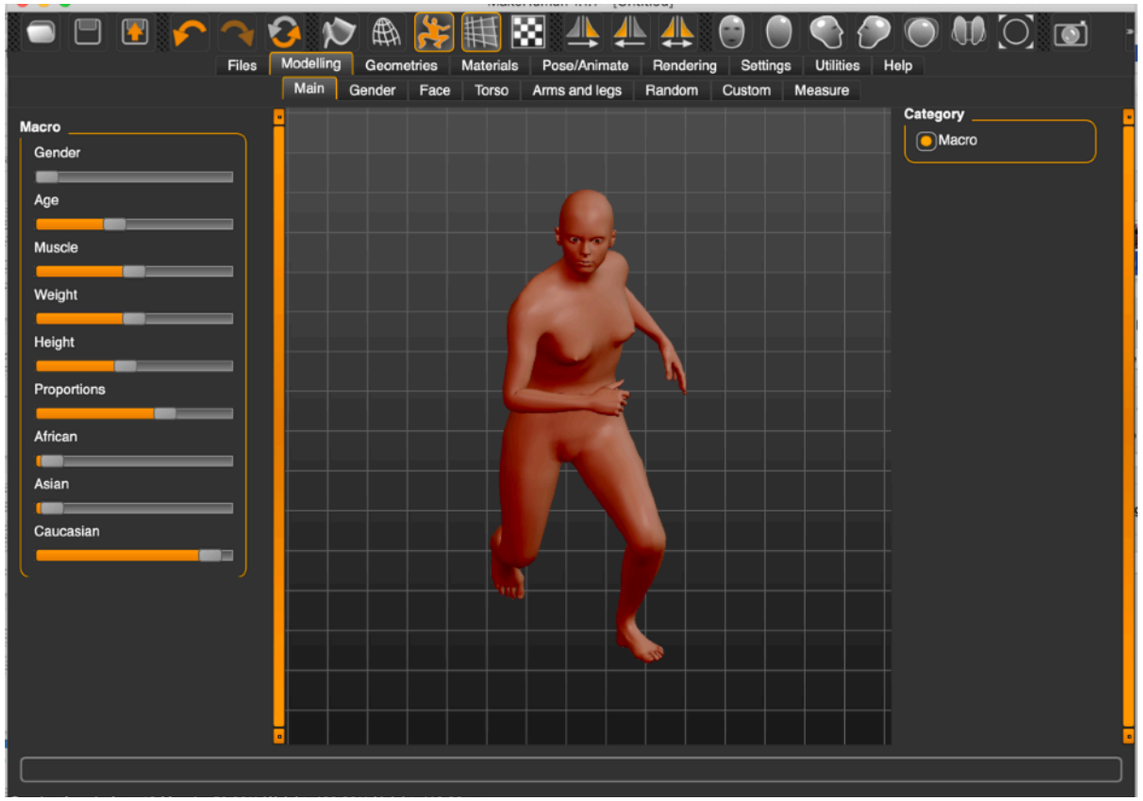
Met een **landschapsgenerator** kan een 3D-artist hyperrealistische landschappen creëren die nauwelijks van echte landschappen te onderscheiden zijn.

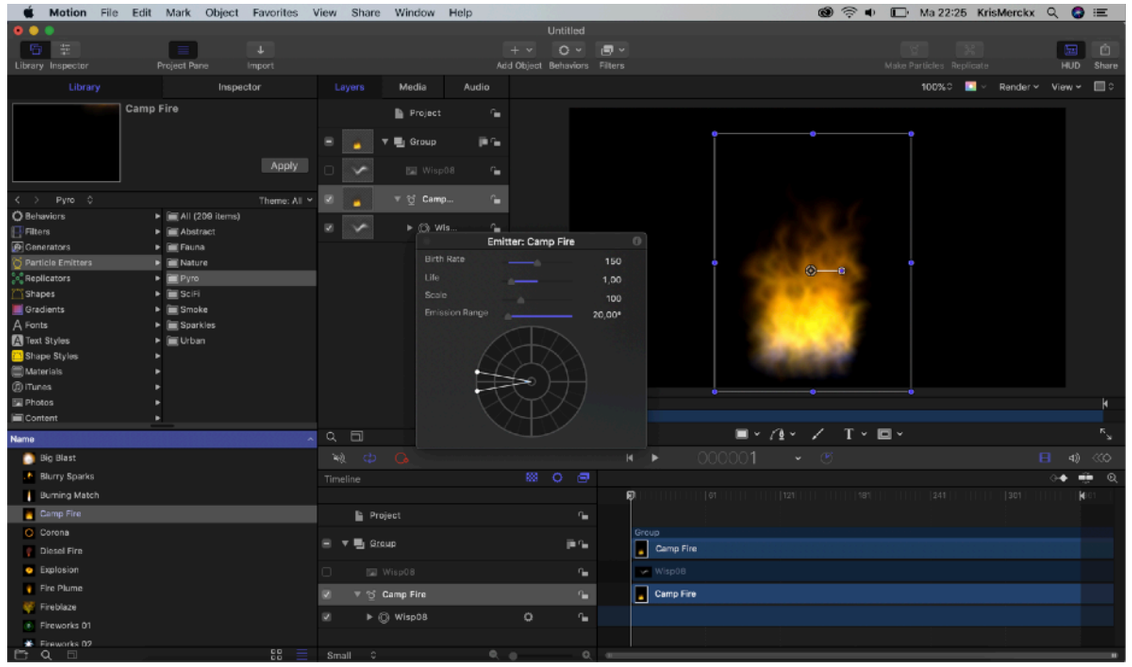
Particle systems

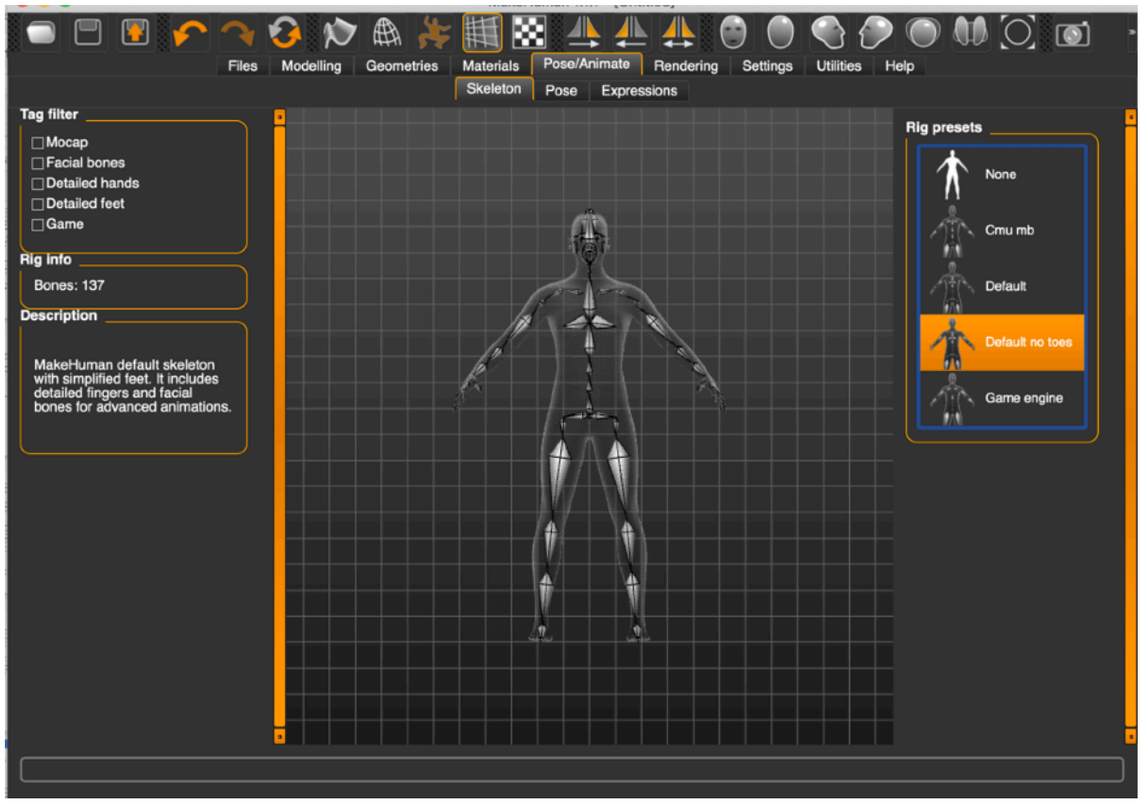
Soms bevat een echte situatie zoveel elementen dat het onnoemelijk veel werk zou vragen om ze onderdeel per onderdeel in 3D vorm te geven. **Particle systems** maken het mogelijk om in een mum van tijd effecten als *regen, sneeuw, ontploffingen* te genereren. Je moet de sneeuwvlokjes of druppels dus niet één per één modelleren.

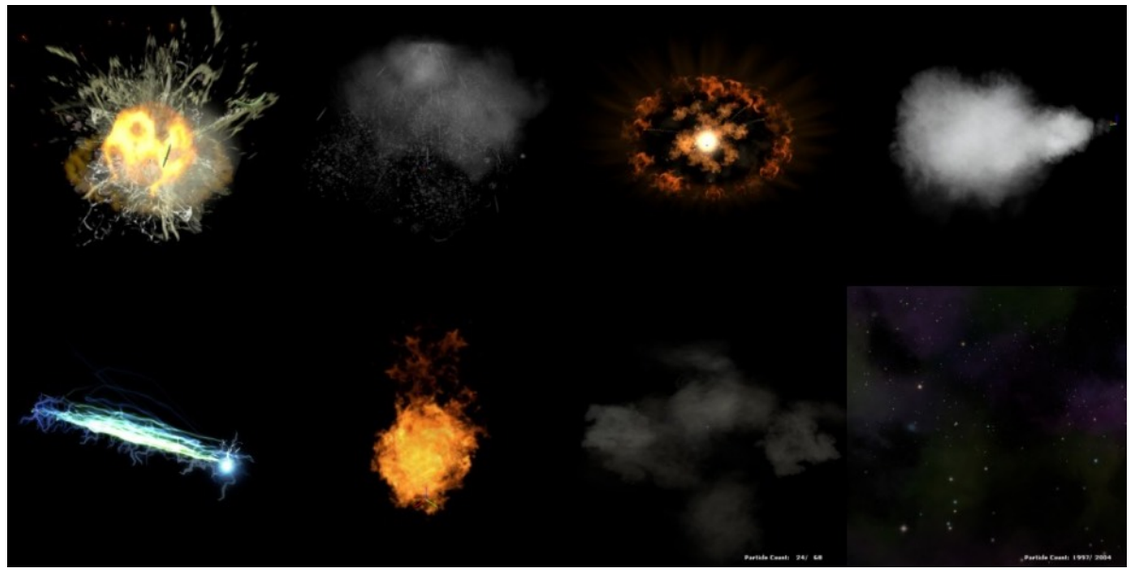
Voor Lord of the Rings gebruikte men op die manier het softwareprogramma Massive om veldslagen met honderden "orks" te genereren. De animatoren tekenden één "ork", de software zorgde voor de rest. Elke "ork" kreeg zijn eigen parameters en beweging, zonder dat de animatoren die ork per ork moesten instellen.

Afbeelding bovenaan: Virtueel landschap gebouwd met de software Terragen.
Afbeelding onderaan: boetseren in Sculptris.









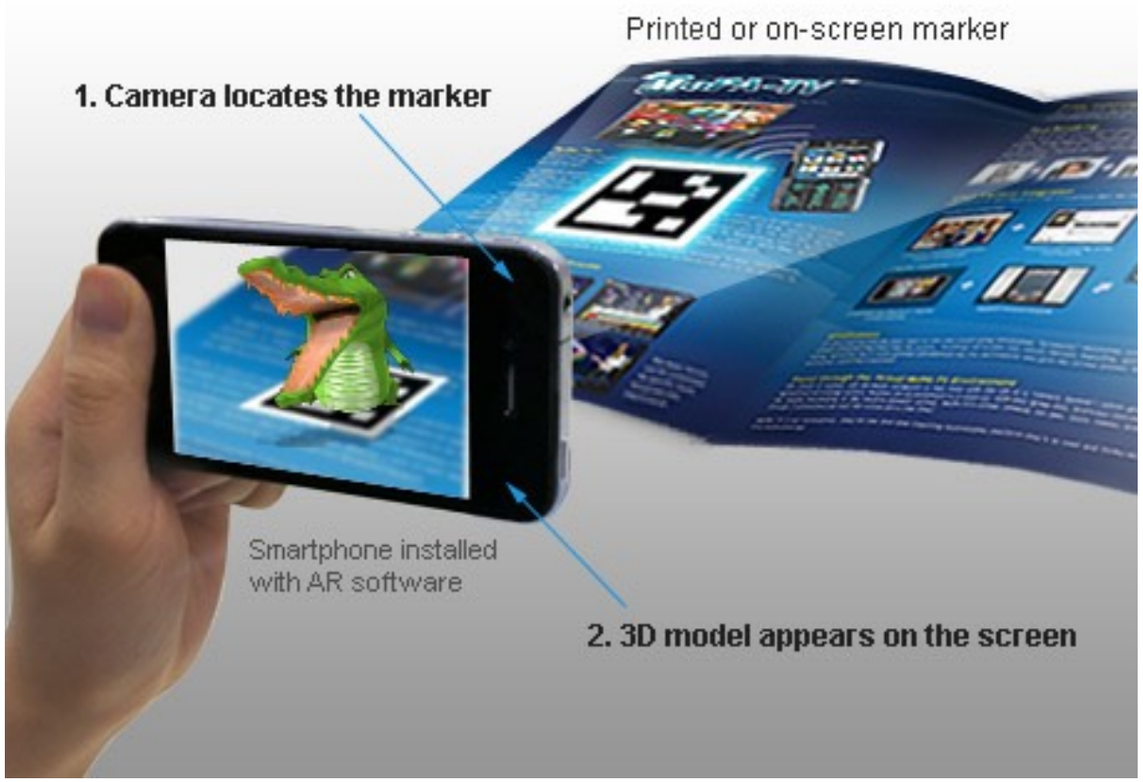
Bovenaan links: Menselijke 3D-figuren bouwen in MakeHuman
Onderaan links: Een virtueel animeerbaar skelet (bones) in MakeHuman.
Bovenaan rechts: Particle emitter in Apple Motion.
Onderaan rechts: Voorbeelden van particle systems.

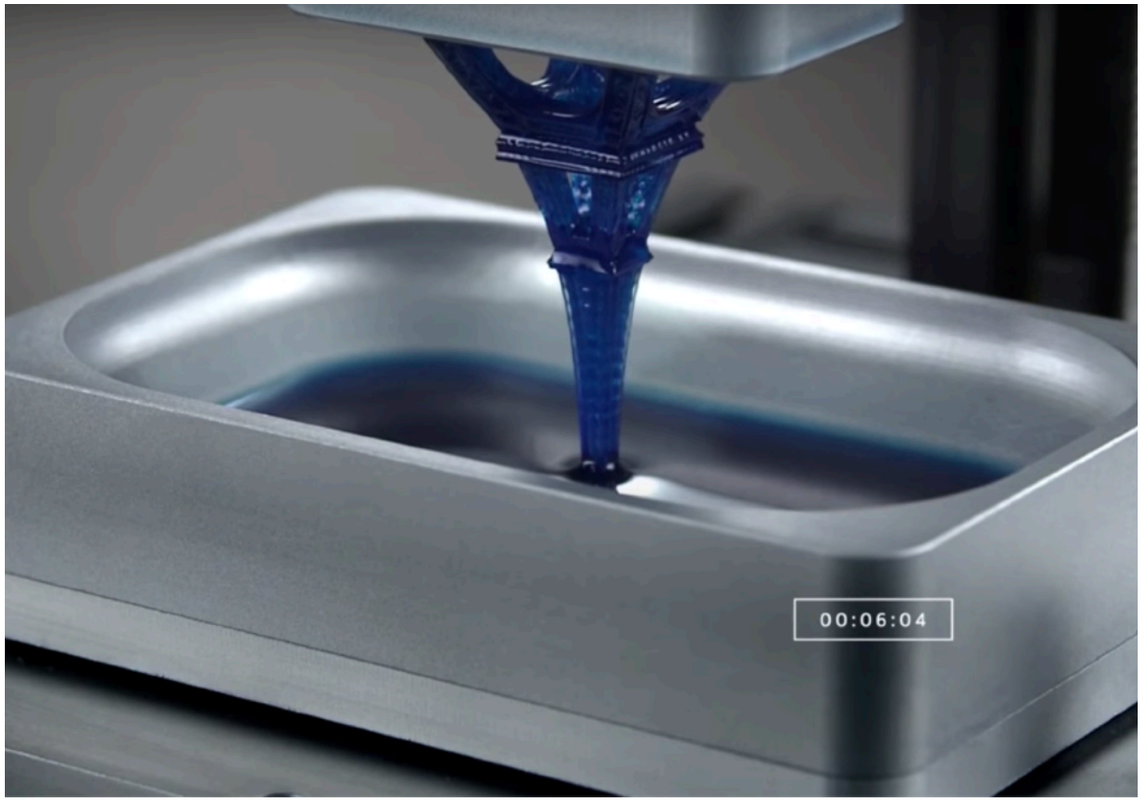
Interactieve 3D: games en virtual reality

Games op een Sony Playstation, Microsoft Xbox of een andere gameconsole worden gespeeld op een 2D-scherm, maar toch kunnen we in veel gevallen spreken van interactieve 3D. Als speler kruip je in de huid van een virtuele **avatar**. Je kan **interactief** door het spel lopen. Wat je te zien krijgt, hangt dus niet enkel af van wat de programmeurs je willen tonen, maar eveneens van je eigen interactie. Gameconsoles beschikken over krachtige grafische processoren omdat ze alle 3D-beelden in real time moeten "**renderen**" (onwikkelen).

Onder **rendering** verstaan we het proces om de 3D-modellen met hun texturen, belichting en schaduwen te ontwikkelen tot een realistisch beeld.

Bij **virtual reality** neem de gebruiker de 3D-scene direct waar. Hij/zij is volledig afgesloten van de echte wereld en kan interactief (bijvoorbeeld door het hoofd te bewegen) door een virtuele 3D-scene rondlopen en er eventueel interactief mee omgaan. Google Blocks is een voorbeeld van een stuk software waarmee je in VR 3D-modellen kan bouwen.





AR en 3D-mapping

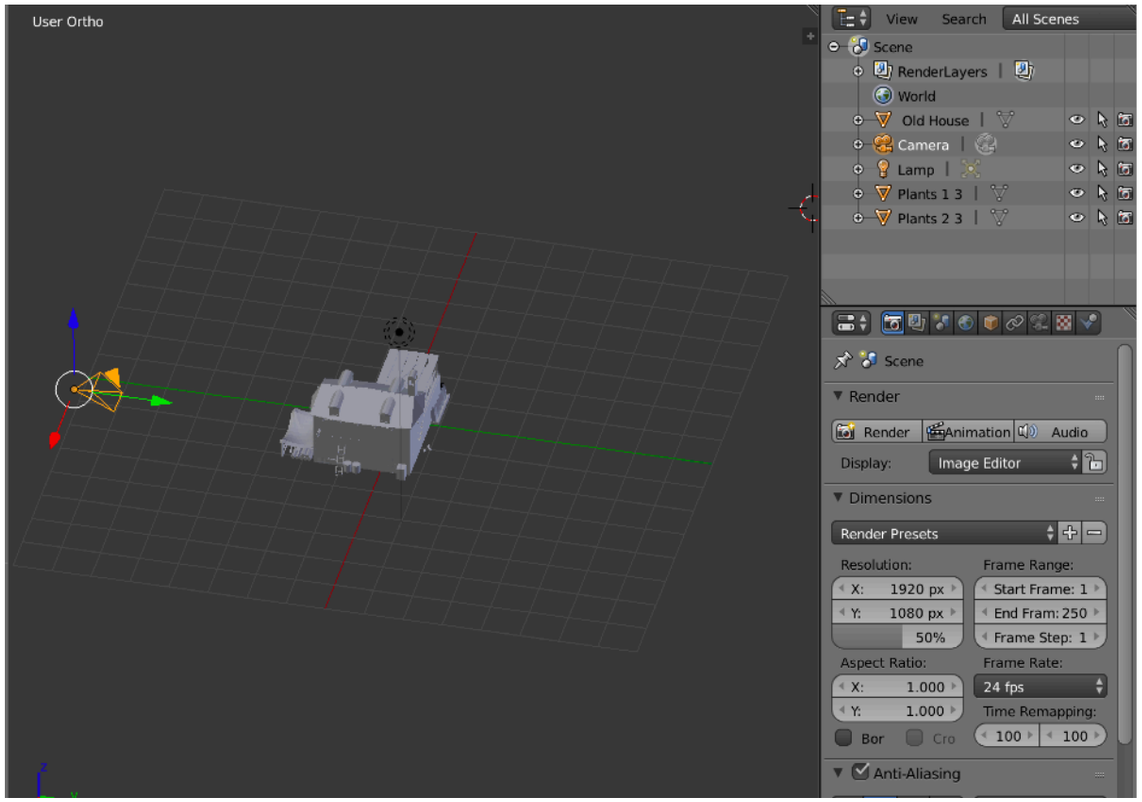
Bij **augmented reality** herkent de software de omgeving of bepaalde markeringen daarin. De software toont op een scherm niet enkel de echte omgeving, maar voegt er een laag digitale informatie aan toe. Zo kan een uitgeverij een soort **markering** in een boek opnemen. Wanneer de gebruiker de code met een smartphone filmt, tovert de software een 3D-object op de markering (**3D-mapping**).

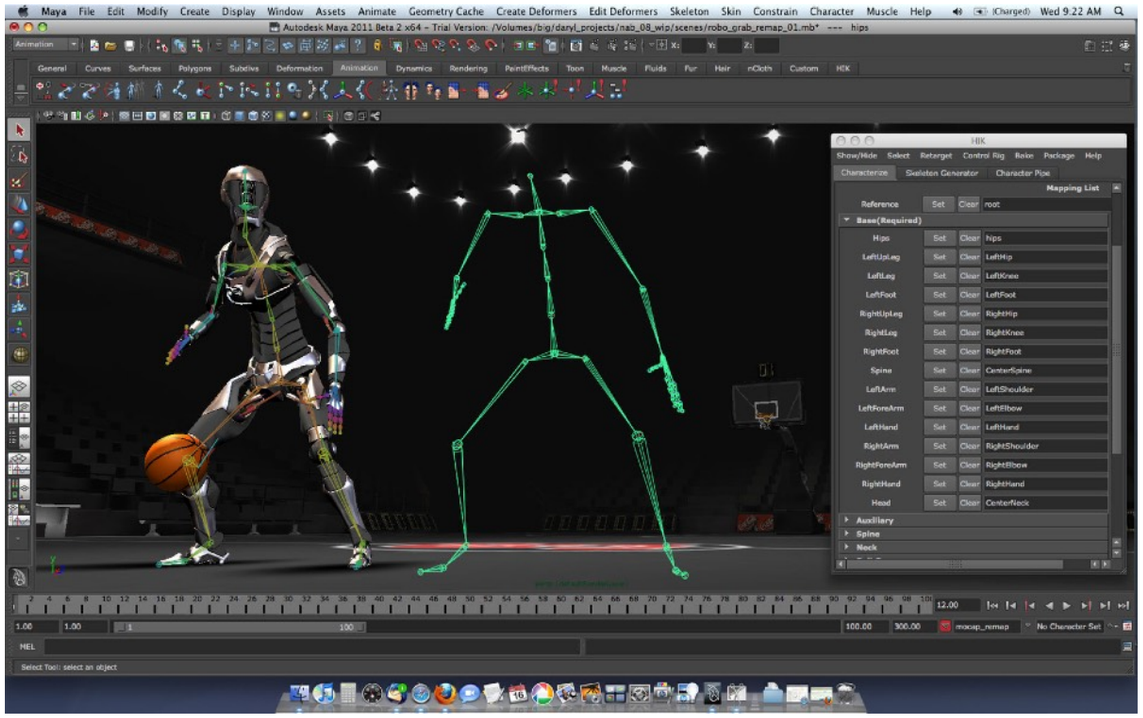
3D afdrukken

Een **3D-printer** drukt een 3D-model in verschillende fijne laagjes kunststof af, zodat we een reconstructie in reliëf kunnen maken van om het even welk echt of denkbeeldig object. 3D-printing zal gewone productietechnieken niet vervangen, maar zal zeker een belangrijke rol spelen in productieprocessen waar slechts kleine aantallen, of productie op maat, nodig zijn (vb. kunstgebitten).

VORIGE PAGINA
Bovenaan: VR-toepassing.
Onderaan: 3D-modeling in VR met Google Blocks.

DEZE PAGINA:
Bovenaan: AR-marker in boek en 3D-model via AR-app op smartphone.
Onderaan: CLIP (Continuous Liquid Interface Printing).





3D-software

Een commerciële licentie voor 3D-software kost doorgaans zeer veel geld. Toch bestaan er heel wat gratis proefversies. Het open source Blender is gratis, maar kent een hoge leercurve.

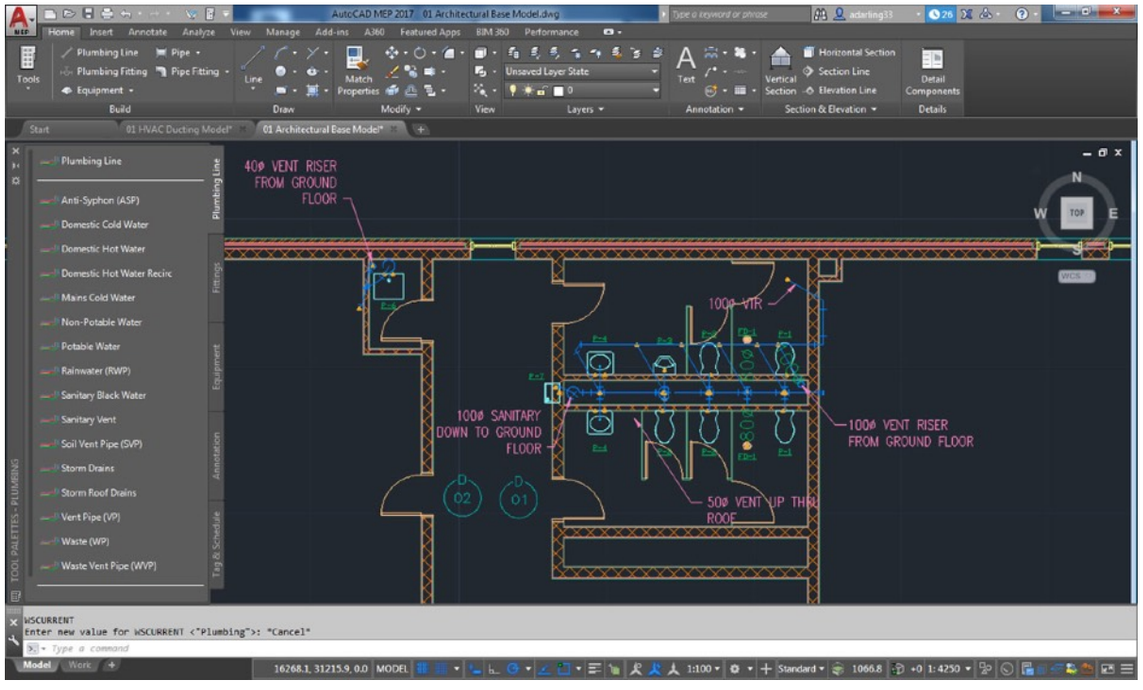
Het aantal gebruikers van 3D-modelleersoftware is eerder beperkt in vergelijking met grafische software als Adobe Photoshop. Programma's die uit de populariteit verdwijnen (omdat er andere of handiger tools ontwikkeld worden), houden vaak op te bestaan.

Een aantal programma's zijn echter uitgegroeid tot industriestandaarden.

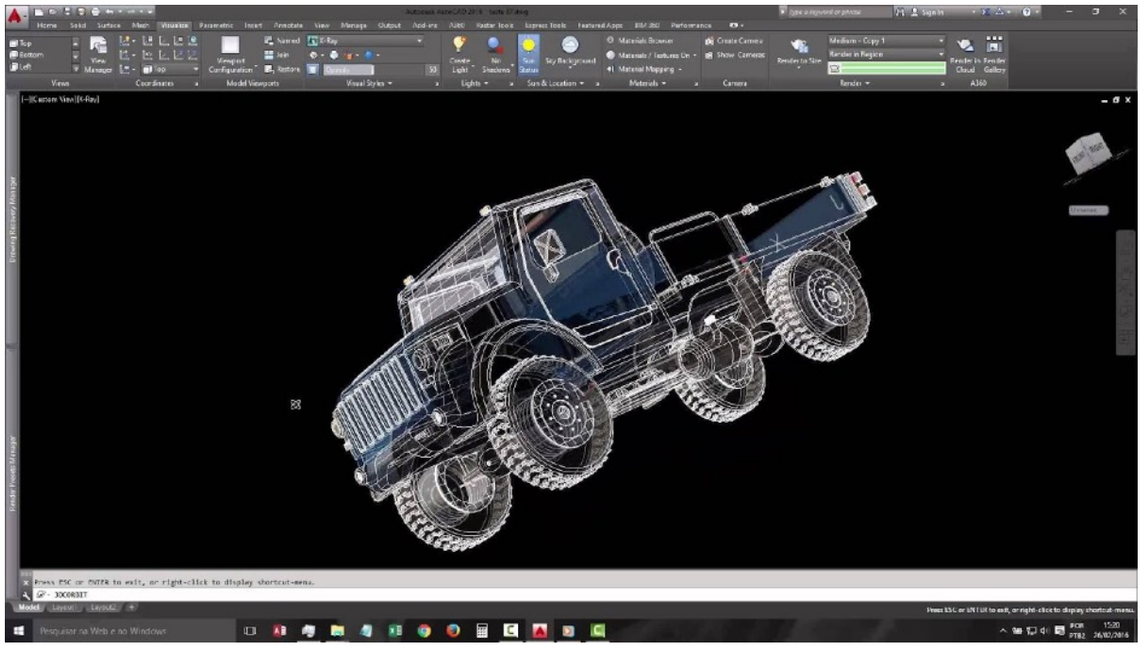
Onvolledige lijst 3D-software:

Software	OS
Maya	CenOS Linux, Mac OS X, Ret Hat Enterprise Linux, Windows
Houdini	Mac OS X, Windows
Cinema 4D	Mac OS X, Windows
Blender	Linux, Mac OS X, Windows
Modo	CenOS Linux, Mac OS X, Ret Hat Enterprise Linux, Windows
Audodesk 3ds Max	Windows
ZBrush	Mac OS X, Windows

Bovenaan: Virtuele camera in Blender.
Onderaan: 3D-model met bones in Maya.



Architectuurtekening (2D) in Autocad



3D-model van een vrachtwagen in Autocad

CAD

Met software voor **computer-aided design** ontwerp je constructies en apparaten. In de architectuur, bouwkunde, landmeetkunde, civiele techniek, stedenbouwkunde, technisch beheer, werktuigbouwkunde, elektrotechniek en elektronica, staalbouw, wegenaanleg, topografie, industrieel ontwerp, archeologie... maken ontwerpers gebruik van CAD-software.

Men maakt een onderscheid tussen 2D-, 2½D- en 3D-systemen. De 2D-systemen dienen hoofdzakelijk voor het maken van technische tekeningen. De 2½D-systemen zijn een uitbreiding met diepte voor CNC-gestuurde machines (CAM) en de 3D-systemen werken zoals de eerder genoemde 3D-modelleerpakketten.

3D-bestandsformaten

Bestandformaat (extensie)		Neutraal	Propriëtair
STL		x	
OBJ	ASCII	x	
OBJ	BINAIR		x
COLLADA		x	
3DS			x

GELUID EN MUZIEK

Nog over te zetten uit oude cursus. NOG DOEN

Audio…

Volume en toon

Geluid bewerken

Rechten (auteursrechten)

Synthesizer (voice)

Geluid profiel= > stem synthese

Samplingtechnieken

mono/stereo