Hello<DIGITAL/>World

<author>Kris Merckx</author>
<date>2015</date>

Deel 0:

Dit boek start met deel 0. In programmeertalen is het eerste element in een reeks objecten nooit gewoon 1, maar element 0. In menselijke taal zouden we dit deel de ***inleiding*** *noemen, maar toch is het belangrijk dat je net deze inleiding even leest. Op het einde van dit deel vind je een* ***afbeelding****, een soort cirkel, een voorstelling van het IPOS-model waarover je meteen meer leest.*
In dit boek probeer ik je digitale systemen beter te leren begrijpen. Je leert meer over hun werking, over hun mogelijkheden en beperkingen en over de uitdagingen voor de toekomst. Je zal interactief leren door te doen en te ervaren.

Definities:

Computersystemen: "Een computer is een apparaat waarmee gegevens volgens formele procedures (algoritmen) kunnen worden verwerkt. Meestal wordt met het woord computer een elektronisch, digitaal apparaat bedoeld, maar er bestaan ook mechanische en analoge computers. Daarnaast kan een computer in verschillende getalstelsels zoals het decimale (tientallige) of binaire (tweetallige) stelsel werken. De huidige computers werken allen in het binaire stelsel." (Bron: nl.wikipedia.org/wiki/Computer)

Open systeem: "Een open systeem is een systeem dat voortdurend interageert met zijn omgeving en daarbij al zijn inherente kenmerken behoudt." (Bron: nl.wikipedia.org/wiki/Open_systeem)

0.1 Het IPOS-model

Wat is een computersysteem eigenlijk en hoe werkt het? Nee, we gaan niet uitleggen wat een harde schijf is en waarvoor RAM-geheugen dient… dat weet je ondertussen ook al wel. Maar we gaan even een heel klein beetje de filosofische tour op. Als je alle computersystemen op een hoop gooit, wat hebben ze dan gemeenschappelijk?

*Meestal komt dan het zogenaamde **IPOS-model** om de hoek kijken: Input – Processing – Output – Storage.*

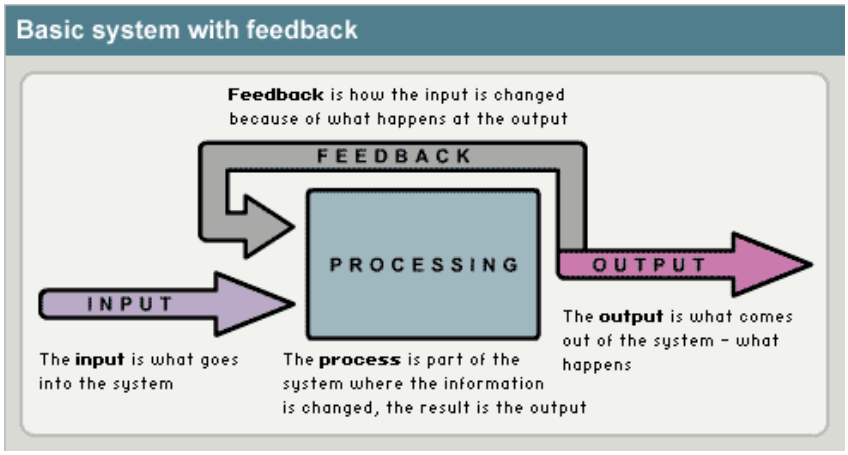
1. Onder **input** verstaan we alle soorten informatie die je in zo'n toestel propt: tekst, foto's, film, muziek… Hiervoor heb je een reeks invoerapparaten nodig: een toetsenbord, een webcam, een USB-kabel, een SD-kaart, een scanner, muis, microfoon of zelfs gewoon een andere computer. Een computersysteem kan ook zelf op zoek gaan naar invoerdata, zoals dat bij BIG DATA wel eens gebeurt.

2. Het toestel zal met die informatie iets gaan doen. In een aantal gevallen vind er een conversieslag plaats, een vertaling van analoge informatie naar digitale informatie (hier leer je nog over). Die verwerking of "**processing**" kan gebeuren tijdens de invoer of achteraf als de computergebruiker die input wil bewerken. Als je een tekst intikt in een tekstverwerker gebeurt de input en verwerking simultaan. Wanneer je een foto bewerkt in een beeldbewerkingsprogramma, dan gebeurt de verwerking meestal achteraf.

3. Om dit mogelijk te maken moet het computersysteem ook in staat zijn de ingevoerde data te bewaren (**storage**).

4. Een computersysteem zal de gebruiker **feedback** geven. Dat kan in realtime door bijvoorbeeld te zeggen dat er iets fout is gegaan, door een bevestiging te vragen enz. Een programma kan ook feedback geven aan zichzelf: als dit gebeurt, dan doe ik dat, in het andere geval… In zo'n geval spreken we van "open loop"-feedback. Wanneer de gebruiker op één of andere manier moet bevestigen of iets doen, dan spreken we van "closed loop"-feedback.

5. Maar de bedoeling is meestal dat de computer toont wat het resultaat van die bewerking is. In veel gevallen ziet de gebruiker het resultaat meteen op een scherm. Je ziet meteen het eindresultaat: what you see, is what you get (**WYSIWYG**). Ook nadien kan je de bewaarde en verwerkte data nog "bekijken" door het bestand te openen, af te drukken, te posten op een website enz. Inderdaad, dat is wat we met **output** bedoelen.

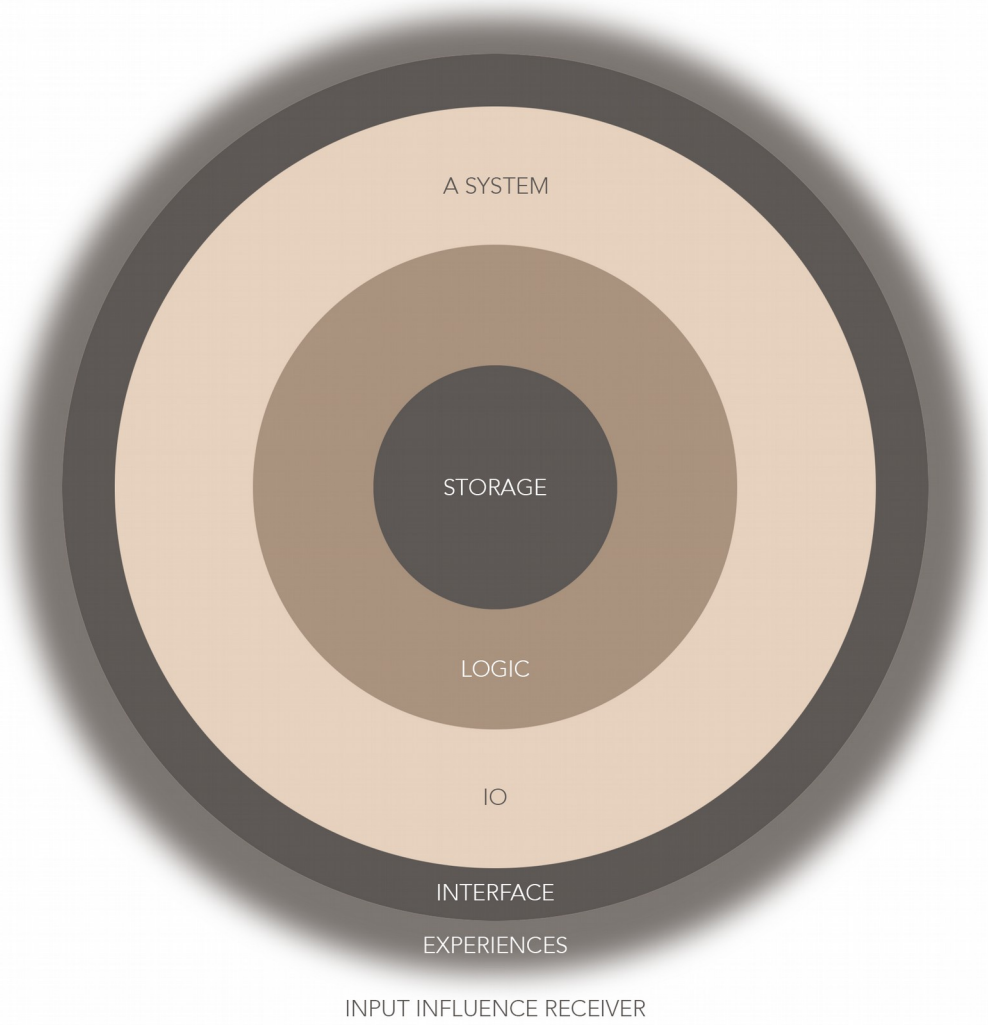


Bron: http://www.bbc.co.uk/schools/gcsebitesize/design/resistantmaterials/processystemsrev1.shtml

0.2 Hoe werkt *onze* "computer"?

In wezen is een computersysteem voor een deel gebouwd naar hoe we onze "eigen" werking als mens zien. Je ondergaat heel wat "invloeden" van je omgeving, van het weer tot die giftige sneer van een opmerking die de docent je gaf tijdens de les omdat je even niet aan het opletten was. Je hoort, ziet, proeft, voelt... en je registreert (INPUT via ondermeer "sensoren"), je selecteert waar je mee bezig wil zijn (je favoriete APPS) of moet zijn (standaard PROCESSEN), je verwerkt het (soms krijg je het niet verwerkt en CRASH je letterlijk). Je zegt soms dat het allemaal een beetje te veel is of dat je geen twee dingen tegelijk kan (te weinig RAM-geheugen, MULTITASKING). Je onthoudt ook zaken, dat kan gaan over pure herinneringen van gebeurtenissen in je leven, verplichte leerstof, ervaringen (kortom FILES en FOLDERS). Sommige dingen heb je niet geleerd, je doet ze automatisch zoals bijvoorbeeld ademen (BIOS, OPERATING SYSTEM). Maar je leert ook dingen doen: je kan fietsen, koffie maken, schrijven (PROGRAMMA's, SOFTWARE, APPS). En uiteindelijk geef ook jij feedback: je kan boos zijn om wat men je aandoet of net heel blij of gelukkig (FEEDBACK). Of je kan een alarminstallatie installeren omdat er werd ingebroken (FEEDBACK = nieuwe SOFTWARE). Je gaat naar school om nieuwe dingen te leren doen (APPS) om later werk te vinden (PROCESSING).

De afbeelding hieronder toont wat een systeem is en hoe we het in deze cursus stapsgewijs maar ook heel erg praktisch gaan behandelen.



Deel 1.
Digitale en elektronische verwerkingssystemen

Computers, laptops, tablets, smartphones, internet of things... Hoe maken we die digitale en elektronische wereld bevattelijk? Er is meer onder de zon dan de strijd tussen Microsoft en Apple of Android en iOS. In dit hoofdstuk leer je je weg te vinden in wat digitaal en elektronisch nu eigenlijk is. Hoe verhouden de diverse toestellen zich tot elkaar? Wat is het verschil tussen de boordcomputer in je auto en je laptop? Is je TV ook een computer als je ermee op internet kan?

1.1 Je weg vinden in het elektronische en digitale landschap

Je zal een stap voor zijn op de meerderheid van het grote publiek dat zulke toestellen wel massaal gebruikt, maar eigenlijk niet goed weet hoe die werken en nog veel minder hoe je zo'n toestellen zelf op maat kan bouwen. Wat leeft er allemaal in die "digitale wereld"? Wat is het verschil of de verhouding tussen analoog en digitaal, tussen elektronica en computers, tussen computers en microcontrollers, tussen embedded systemen en firmware, tussen hardware en software? Volg je nog? Ok, laat het ons een even allemaal op een rijtje zetten. *Opgelet: we staan niet stil bij klassieke computers of televisietoestellen of diverse "merken".*

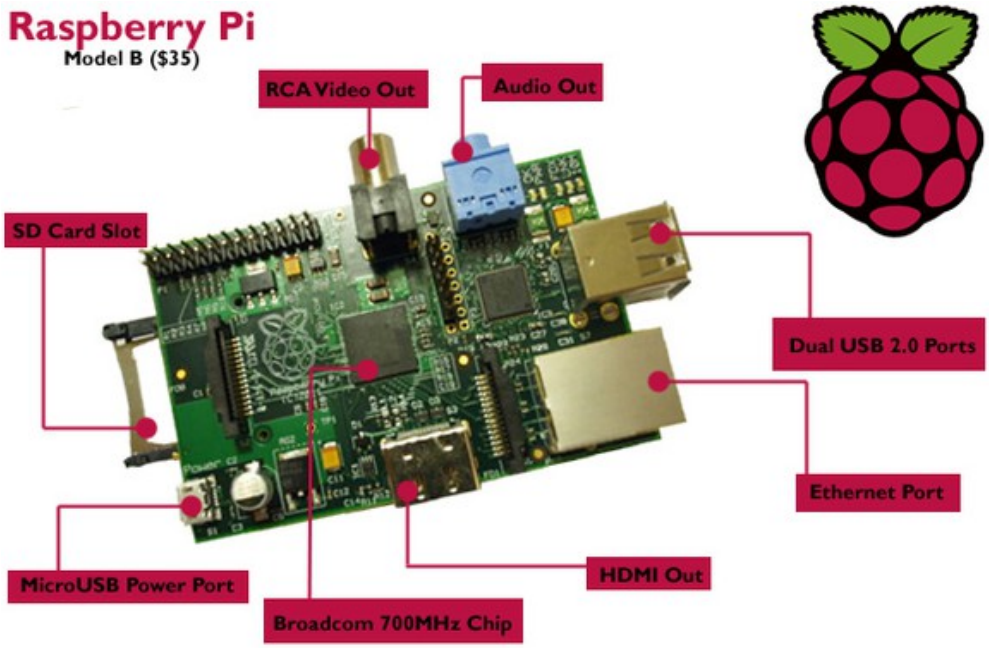
1.1.1 Hardware en software

Een **computer** (de **hardware**), of het nu gaat om een klassiek desktop-onding, 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...) en ook 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.*** Dat heb je op de schoolbanken geleerd. 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:

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

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...



Er bestaan ook **kruisbestuivingen** tussen beide. De immens populaire *Raspberry Pi* is daar een voorbeeld van.

1.1.2 Embedded systemen en firmware

Ongetwijfeld heb je op het vlak van "consumentenelektronica" al gehoord over "**embedded systemen**" en "**firmware**". Een embedded system (ingebed of geïntegreerd systeem) is heel vaak gebaseerd op een microcontroller (met een microprocessor en/of DSP of 'digital signal processor') of op een SoC (System on a chip). **Het vervult een bepaalde functie binnen een groter mechanisch of elektrisch systeem.** In een koelkast kan het de temperatuur regelen, in een wasmachine regelt het de functie van alle "programma-instellingen", in je auto zorgt het voor alle foutmeldingen die op je dashboard verschijnen, maar ook voor alle andere digitale informatie, voor het aansturen van bepaalde onderdelen van je motor enz.

Vroeger bestonden **regelsystemen** hoofdzakelijk uit mechanische en later elektronische onderdelen. Nu neemt de in een "microcontroller" geïntegreerde software veel van die taken over. Een microcontroller integreert hardware en software. Er zit geen "harde schijf" in waarop de software manueel kan geïnstalleerd worden, **vaak zit de software geïntegreerd in de hardware, als vrijwel niet te wijzigen programma's. Soms kan de ingesloten software geüpdated worden**, dan spreekt men van "**firmware**" .

1.1.3 Voordelen en nadelen

Zo'n integratie biedt 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 je koelkast of videomontagesoftware in je auto.

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. Even bekeken volgens et IPO-model: input = beeld van camera, processing = herkennen van ongewenste elementen, output en feedback = verwijderen van ongewenst element.

Vaak worden embedded systemen met één bepaald doel voor ogen ontwikkeld, zijn ze heel "gespecialiseerd", maar je kan ook herbruikbare microcontrollers ontwikkelen... en dan zitten we bij de **Arduino** aan het juiste

adres als "startersmodel".

Een eigenhandig gebouwde smartphone op basis van een Arduino-microcontroller.

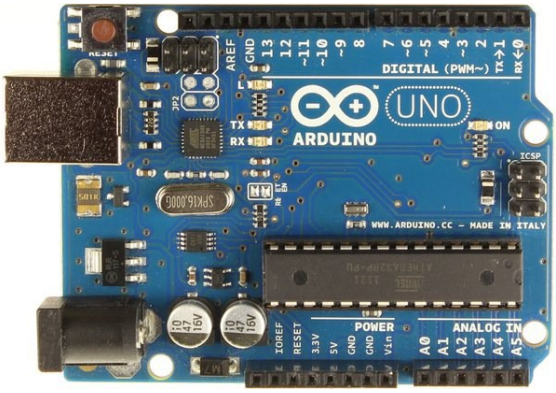
1.1.4 Embedded systeem

Een "embedded systeem" bevat een **sensorgedeelte** dat informatie uit de werkelijkheid kan registreren. In een tweede fase kan het die informatie **digitaliseren**. Die digitale informatie moet verwerkt worden door de **"microprocessor" of de firmware**. In een laatste fase zal een "**actuator**" bepaalde onderdelen aansturen of zorgen voor zichtbare of hoorbare **output**.

1.1.5 SoC's en microcontrollers

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.

Oeps, waar ligt dan de grens tussen een "microcontroller" en een "SoC"? De term "System on a chip" is in dat geval niet de beste keuze geweest. Immers, een "microcontroller" is vaak pas echt gebaseerd op één enkele chip, waardoor het beschikbare RAM-geheugen vaak onder de 100 kb blijft. Een SoC bevat vaak een veel snellere processor met meer werkgeheugen waardoor het in staat is om besturingssystemen zoals Linux te draaien. Het is dus niet altijd echt één chip, want een SoC maakt vaak gebruik van externe geheugenchips (SD-kaartje, RAM...). Een modern besturingssysteem, hoe compact ook, heeft wel wat meer nodig dan 100 kb. **Het begrip SoC verwijst dus eerder naar een zeer sterke integratie van de chips, waardoor kleinere systemen mogelijk worden.**



Een Arduino-microcontroller.

1.2 Digitale "dingen"

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

Kortom, embedded systemen, SoC's en microcontrollers zijn niet meer weg te denken uit de wereld van vandaag en morgen. *Be prepared!* Wees voorbereid, want wie het kent, kan er zijn voordeel uit halen en zich behoeden voor al te grote 'schendingen' van zijn vrijheid en privacy. Hoe meer je erover weet, hoe beter je er mee kan omgaan, hoe beter je het systeem kan "hacken" (=uit elkaar halen en begrijpen).

1.3 Analoog naar digitaal

De fysieke wereld waarin wij leven gedraagt zich **analoog**. Natuurkundige fenomenen zoals licht, geluid, temperatuur ... kunnen voortdurend wijzigen. Een **elektronisch apparaat zet signalen uit de omgeving (geluid, licht, warmte ...) om in elektrische signalen (spanningen, stromen ...)**. Net zoals een foto-elektrische cel licht omzet in een elektrisch signaal, zet een microfoon drukgolven in de lucht om in een veranderlijke elektrische spanning.

1.3.1 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[1] 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.

1.3.2 Actuatoren

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*

1. Een **membraan** of **vlies** is een dunne, vlakke structuur die twee ruimtes van elkaar scheidt. (…) In de geluidstechniek komen membranen veel voor, met name in: a. membranofonen, die doorgaans bestaan uit een 'ketel' met een gespannen vel over de bovenkant, waarop geslagen of gewreven wordt. De holle vorm - waarvan soms ook onderkant door een vel is bespannen - werkt als klankkast en versterkt het geluid van het membraan dat wordt beroerd. De toonhoogte hangt van allerlei factoren af, zoals de spanning in het membraan en het materiaal waaruit dit is gemaakt, de vorm van de ketel heeft veel invloed op het timbre. Voorbeelden zijn: de trommel, de pauk, de bongo, de conga, de djembé, de darbuka, de batá-drum en de sabar
b. een microfoon om de geluidstrillingen op te vangen en om te zetten in elektriciteit
c. luidsprekers om elektromagnetische of elektrostatische kracht in beweging om te zetten als geluid
Deze membranen mogen geen medium (lucht) doorlaten omdat dat het rendement van de geluidsopbrengst nadelig zou beïnvloeden.
… (Bron: nl.wikipedia.org/wiki/membraan)

beweging, een lamp en een beeldscherm zetten elektrische energie om in licht enzovoort.

Naast de invoer via sensoren en de uitvoer via actuatoren voert de elektronica zelf een aantal **berekeningen**, algoritmes of programma's uit op de binnenkomende signalen. Elektronica kan naar analogie met de werkelijkheid het elektrische signaal continue bewaren, zoals dit bijvoorbeeld met geluid op een magneetband gebeurt. Toch kunnen hier wat fouten of afwijkingen optreden. Als je destijds een audiocassette te lang op een warm dashboard liet liggen, merkte je het aan de geluidskwaliteit!

Een klassieke audiocassette bewaarde geluid analoog op een magneetband.

1.3.3 Discretisatie

Elektronisch is nog wat anders dan digitaal, ook al hoor je de termen weleens door elkaar gebruiken. *Het sleutelwoord bij digitaal is '**discretisatie**' in ruimte en tijd.* Wanneer we analoge informatie digitaliseren, plaatsen we een denkbeeldig raster over de fysieke wereld en *meten we de binnenkomende signalen enkel op bepaalde tijdstippen, meestal op vaste intervallen*. Dit gebeurt doorgaans zeer snel, want anders zouden we veel 'informatie' missen. Discretisatie in ruimte en tijd heet in het vakjargon ook wel bemonstering, maar is vooral bekend onder de Engelse benaming 'sampling'. Een sample of een monster van een beeld noemen we een 'picture element' of 'pixel'. Een digitaal systeem meet ook de signaalniveaus (discretisatie van de signaalwaarden), zoals de lichtintensiteit van een pixel.

1.3.4 Afspraken

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.

1.3.5 Binair coderen, ruis en foutonderdrukking

Simpel gezegd werkt een elektronisch of digitaal toestel op basis van elektriciteit en die kent ruwweg maar twee toestanden: aan (1) of uit (0). Het lijkt dus op het eerste gezicht niet zo eenvoudig om getallen hoger dan de waarde 1 te meten of te bewaren in een digitale omgeving. Daarom maken digitale systemen gebruik van het **binaire stelsel**, een tweetallig getalsysteem dat alle getallen weergeeft met de symbolen 0 en 1.

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:

```
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.

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 altijd met binaire en niet met decimale getallen weergegeven. In het binaire stelsel wordt elk getal voorgesteld als een combinatie van nullen en enen. 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.

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 ...

1.3.6 Van digitaal naar analoog

Pulsbreedtemodulatie (PWM of pulse width modulation) is het begrip als het gaat om het omzetten van digitale signalen naar analoge signalen. Cool, echt een begrip om mee uit te pakken aan de toog van je stamcafé, maar je kan best toch zorgen dat je dan tenminste weet waarover je het hebt. Zeker als die onbekende man op de hoek van de toog een ingenieur blijkt te zijn. Ben je er klaar voor?

Stel dat je dit boek nu niet zou lezen, maar dat je naar de audioversie ervan zou luisteren, dan zou het geluid van mijn opgenomen stem de lucht laten trillen. De luchtdeeltjes zouden als een voortdurend wijzigende golf je oren bereiken, met steeds wisselende pieken en dalen: een analoog geluidssignaal, naar "analogie" met de werkelijkheid. Nochtans werkt die luispreker op basis van elektrische signalen die nu eens uit en dan weer aan kunnen staan. Ofwel stuur je een pulsje stroom, ofwel niet, want elektriciteit beweegt nu eenmaal niet zoals een golf zeewater. Het is het een of het ander bij elektriciteit. De stroom staat aan of uit, 1 of 0. Hoe kan je dan met elektrische stroom een geluidsgolf genereren, een motor zachter laten draaien, een LED-licht dimmen?

Je kent het antwoord al, met PWM! Maar hoe gaat dit nu in zijn werk? Met PWM moduleren we de stroom. We schakelen bliksemsnel tussen 5V en 0V. Stel dat je de voltage de helft van de tijd instelt op 5V en de helft van de tijd op 0V dan krijg je een gemiddelde stroom van 5V of een "duty cycle" van 50%. Laat je de stroom 10% van de tijd op 5V staan, dan krijg je gemiddeld 0.5V. Als je op deze manier een motor aanstuurt, zal de motor langzamer draaien bij een lagere duty cycle. De motor krijgt dan wel zijn maximale spanning, maar niet continu. M.a.w. als je de schakelaar continu aan laat staan, zal je motor constant snel draaien, je LED-licht continu "fel" branden. Als je snel aan en uit pulst, zal de motor zachter beginnen draaien en je licht dimmen.

1.3.7 Programmeerbare automaten zijn zeer oud (randinformatie)

<randinformatie>

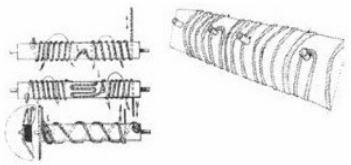
Het binaire talstelsel werd "bedacht" door de wiskundige en filosoof Gottfried Wilhelm Leibniz (1646– 1716), een tijdgenoot van het onovertroffen genie Isaac Newton (1643–1727). In zijn tijd zag niemand het nut in van zo'n binair talstelsel en hij zelf gebruikte het vermoedelijk enkel in filosofische discussies over godsdienst: de 1 zou daarbij staan voor het bestaan van God en de 0 voor de afwezigheid van God. Andere vondsten van Leibniz, zoals het 'is gelijk aan'-teken (=) of de dubbele punt bij delingen (:), raakten wel ingeburgerd. Het binaire getalsysteem werd in de eerste helft van de 20e eeuw dankbaar opgevist voor gebruik in de eerste digitale computers.

Ook al was het binaire talstelsel nog niet uitgevonden, toch bestond het "digitaal" programmeren van machines al veel langer. Heron van Alexandrië bouwde tweeduizend jaar geleden automatische theaters met gewichten, touwen, assen en pinnen. Aan het ene uiteinde van het touw bevestigde hij een gewicht, dat hij boven op een met tarwekorrels gevulde cilinder plaatste. Door een sleufje onder in de cilinder open te trekken, vloeiden de korrels weg. Hierdoor begon het gewicht met enige vertraging te zakken. Aan de andere kant hingen twee touwen. Elke touw was rond een afzonderlijke as (de linker en de rechter vooras) gewikkeld. In beide assen waren op gelijkmatige afstanden gaten geboord. Het zakkende gewicht trok niet alleen aan de touwen, maar liet de beide assen ook draaien. De machine kon eenvoudig geprogrammeerd worden door in de gaten pinnen te plaatsen. Als je een touw rond een bepaalde pin liet teruglopen, kon je een van beide assen op elk gewenst moment in de andere richting laten roteren. Net zoals in moderne programmeertalen slaagde Heron er ook in om een 'timer'-functie in te bouwen. Hiervoor plakte hij met was een stuk van het touw vast aan de as. Het zakkende gewicht trok het was stilaan los. Als het touw eenmaal was losgekomen, begon de as weer te roteren.

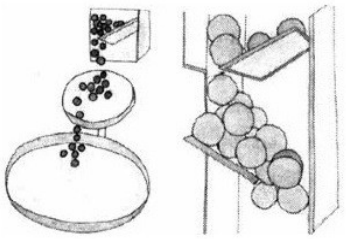
Je zou de instructies voor Herons robot kunnen uitschrijven in de code voor Lego® Mindstorms- robots:

```
task main(){
      OnFwd(OUT_A,75);
      OnFwd(OUT_B,100);
      Wait(3000);
}
```

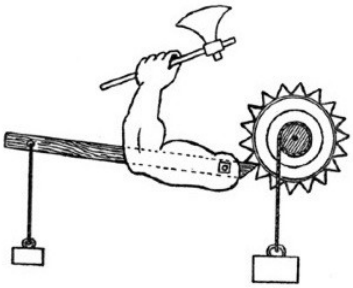
De OnFwd-instructie laat de motor naar keuze (A of B) vooruitbewegen. De Wait-instructie laat het toestel voor een bepaalde duur (milliseconden) halt houden. Uiteraard schreef Heron geen programmeerinstructies. Ze waren vastgelegd in zijn machine, maar op zo'n manier dat je het toestel kon herprogrammeren door de pinnen te verplaatsen. Het programma werd 'opgeslagen' in de assen met wat we binaire instructies zouden kunnen noemen. Een pin staat voor een 1, een gat zonder pin voor een 0 (of omgekeerd). Het automatische theater van Heron reed volkomen zelfstandig het podium van het theater op. Daar stopte het en toonde een toneelstuk van mechanische poppen, die eveneens door een mechanisme van pinnen, gewichten, touwen, assen, tandwielen en hefbomen werden aangedreven. Het theater was voorzien van decorwissels en geluidseffecten. Door op een bepaald moment een sleuf open te trekken, vielen loden ballen op een trom, wat het geluid van donder simuleerde.



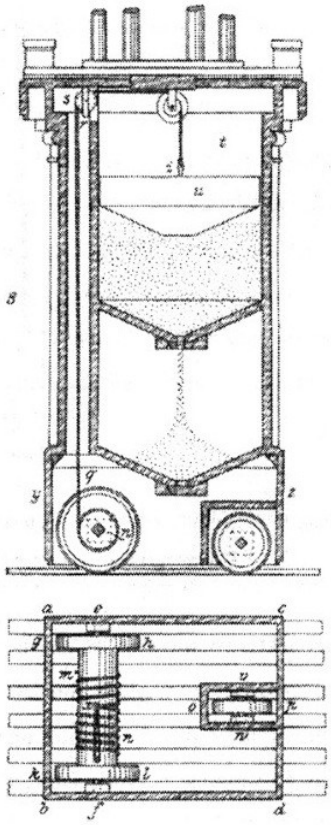
De touwen en de pinnen bepalen de beweging van de assen. (Bron: Manuela Rausch, *Die Programmgesteuerten Automaten Des Heron Von Alexandria*, 2012)



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



Bewegende onderdelen in het theater van Heron (Bron: Chapuis A. & Gelis E., *Le monde des automates*)

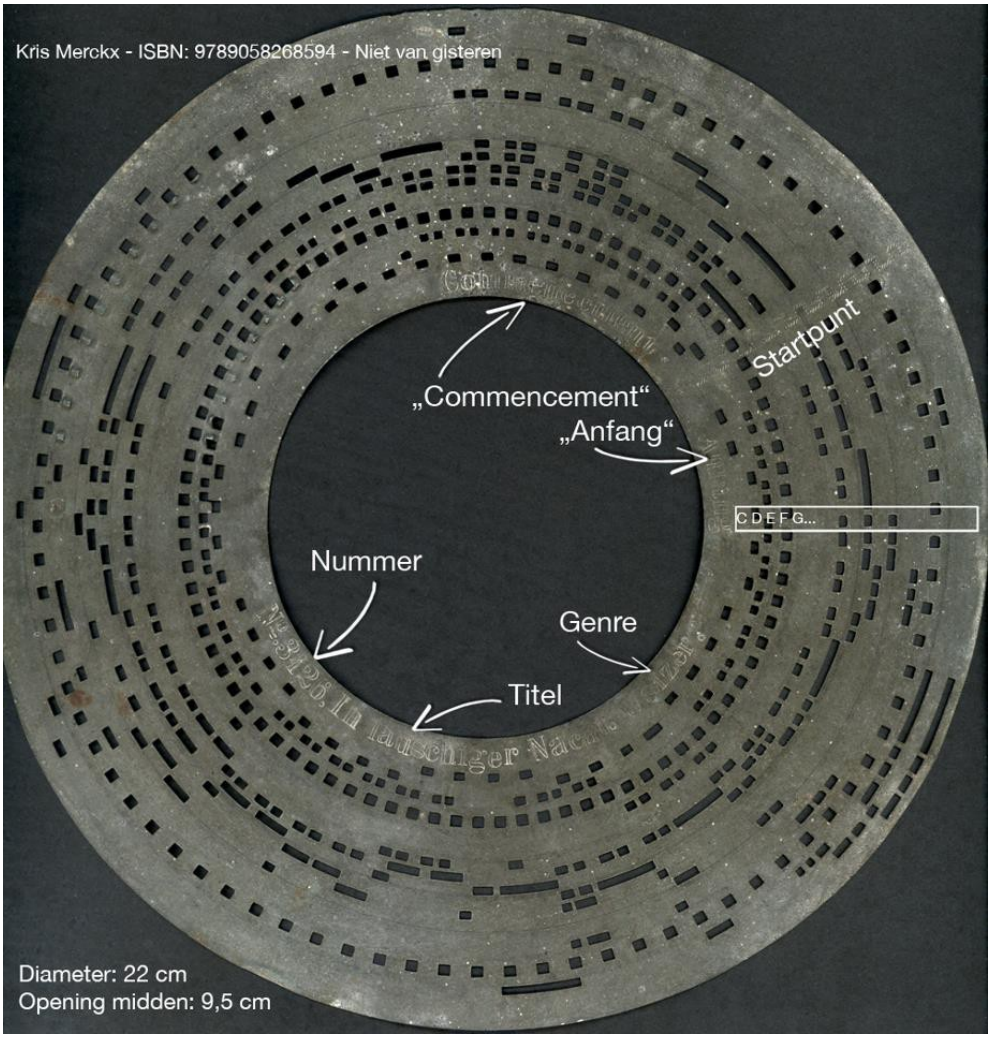


(Bron: Beck T., *Herons des älteren Automatentheater*, Jahrbuch des Vereines deutscher Ingenieure, 1909)

De werking van het automatisch theater van Heron.

Wellicht bestond de techniek voor het programmeren van toestellen al langer en heeft Heron het niet zelf bedacht. De techniek om toestellen te programmeren dook ook op bij Islamgeleerden, in middeleeuwse klokken, bij Leonardo da Vinci. In de 18e eeuw paste men de techniek enigszins aan door de cilinders met pinnen te vervangen door kaarten met gaten (ponskaarten). Men begon ze in te zetten voor het automatisch weven van kleren in weefgetouwen (Jacquard). De techniek van de ponskaart als middel om gegevens of programma-instructies te bewaren, bleef in gebruik als opslagmedium voor computers tot in de jaren 1980.

Een ponskaart voor de opslag van geprogrammeerde muziek (19e eeuw).

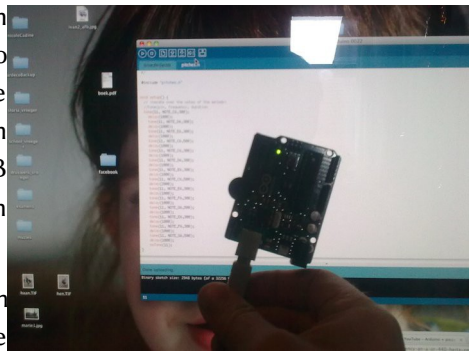


</randinformatie>

1.4 Microcontrollers en SoC's voor doe-het-zelvers

1.4.1 Arduino

De Arduino is niet enkel een microcontroller. Het is een speciaal ontworpen bord voor het programmeren en het maken van "prototypes" met Atmel-microcontrollers. Als je er al eentje hebt aangeschaft, zal je ondertussen wel weten dat het goedkoop is. Voor een kleine 25 euro haal je een Arduino UNO in huis. Bovendien is het aardig makkelijk om mee te werken. Je sluit het via USB aan op je computer. Arduino-programma's schrijf je met een gratis stuk software. (http://arduino.cc/en/Main/Software).



De Arduino krijgt stroom via een USB-verbinding, maar kan ook los van de computer functioneren via een netadapter of batterijen. De Arduino meldt zich bij uw computer aan als een "virtuele seriële poort". Als je code schrijft, kan je ze met een eenvoudige klik op de knop uploaden naar de Arduino. De standaard Arduino beschikt over 32 KB flashgeheugen om je code te bewaren. Dat is niet veel, maar ruim voldoende voor de meeste projecten.

Een Arduino telt 13 digitale en 6 analoge pins om externe hardware en sensoren aan te sluiten. Voor wie wat meer thuis is in de elektronicawereld: het bord heeft ook een ICSP-connector waarmee je de Arduino rechtstreeks als een serieel toestel kan aanspreken. Via deze poort kan je de Arduino "herstarten" (bootload) wanneer de chip niet meer met je computer wil "spreken".

Afhankelijk van het project dat je wilt bouwen, kan je extra bordjes, sensoren of uitvoerapparaten op de Arduino aansluiten of in de pins "klikken". Wanneer je een gewenst onderdeel aankoopt, vind je op de website van de fabrikant over het algemeen het schema voor het bouwen van de hardware-applicatie en de benodigde code. Controleer dit vooraf zodat je niet voor ongewenste verrassingen komt te staan. Houd er ook rekening mee dat je voor sommige modules met een soldeerbout overweg moet kunnen.

Sensoren	Gyroscoop, accelerometer, ultrasone sensor, GPS, IR-afstandssensor, verstelbare IR-sensor, bewegingssensor
Shields	Wifi, Ethernet, Motor, Color LCD, AM&FMontvanger, VoiceBox (voice en sound synthesizer), Musical Intrument, Audio Player, Joystick, Xbee
Modules en adapters	Voide recognition, radiocommunicatie, Xbee, Bluetooth, Wiichunkadapter, SD lezer, Power over ethernetmodule, GSMmodule

1.4.2 Wiring

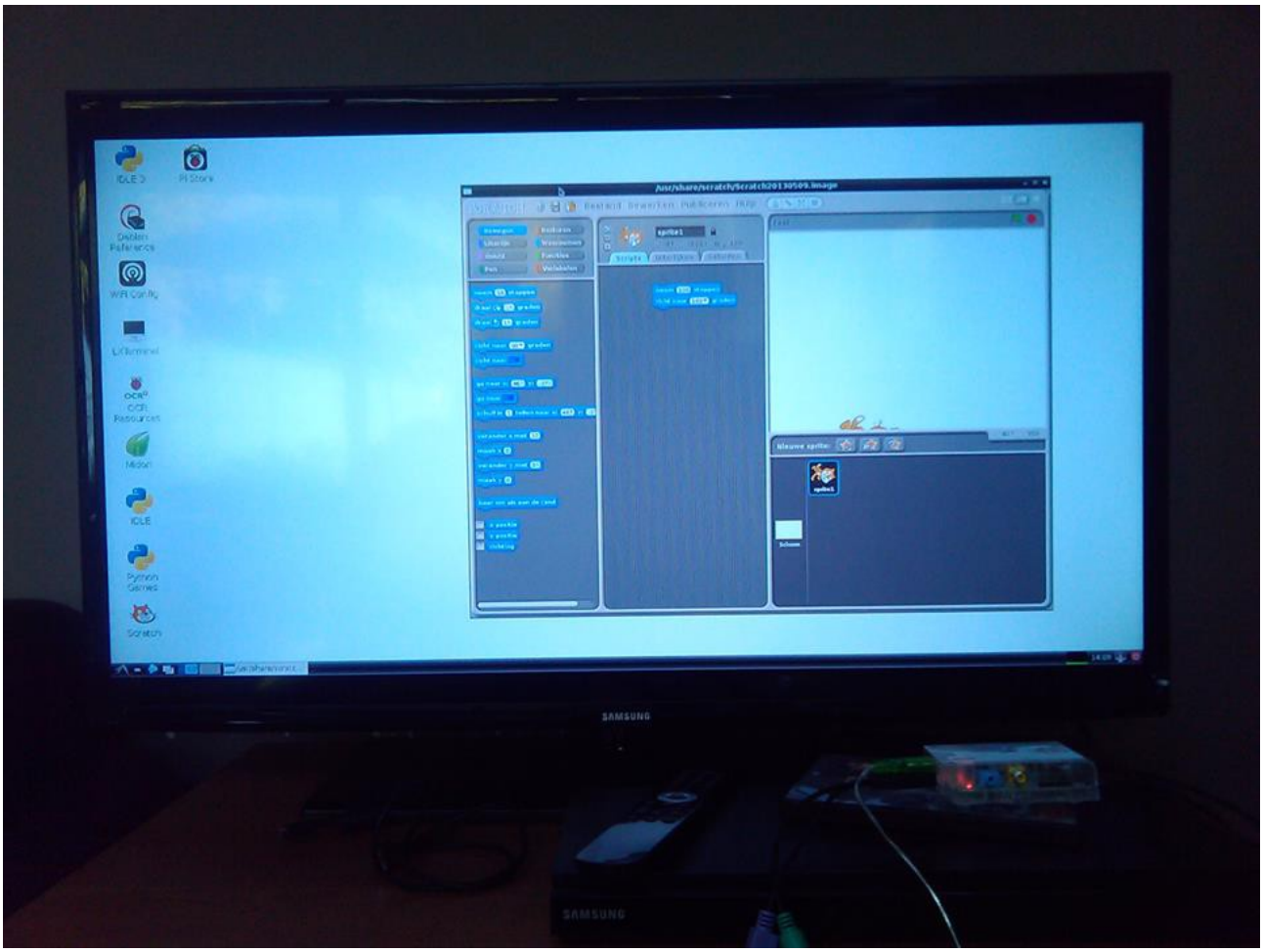
Wiring is de naam van een hardwareproject, een soort broertje of zusje voor Arduino. Meer informatie vind je op http://wiring.org.co/ . Net zoals Arduino is Wiring een open-source **elektronica prototyping platform**. Wat open source is, leer je in een later hoofdstuk.

Je kan er op heel eenvoudige manier een "prototype" van een programmeerbare elektronische schakeling mee bouwen.

1.4.3 Raspberry Pi

De Raspberry Pi is een minicomputer (ook wel een singleboardcomputer genoemd) met een ARM-processor die ook dienst kan doen als microcontroller voor kleine elektronicaprojecten. Vermits het een gewone computer is, kan je er ook randapparatuur zoals een muis en toetsenbord (via USB), een microfoon, luidspreker of beeldscherm (TV via HDMI) op aansluiten. Een hele rits Linux-distributies (maar ook bijvoorbeeld Risc OS en Plan 9) zijn geschikt om op de Pi geïnstalleerd te worden. Het toestel kan ingezet worden als fileserver, NAS, mailserver... Voor een kleine 50 euro koop je een Raspberry Pi wat mede heeft bijgedragen tot zijn gigantische succes.

Een Raspberry Pi met een Linuxbesturingssysteem via HDMI aangesloten op een televisie.



1.4.4 Intel Galileo

Processorfabrikant Intel keek al snel met groeiend ongenoegen naar het succes van de op een ARM-processor gebouwde Raspberry Pi. Daarom lanceerde het bedrijf zijn eigen minicomputer aan onder de naam Intel Galileo. Het toestel draait op de energiezuinige Quark x1000-SoC.

Omdat de Raspberry Pi ook kan gebruikt worden voor elektronicaprojecten zag Intel de Arduino als een nuttige bongenoot in de zijn "oorlog" tegen het ARM-geweld. De Galileo is daarom niet alleen compatibel met zijn "Pentium Instruction Set Architecture", maar ook met de Arduino-bibliotheken en IDE.

1.4.5 Little bits

Wie toch zijn eigen elektronica wil "programmeren" en samenstellen zonder één letter code te schrijven of met kabels of soldeerbouten aan de slag te moeten gaan, moet zeker eens kijken naar Little bits. Met de eenvoud van Lego bouw je met Little bits coole elektronicaprojecten. Je klikt de diverse onderdelen poepsimpel via kleine magneetjes aan elkaar en klaar is kees.

1.5 Oefeningen

Audacity: Tijdens de les digitaliseren we audio met het open sourceprogramma Audacity (www.audacityteam.org). We bekijken hoe het audiosignaal er in digitale vorm uitziet.

Adobe Photoshop / GIMP: We zetten een foto om in zwart-wit en verlagen het aantal pixels drastisch. We bekijken hoe discretisatie aan het werk is gegaan.

http://www.ardeco.be/sirk/screen.html: Pixelweergave met een raster. Hoe stuur je boodschappen door in binair formaat?

Deel 2: 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. Een computer zet alle informatie om in enen en nullen, zonder echt te begrijpen waar de inhoud over gaat. 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.
In dit deel bekijken we hoe een mens met een computer communiceert. Welke mogelijkheden tot interactiviteit bestaan er op dit moment en wat wordt misschien ooit mogelijk? Hoe ontwerp je een gebruiksvriendelijke en aantrekkelijke interface? Hoe vertellen mensen aan computers welke taken ze moeten verrichten?

2.1 Digitale systemen in de omgang: interfaces

2.1.1 Interactiviteit

Niet alle computers staan rechtstreeks in wisselwerking met de mens. Volgens het IPOS-model definiëren we een computer als een machine die inkomende signalen verwerkt volgens een reeks instructies en op één of andere manier output doorgeeft. Die output kan naar een mens gaan, maar evenzeer naar een andere computer of een databank. In dit deel bekijken we echter die computersystemen die in directe wisselwerking staan met een mens: een smartphone, een PC, een laptop, een tablet... De laag of schil waar mensen en computers in wisselwerking treden, noemen we een **interface**. Omdat de wisselwerking in twee richtingen gaan spreken we van een **interactieve interface**. Men spreekt in dit geval van human-machine interaction (HMI), man-machine interaction (MMI) of computer-human interaction (CHI).

De interface richt zich in grote lijnen op visuele output op een display. Via grafische bedieningselementen zoals knoppen, pictogrammen, schuifbalken enz. is interactieve omgang met de computer mogelijk. We spreken daarom van een **Graphical User Interface of GUI**.

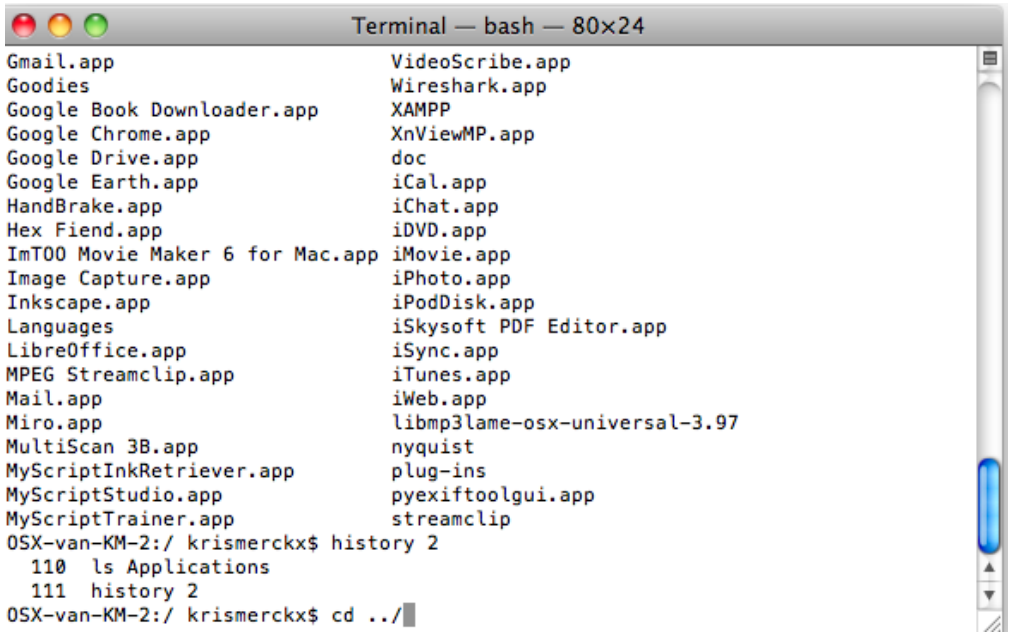
2.1.2 Soorten interactieve interfaces

Command Line Interface (CLI)

Via een command line interface geeft de gebruiker opdrachten aan het computersysteem door het invoeren van commando's. Nadeel is dat de gebruiker de benodigde commando's moet onthouden. Om dit probleem te verhelpen voorzien veel CLI's ook 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. Onder Mac OS X heeft dit net zoals op tal van UNIX- en Linux-systemen de Terminal. Linuxnerds zweren bij het gebruik van de 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).

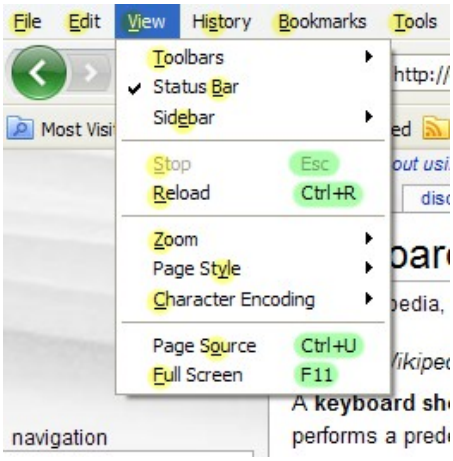
Handleiding CLI-commando's	CLI	Besturingssysteem
https://github.com/0nn0/terminal-mac-cheatsheet/wiki/Terminal-Cheatsheet-for-Mac-(-basics-)	Terminal / Bash (Born Again Shell)	Mac OS X
https://nl.wikibooks.org/wiki/Linux_voor_beginners/De_terminal en http://ss64.com/bash/	Console / Shell / Bash (Born Again Shell)	Linux
http://mally.stanford.edu/~sr/computing/basic-unix.html	Sh (shell)	Unix
http://ss64.com/nt/	Command Prompt	Windows

De huidige CLI's zijn beschikbaar naast de GUI of grafische interface. Oorspronkelijk vormden ze echter de gebruikersinterface voor de meeste computersystemen. Gebruikers gaven via tekstcommando's de opdrachten in. Een CLI ondersteunt over het algemeen het gebruik van de muis niet, enkel invoer van en via het toetsenbord. De monitor toonde de invoer van de gebruiker en de uitvoer via tekstregels die onder elkaar werden weergegeven. De volledige ruimte van het scherm kan niet benut worden. De computermonitoren konden doorgaans slecht een beperkt aantal kleuren of slechts één kleur weergeven.

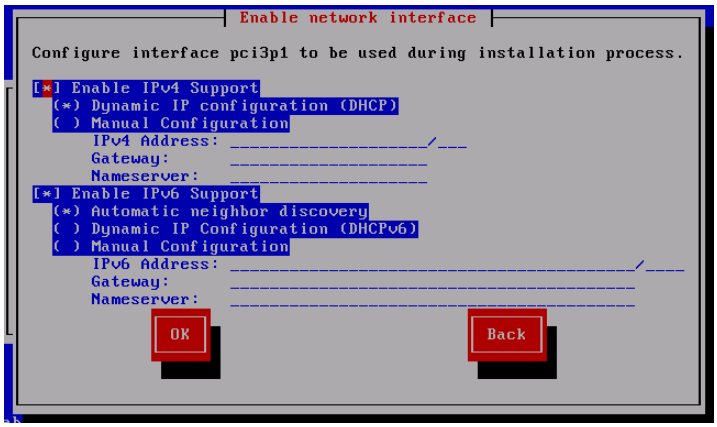


Text User Interface (TUI)

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 GSM's.



Afbeelding: Mnemonics en “shortcuts” in Firefox 3 onder Windows.



Graphical User Interface (GUI) en WIMP

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.

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.

De WIMP-interface is niet bedacht door Apple of Microsoft, ook al lees je vaak het verhaal dat Microsoft de GUI is gaan "stelen" van Apple. Douglas Engelbart (1925) startte een eigen bedrijf in opslagsystemen, maar succes bleef uit. Hij begon te werken aan het 'Augmentation of Human Intellect project' aan het Augmentation Research Center van sri International in Menlo Park (Californië). In 1963 experimenteerde hij daar met een aantal aanwijsapparaten (pointing devices), zoals de trackbal, de lichtpen, joysticks en ... de muis. Hij vergeleek hun snelheid en nauwkeurigheid. Uit alle 'kandidaten', waaronder ook knie-, neus- en hoofdbesturingen, kwam de muis als grote overwinnaar uit de strijd. Hij bedacht ook de grafische gebruikersinterface met vensters, netwerksoftware, het knippen en plakken van tekst op een monitor, hypertekst (aanklikbare tekst) en teleconferencing. Hij vertrok vanuit het leerproces van kinderen zoals de oog-handcoördinatie en niet vanuit wat professionele computergebruikers in die tijd verwachtten. Op 8 december 1968 presenteerde hij zijn technieken in de allereerste teleconferencing ooit via het vooruitstrevende nls (oNLine System). In de vroege jaren zeventig stapten heel wat werknemers van sri over naar Xerox Parc, dat een paar jaar later startte met de ontwikkeling van een computer voor persoonlijk gebruik (pc). Het mag dan ook geen verbazing wekken dat veel ideeën van Douglas werden geïntegreerd. Het was de eerste computer met een grafische gebruikersinterface (gui) met een virtueel 'bureaublad' als schermmetafoor. Veel ideeën van deze eerste gui leefden en leven door in latere besturingssystemen zoals Microsoft Windows, linux kde en gnome, Mac os ... Het virtuele bureaublad had vensters, aanklikbare iconen, uitklapmenu's voor terugkerende taken zoals het openen, verplaatsen of verwijderen van bestanden. In 1974 begon de ontwikkeling van het programma Gypsy een soort teksteditor die werkte volgens het 'What you see is what you get' principe (wysiwyg): wat je op het scherm zag, kun je ongewijzigd afdrukken op een 'printer'. In 1981 kwam de Xerox Star op de markt, maar de prijs was te hoog om door te breken bij het grote publiek. Andere ontwikkelaars zoals Microsoft, Apple en Amiga (deze firma werd in 1984 opgekocht door het beroemde Commodore om een vervolg te breien aan hun Commodore64-succesverhaal) hadden op dat moment hun ogen al lang de kost gegeven.

Natural User Interface (NUI): Speech, Touch and Gestures (STAG)

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 inet van een leien dakje. Hoe dit komt, leer je in deel 3.

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. Samen met de Wiimote, tablets (iPad) en multitouch-smartphones behoort het tot de eerste generatie STAG (Speech, touch and Gestures)- toestellen. En de Kinect is al zeker zonder enige twijfel het eerste NUI (Natural User Interface)- apparaat voor de massa! 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 6

personen en kan het bewegingsanalyse uitvoeren op 2 actieve spelers. 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.

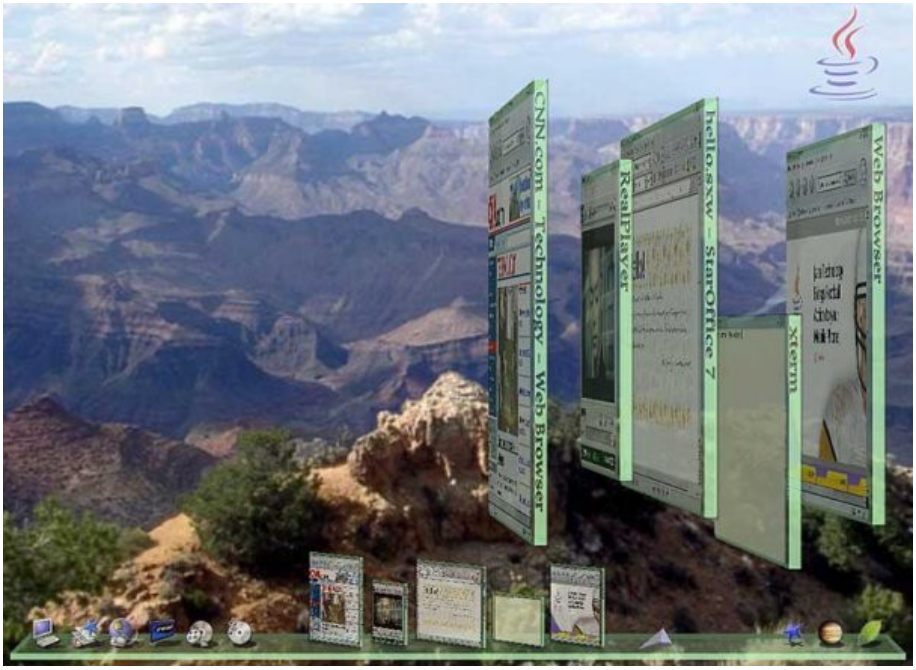
Tangible user interface (TUI)

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.

Zooming/zoomable user interface (ZUI)

Jef Raskin, ontwikkelaar van de GUI voor het oorspronkelijke Mac OS-systeem, ontwierp THE (The Humane Environment), later omgedoopt tot Archy. Hij combineerde in die omgeving de kracht van een CLI en een GUI. Archy had geen "eindige" virtueel bureaublad (zoals we dat bij Windows en Mac OS X nog steeds terugvinden), maar een oneindige ZUI. Raskin doopte zijn zoomable interface Zoomworld. Zijn ideeën vonden hun weg naar de Canon Cat-computer en worden nu verder ontwikkeld door het Raskin Center for Humane Interfaces. Sinds zijn dood in 2005 probeert zijn team de focus te verleggen naar de integratie van de "Archy"-ideeën in andere software, zoals de Ubiquity-extensie voor Firefox.

De pogingen om een GUI te vervangen door een volledige ZUI, zijn tot nog toe niet echt succesvol te noemen (https://en.wikipedia.org/wiki/Zooming_user_interface#History). Ook de combinatie van een ZUI met een 3D-ervaring zoals bij Sun's Looking Glass-GUI voor Solaris UNIX, leverde niet het verhoopte succes. In combinatie met een traditionele GUI raakte de ZUI wel ingeburgerd via de "zooming"-functionaliteit in Apple's iOS voor de iPhone en iPad. De zoomfunctionaliteit wordt door gebruikers als zinvol ervaren als men hierdoor bepaalde informatie in meer detail kan bekijken zoals bij Google Earth, Google Maps, het bekijken van producten in Ebay... of wanneer alle informatie niet in één keer op een "klassiek" bureaublad kan weergegeven worden zoals bij de online presentatietool Prezi.



De 3D-ZUI Looking Glass van Sun Microsystems

Augmented Reality

Het is niet zo eenvoudig om het begrip **augmented reality** duidelijk te omschrijven. In het Nederlands vertaalt men het vaak als 'toegevoegde realiteit', maar 'uitgebreide' of 'verhoogde werkelijkheid' dekt misschien beter de lading. In de rest van het boek zullen we de Engelstalige afkorting AR gebruiken. Bij AR worden directe of indirecte live beelden aangevuld met digitale door de computer gegenereerde informatie. Die informatie wordt als een gedeeltelijk transparante laag over de live-beelden heen gelegd. De term AR zou in 1990 voor het eerst gebruikt zijn door Thomas Caudell, een ontwikkelaar bij Boeing. Hij gebruikte de term voor een head mounted display (HMD) die de arbeiders die de bedrading in de vliegtuigen aanbrachten, moest helpen bij die ingewikkelde taak. De digitale beelden verrijken hun beeld van de werkelijkheid. De HMD zelf was geen uitvinding van Thomas Caudell, maar werd reeds in de jaren '60 van de vorige eeuw ontwikkeld door de computerwetenschappers Bob Sproull en Ivan Sutherland, voor de weergave van virtual reality-beelden. Tegenwoordig kennen we AR vooral van applicaties zoals Layar op de smartphones met iOS en Google Android. Die AR-apps nemen met de webcam een live beeld van de omgeving en voorzien dit van een extra laag informatie. Omdat de zoekopdracht niet meer door de gebruiker wordt ingegeven, maar automatisch door het toestel wordt opgevraagd en weergegeven, lijkt het alsof we hier met kunstmatige intelligentie of artificial intelligence (AI) te maken hebben. Vaak is de waarheid iets minder geavanceerd, maar je kan het gerust als een eerste stap zien in de richting van AI in de dagdagelijkse werkelijkheid. De invoer voor AR komt soms, maar lang niet altijd via user-input, maar tegenwoordig vaak via sensoren en chips. Een smartphone met AR-functionaliteit registreert via een GPS-chip de lokatie van de gebruiker en via een digitaal kompas zijn oriëntatie, vergelijkt de resultaten met de informatie in een databank en toont het eindresultaat aan de gebruiker. Bij indirecte AR zijn de gegevens vaak wel door "onzichtbare" personen ingegeven, daarom stellen gebruikers zich in dit geval minder vragen.

Wat is het verschil tussen Augmented Reality en Virtual Reality? Paul Milgram en Fumio Kishino definieerden in 1994 het "Milgram's Reality Virtuality Continuum". Ze onderscheiden in "mixed reality" vier stappen:

1. real environment (de echte omgeving)
2. augmented reality (AR)
3. augmented virtuality (AV)
4. virtual environment (volledig virtuele omgeving)

Bij "real environment" gaat het niet zo zeer over het waarnemen van de werkelijkheid, maar nog steeds over het mengen van echte met virtuele beelden. Een bekend voorbeeld zijn TUI's (Tangible User interfaces) of NUI's (natural user interfaces). Hierbij gebruik je echte voorwerpen (eventueel uitgerust met sensoren) of mensen voor de interactie met computers. Een aantal voorbeelden van TUI's vind je op http://www.guillaumeriviere.name/collection/tui.html.

Heel erg ingeburgerd zijn toestellen als de Wii en de Xbox Kinect zijn de KUI's (Kinetic User Interfaces) waarbij natuurlijke bewegingen van voorwerpen of mensen zorgen voor de interactie met computers.

Bij AR maakt men een onderscheid tussen Spatial AR, See-through AR en Projection Augmented Models (PA-models).

1. Bij Spatial AR worden door de computer gegenereerde informatie in de omgeving geprojecteerd. De gebruiker kan interactief met de projecties omgaan of de projectie/computer reageert op de gebruiker.
2. Bij See-through AR draagt de gebruiker een HMD, AR-contactlens of bril.

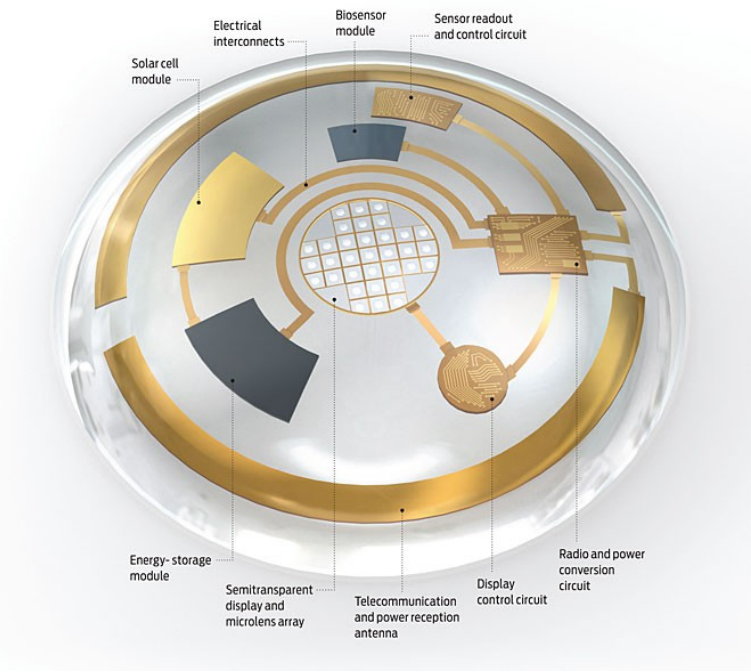
AV doet net het omgekeerde van AR. Het voegt echte beelden toe aan een virtuele omgeving.

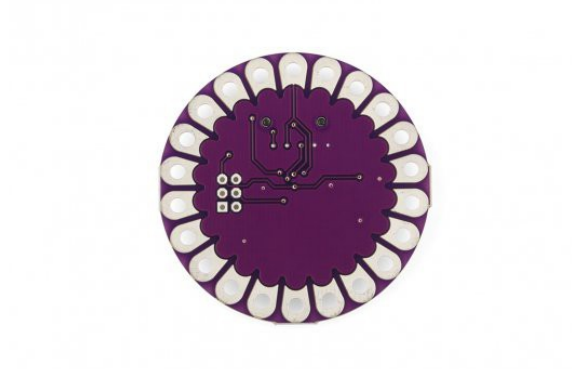
Vormen van AR en de fysieke wereld als interface

De meest geavanceerde head mounted display of head-up display is in gebruik bij de Amerikaanse luchtmacht. Piloten van een F-35 kunnen de gewenste vorm van informatie selecteren. 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.

Oorlogvoering is gelukkig niet het enige toepassingsgebied van HUD's en HMD's. Ook voor games, extreme sporten, geneeskunde, wetenschappelijk onderzoek enz. bestaan er gelijkaardige systemen die de gebruiker bij zijn taak moeten helpen. Voor de consument is een HUD natuurlijk een beetje te veel van het goede, maar de ontwikkeling gaat ook op dit vlak razendsnel. Een simpele AR-bril of zelfs AR-contactlens behoort tot de mogelijkheden. Het idee van AR-contactlenzen stamt uit SF-verhalen. Het is misschien wel wat raar om aan de cyborg in Terminator AR-contactlenzen te geven. In het geval van een robot zullen de digitale data rechtstreeks in het virtuele brein zitten als een reeks te verwerken instructies. Maar het schept natuurlijk wel de juiste sfeer voor de film en de achtervolgingsscenes. In de boeken van de SF-schrijver, wiskundige en informaticus Vernor Vinge hebben de personages geen hersenimplantaten of smartphones voor AR-informatie, maar elektronische contactlenzen.

De **Virtual Retinal Display** van Microvision (www.mvis.com) is een vorm van HUD waarbij de beelden rechtstreeks op het netvlies worden geprojecteerd. Babak A. Parviz, professor bionanotechnologie aan de universiteit van Washington (Seattle) werkt samen met een team van professoren en studenten aan een AR-contactlens. Een eerste werkende versie is klaar voor "massaproductie", ook al is de functionaliteit op dit moment nog erg beperkt. Naast een glucosemonitor bouwde zijn team ook biosensoren die informatie met bepaalde molecules kunnen uitwisselen via elektrische signalen. Ze hebben m.a.w. alle technologie voor handen die nodig is voor de verdere ontwikkeling van AR-contactlenzen. De lenzen zijn ingepakt in een polymeerlaag en grondig op veiligheid getest (op konijnen!). De lens voorziet zichzelf van energie via lage radio-frequentie-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. Anders dan bij een computermonitor of smartphonescherm waar het grootste deel van de uitgezonden energie verloren gaat, helpt bij de AR-lens elk foton bij de opbouw van het AR-beeld omdat de lens meteen op het oog ligt. Eens voldoende ontwikkeld kan de lens tekst, grafieken, kaarten, afbeeldingen of zelfs film direct 'op het oog' weergeven. Uiteraard moet in dat geval nog steeds een extern draagbaar toestel zorgen voor het draadloos doorgeven van de informatie aan de lens. Maar het is niet moeilijk om toekomstige mogelijkheden voor deze vorm van nanotechnologie te bedenken. Mogelijk vervangt het in de toekomst computerschermen!





Het uitblijven van succes van **Google Glass** bewijst dat AR niet meteen zijn weg vindt naar het grote publiek. Toch is de vermening van de digitale en de fysieke wereld reeds ingezet: de Apple Watch verzamelt informatie over onze gezondheid en bewaart die in de "cloud" (big data). De **Apple Watch** behoort bij de categorie van de wearables, waar ook "intelligente kleding" ondergebracht kan worden. Ook van de Arduino-microcontroller bestaat er een wearable versie en de nodige materialen om zelf intelligente kleding samen te stellen (https://store.arduino.cc/category/30). De **LilyPad Arduino** is speciaal ontwikkeld voor wearables en e-textile. T. Olsson, D. Gaetano, J. Odhner en S. Wiklund beschrijven in hun boek "Open Softwear" hoe je software ontwikkelt voor e-textile.

Niet enkel kleding, maar ook in je woning of klaslokaal kan je de fysieke en de digitale wereld in elkaar laten overvloeien. Met **Bare Conductive Touch Board Starter Kit** en elektrisch geleidende verf kan je AR-applicaties bouwen.

Virtual Reality

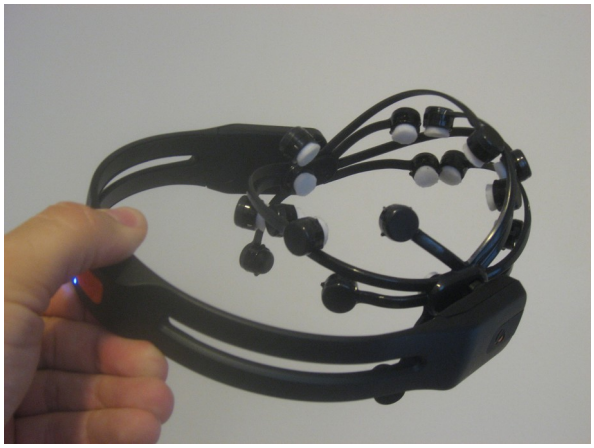
Virtual Environment of Virtual Reality is het andere uiterste. Het volledige beeld wordt door de computer gegenereerd. Wel kunnen er net zoals in een 3D-programma afbeeldingen uit de echte wereld als textures (vullingen) op virtuele objecten worden 'gemapt' (virtueel geprojecteerd).

Hier maakt men een onderscheid tussen Semi immersive VR en Immersive VR. Bij Semi Immersive VR vult het VR-beeld slechts een deel van het gezichtsveld van de gebruiker. Denk bijvoorbeeld aan een "flight simulator". Een Immersive VR-beeld vult het volledige gezichtsveld door gebruik van een HMD of projecties waarbij beelden over gans de omgeving van de gebruiker.

onderschrift: Het programma Terragen is gespecialiseerd in het 'uit het niets' genereren van landschappen, zeeën, gebergtes, wolken... Het resultaat zijn hyperrealistische beelden die in vrijwel niets te onderscheiden zijn van echte foto's (Bron afbeelding: www.picolous.com).

Het onderscheid tussen de verschillende fasen in het Milgram's continuum zijn niet altijd even duidelijk en vastomlijnd. Zo vormen PA-models echt een randgeval tussen Spatial AR en TUI's. Bij een PA-model wordt op een fysisch object een computerafbeelding geprojecteerd (gemapt), op een gelijkaardige manier als in een 3D-programma een vulling op een object wordt geplaatst.

Brain Computer Interfaces



Het lijkt ontsnapt uit een goedkope SF-film, maar het kan en het is beschikbaar voor een betaalbare prijs, het eerste invoerapparaat op basis van hersengolven (een neurale interface). De **Emotiv Epoc** is voor een slordige 300 dollar te bestellen. Een SDK (software development kit) is beschikbaar voor de iets minder gebruikersvriendelijke prijs van ongeveer 750 dollar. Met de open source **OpenViBE**-software, een C++-library voor Linux en Windows, kan je op basis van de Emotive-headset je eigen applicaties schrijven. De eveneens open source EMOKIT-software is speciaal ontworpen om samen te werken met de Emotive. OpenViBE wordt ontwikkeld door het Franse Institut national de recherche en informatique et en automatique (http://openvibe.inria.fr/) en is bestemd voor het ontwerpen, uittesten en gebruiken van **Brain-Computer Interfaces (BCI)**. Een onwaarschijnlijke demonstratiefilm vind je op http://openvibe.inria.fr/video.php?q=video-openvibe-introduction-en.

EMOKIT daarentegen is een eenmansproject van de hacker DAEKEN en is ultrasimpel in gebruik. De softwarekit is gebaseerd op de programmeertaal Python (maakt gebruik van pywinusb en python-hid) 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. Je vindt de software op http://daeken.com/emokit-hacking-the-emotiv-epoc-brain-computer-0 en https://github.com/daeken/Emokit

Japanse onderzoekers slaagden er in 2008 reeds in om op basis van een fMRI-scan van de hersenen beelden te reconstrueren. Het onderzoek naar visual image reconstruction from brain activity gaat door aan verschillende onderzoekscentra en men bereikt op dit moment al onvoorstelbare resultaten.

Links:

- http://www.cns.atr.jp/dni/?id=30
- http://www.youtube.com/watch?v=MElU0UW0V3Q&feature=player_embedded

Leven we in een computer? (The Matrix)

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."

2.1.3 User Interface design en UX-design

UX-design staat voor User Experience design. Bij het ontwikkelen van digitale interfaces moet men uitgaan van een "User centered design process". De ontwikkelaar(s) moeten verschillende factoren tegen elkaar afwegen:

- User Centered Design Process
- User needs
- Business needs

Bij het ontwikkelen van een interface komt heel wat kijken:

- visual design
- information architecture
- user interface design
- usability

Bekijk zeker eens dit filmpje: https://www.youtube.com/watch?v=Ovj4hFxko7c

De ontwikkelaar bevindt zich op het knooppunt tussen het digitale systeem en de eindgebruiker. Aan de kant van de machine moet hij 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 HCI (Human Computer Interface) zelden het werk van één persoon.

2.1.4 Interactie-ontwerp

In dit deel bekijken we niet zo zeer hoe je een interactieve interface ontwerpt, want daarvoor moet je doorgaans al aardig wat kunnen programmeren. We bekijken echter een aantal universele ontwerpprincipes, die niet enkel gelden voor HC-interfaces, maar voor elk product dat een bedieningsinterface nodig heeft, en bij uitbreiding voor nagenoeg elk ontwerp (product, affiche...) dat je ooit zal maken of bedenken. De tips die je in dit onderdeel leest, zijn niet altijd "wetenschappelijk" bewezen, maar in veel gevallen eerder "good practice". Hanteer ze bij het ontwerp van presentaties, websites, producten, affiches, logo's, reclamefolders. Uiteraard bespreken we onderstaande tips vooral in het kader van HC-interfaces.

Tip: *Met de open source tool Pencil (http://pencil.evolus.vn/) kan je snel een GUI prototypen. Dat wil zeggen dat je er een interface mee kan bouwen van hoe een programma of GUI voor een applicatie er volgens jou zou moeten uitzien.*

1. Modulariteit

Arduino en Little bits uit het vorige deel zijn voorbeelden van modulair ontwerp. Ook desktopcomputers en in wezen elke "thuis"computer, zijn voorbeelden 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 in moderne programmeertalen (zogenaamde object-georiënteerde talen) werken modulair, met klein 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. Applehardware (iPhone, iPad) is vaak bewust heel beperkt modulair. Je kan de opslagcapaciteit niet

eigenhandig uitbreiden, je bent als gebruiker beperkt bij het aansluiten van randapparatuur. Nochtans is ook de "binnenkant" van pakweg een iMac heel erg modulair opgebouwd (www.ifixit.com).

2. 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.

3. Toegankelijkheid en gebruiksvriendelijkheid

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.

a. gelaagd ontwerp via de behoeftepiramide

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. Een tekstverwerker bijvoorbeeld moet in eerste instantie de mogelijkheid bieden om tekst in te voeren, tekstopmaak, het toevoegen van tabellen en afbeeldingen zijn hogere behoeften.

Hierin volgt een ontwerper de behoeftepiramide.

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)

b. flexibiliteit versus gebruiksgemak:

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. Een PC is bijvoorbeeld heel flexibel, maar moeilijker aan te leren en te bedienen dan een smartphone. 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 dathij enkel af en toe wat wil surfen op het internet, volstaat een tabletcomputes. 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 steevast voor de simpele (de regel van "Ockhams scheermes").

c. Mapping:

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.

d. Prestaties

Het is niet omdat iets beter "presteert", dat de voorkeur van de gebruiker daarnaar uitgaat.

e. Herkenning 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. De in besturingssystemen aanwezige API's maken het voor programmeurs mogelijk om herkenbare dialoogvenster 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.

4. Stijl en opmaak

a. Gepercipieerd gebruiksgemak:

Esthetische ontwerpen wekken de perceptie dat ze eenvoudiger in gebruik en dus gebruiksvriendelijker zijn, ook al is dit niet automatisch zo. Esthetische ontwerpen nodigen meer uit tot een positieve attitude. Mensen zijn ook toegeeflijker en toleranter tegenover ntwerpfouten als het ontwerp er esthetisch goed uitziet. Een mooi interface-ontwerp werkt als katalysator voor de creativiteit en probleemoplossend denken. Minder esthetische ontwerpen, verstikken de creativiteit. 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.

b. Kleuren:

"Over kleuren en smaken valt niet te discussiëren" zegt het spreekwoord. Nochtans klopt dit niet helemaal. Je kan twee kleuren die op het eerste zicht met elkaar "vloeken" bij elkaar plaatsen als dat uw smaak is. Maar wanneer je er andere kleuren of tinten bij betrekt, gaat die stelling niet meer op. Passende kleurcombinaties kan je wiskundig berekenen. Niet alle kleuren of tinten passen bij elkaar. Op paletton.com kan je passende kleurpaletten berekenenen voor je eigen ontwerpen.

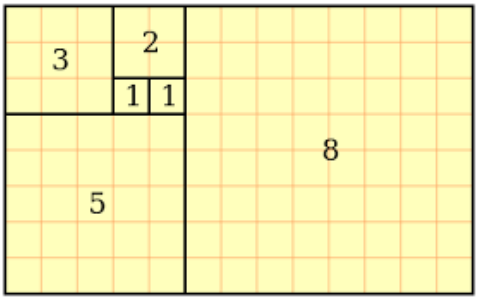
Consistentie van stijl en kleur maken een systeem meer helder en bruikbaar.

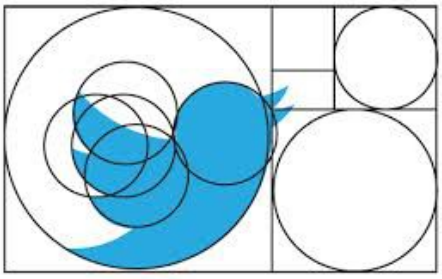
c. Fibonaccireeksen en de Gulden Snede

Oeps, dit lijkt wel een stukje wiskunde en dat is het ook. Fibonaccireeksen (oneerbiedig ook wel eens "konijnenreeksen" genoemd naar het voortplantingsgedrag waarmee deze dieren worden vereenzelvigd) komen in de natuur heel vaak voor. In een Fibonaccireeks is elk volgend getal de som van de twee voorgaande getallen

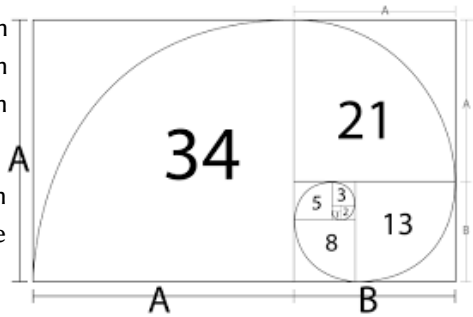
```
1 1 2 3 5 8 13 21 34 …
```

Fibonaccireeksen duiken op in de natuur, maar ook in de vormgeving.





Vaak gebruikt men Fibonaccireeks in nauw verband met de zogenaamde Gulden Snede, één van de invloedrijkste patronen in wiskunde en vormgeving. Bij de Gulden Snede snijdt men een rechthoek zodanig in twee delen dat (...*lees nu even heel aandachtig* ...) het grootste deel en het kleinste deel zich net zo verhouden als het grootste deel tov. de ganse rechthoek. Euh???? Je merkt het al in het bovenstaande voorbeeld. In de rechthoek hebben we een vierkant afgezonder van 8 bij 8 ruitjes en eentje van 5 bij 8 ruitjes. Je kan die kleinste rechthoek opnieuw gaan indelen zodat je (net zoals in een Fibonaccireeks) opnieuw een gelijkaardige verhouding verkrijgt.



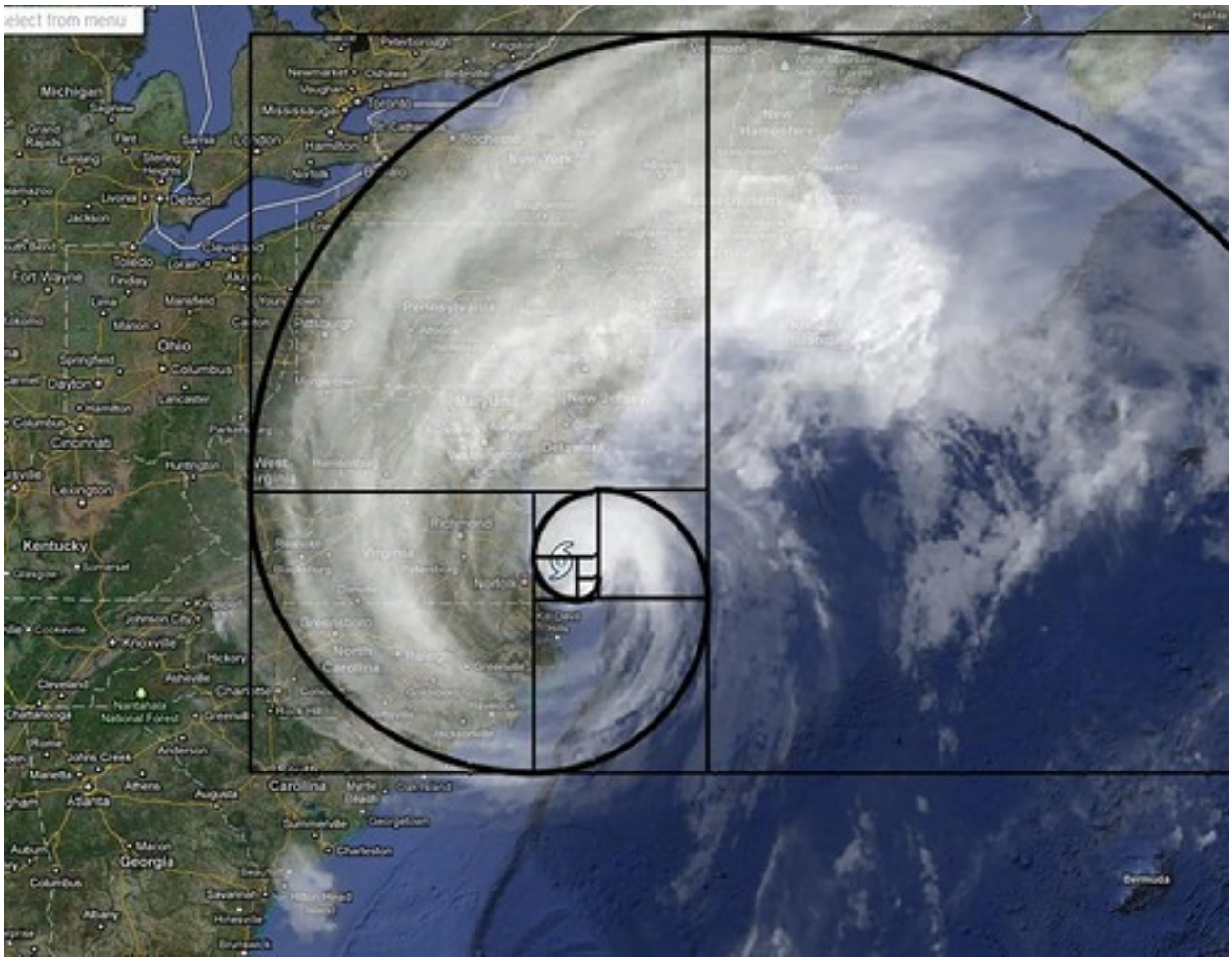
Plaats je een passer op het scheidingspunt tussen twee vierhoeken en trek je een kwartcirkel, dan bekom je een soort van spiraalvorm. De

Gulden Snede duikt op in de natuur en reeds in de Oudheid gebruikten kunstenaar het in de architectuur en de kunst.

De Gulden Snede bereken je op de volgende manier: a : b = (a+b) : a (waarbij a en b de verhouding zijn van de brede en de lange zijde). De verhouding tussen a en b heet het Gulden Getal en uitgedrukt met de Griekse letter PHI (opgelet: niet PI of 3,14…)

$$\varphi = \frac{1+\sqrt{5}}{2} \approx 1{,}618$$

Is het een voorbeeld van onze onbewuste voorkeur voor esthetiek en verhoudingen?



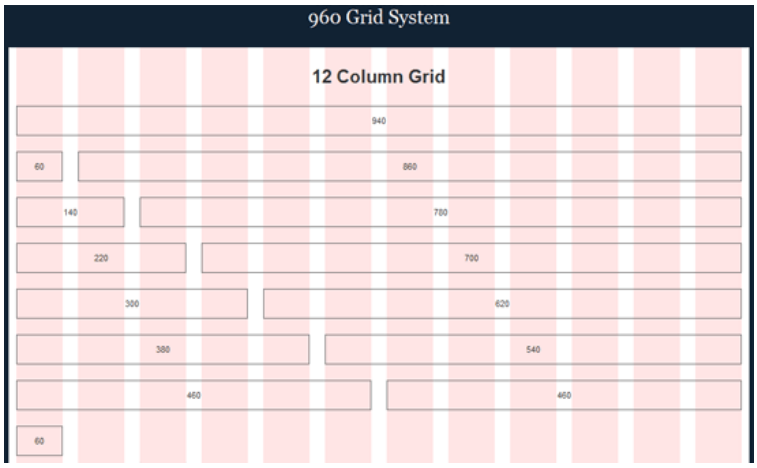
d. Vorm volgt functie:

In de architectuur en vormgeving dook vooral sinds de twintigste eeuw de idee op dat esthetiek ondergeschikt is aan de functie. Functies zijn inderdaad minder subjectief dan esthetische overwegingen. Toch heeft het voorgaande al duidelijk aangeduid dat esthetiek niet onbelangrijk is. Als ontwerper kies je niet vorm en functie. De vorm kan best wel afgestemd zijn op de functie. De functionaliteit mag niet verloren gaan achter het uitzicht.

e. Symmetrie:

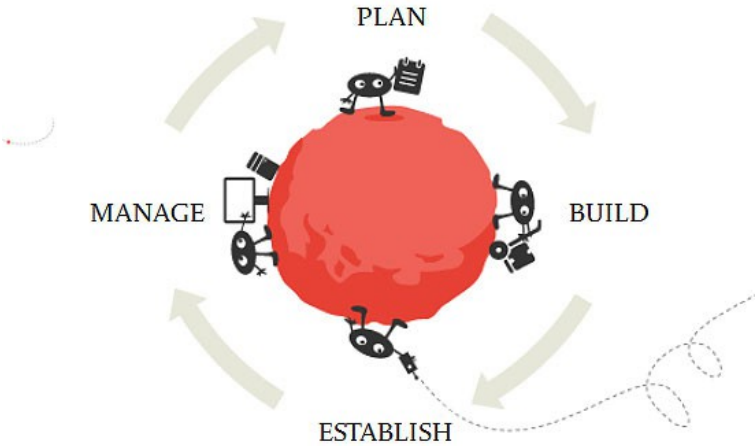
Symmetrie wordt altijd als schoon ervaren. We onderscheiden drie vormen van symmetrie

- Weerspiegeling,
- rotatie,
- translatie

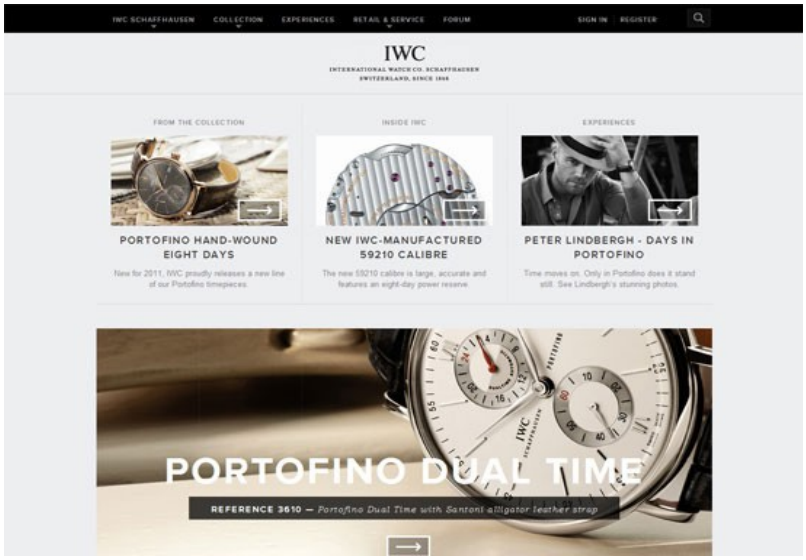


Translatie-symmetrie in een gridsysteem voor websites.[2]

Rotatie-symmetrie[3]



Symmetrie in webdesign.[4]



2 KNIGHT, K., "Symmetry in Design: Concepts, Tips and Examples", (http://sixrevisions.com/web_design/symmetry-design/), Geraadpleegd op 9 september 2015.
3 Ibid.
4 Ibid.

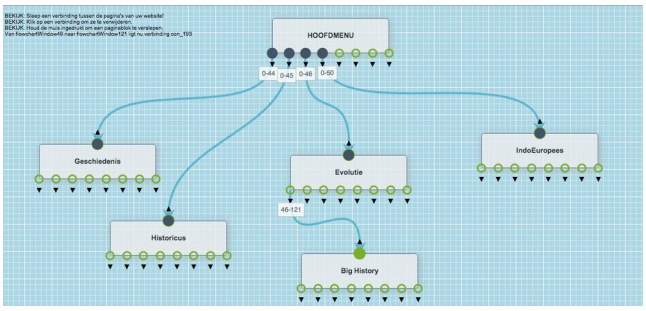
5. Leesbaarheid en bestendiging

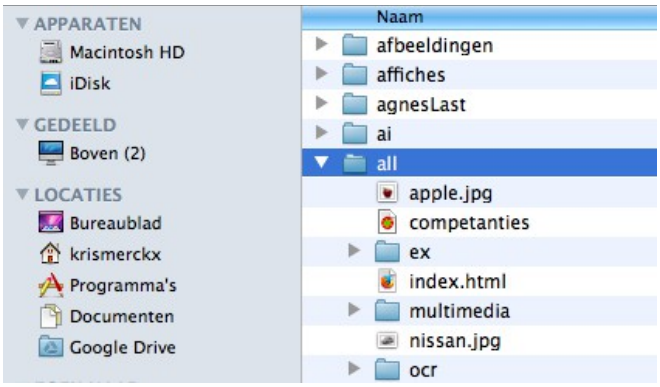
Een interface/ontwerp moet vlot leesbaar zijn. Het letterbeeld, het gekozen lettertype, de lettergrootte kunnen de leesbaarheid bevorderen of kraken. In doorlopende tekst gebruik je een vlot leesbaar lettertype. Schreefloze letters zijn vlotter leesbaar voor dyslectici, maar uiteindelijk haalt iedereen er voordeel uit.

Bron[5]

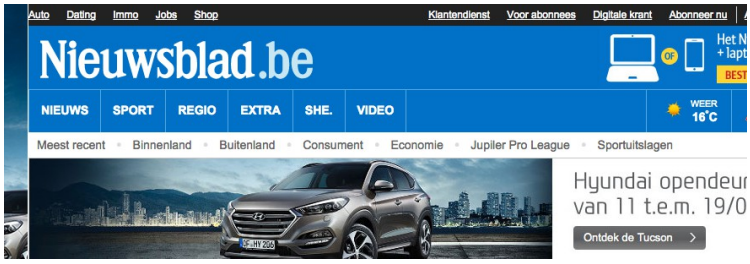
Leesbaarheid gaat verder dan enkel het kunnen lezen van een aanwezige tekst. De informatie moet vlot opgenomen worden en bestendigen in het geheugen en de herinnering van de eindgebruiker. Informatie in welke vorm ook (teksten, afbeeldingen, artikels, invulformulieren...) kunnen zo gepresenteerd worden dat de informatie vlotter toegankelijk is en de gebruiker meteen een duidelijk overzicht krijgt.

a. Hiërarchie: De zichtbaarheid kan erg verbeteren door hiërarchische verbanden te visualiseren. Die techniek is zeer effectief om de structuur en de toegankelijkheid te te bestendigen in het geheugen van de gebruiker. Dit kan op diverse manieren met boomstructuren, netstructuren, trapstructuren, lagen...





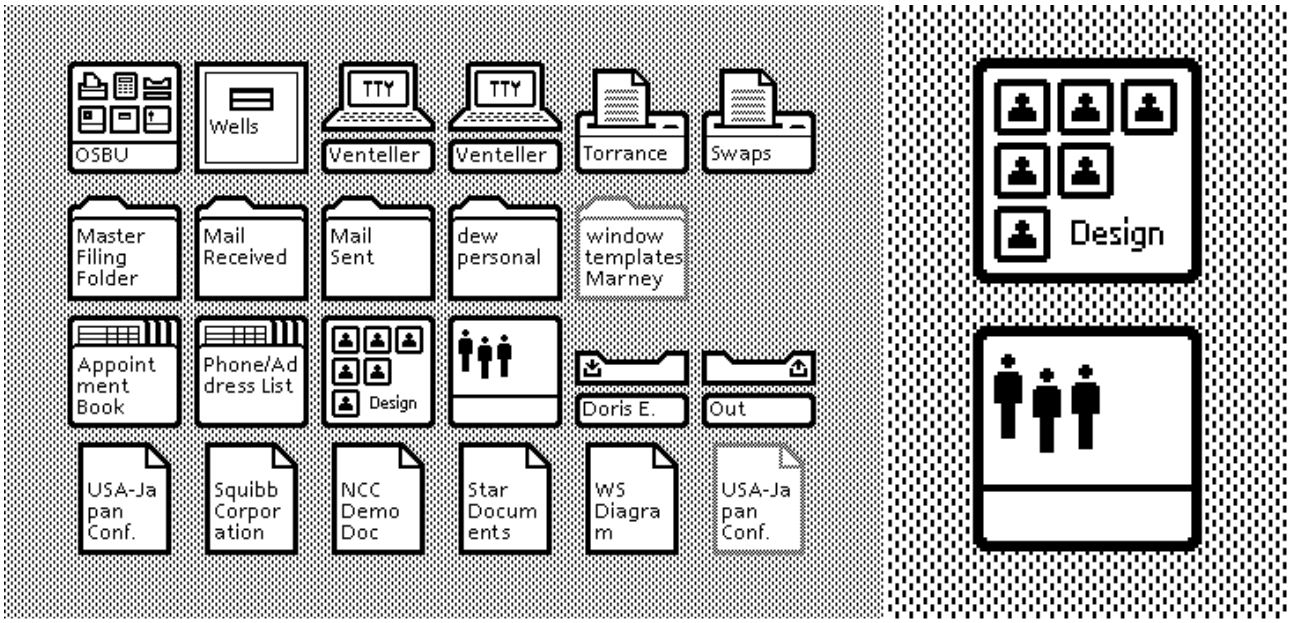
b. Chunking: Splits de informatie die je aanbiedt op in kleine eenheden. Mensen kunnen 5 tot 7 onderdelen vlotjes onthouden. Om die reden vind je in dit cursusboek 7 delen en vaak 5 tot 7 subonderdelen.



5 "Welk lettertype gebruik ik het best?", (http://www.wallacesanders.be/veelgestelde-vragen/welk-lettertype-gebruik-ik-het-best/), Geraadpleegd op 10 september 2015.

c. Beeldende representatie: gebruik van pictogrammen en afbeeldingen

De meeste GUI's maken veelvuldig gebruik van pictogrammen. De metrointerface van Microsoft Windows is daar tendele 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" voor de Xerox Alto-computer in de vroeger jaren 1970.



Bron afbeelding[6]: Schermiconen op het bureaublad van de Xerox Star, de opvolger van de Xerox Alto (1981).

Men onderscheidt in interfacedesign verschillende soorten pictogrammen:

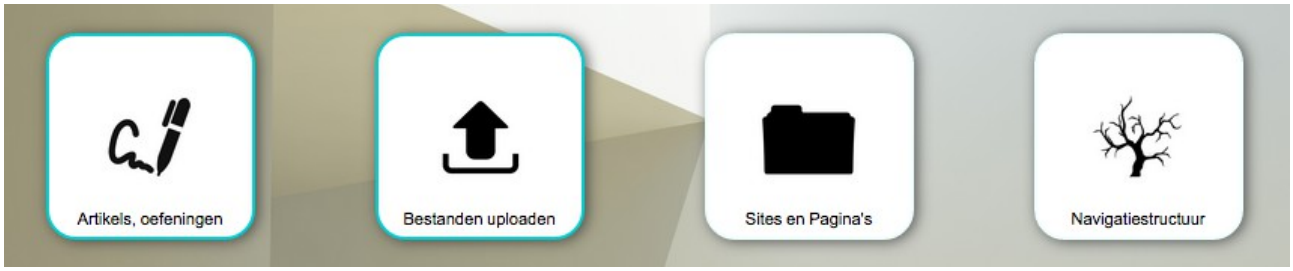
– vergelijkbare pictogrammen: tekeningen die lijken op een handeling, object of concept.

– voorbeeldpictogrammen: dingen die geassocieerd worden met een handeling (vb. hangslot voor beveiliging).

– symbolische pictogrammen: afbeeldingen van een handeling op eenhoger abstractieniveau.

– weinig of geen verwantschap (vb. een schelp in SmartSchool om naar je puntenboek te gaan).

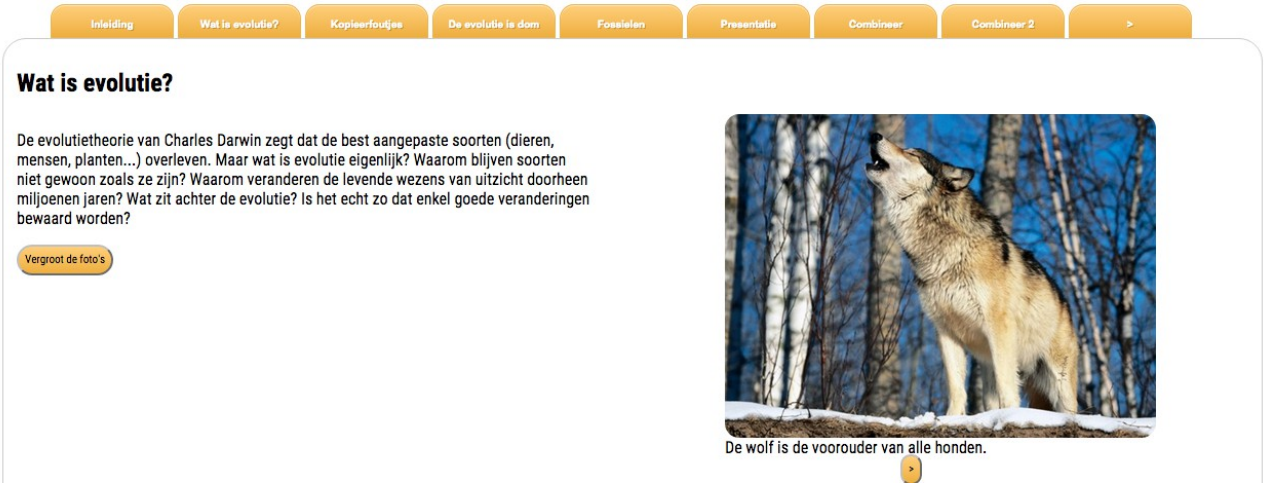
"*Een beeld zegt meer dan duizend woorden*," is een veel gehoorde uitspraak, maar klopt slechts ten dele. Men doelt dan op het zogenaamde visuele geheugen, maar er is weinig wetenschappelijk bewijs voor het bestaan van iets als een "visueel" of "auditief" geheugen. Uiteraard geldt ook hier "herkenning boven herinnering", maar meer afbeeldingen leiden 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**.



6 INTERACTION DESIGN FOUNDATION, *"A Brief History of the Origin of the Computer Icon"*, (https://www.interaction-design.org/ux-daily/21/a-brief-history-of-the-origin-of-the-computer-icon), Geraadpleegd op 10 september 2015.

d. Stapsgewijze informatieverstrekking (progressive disclosure):

De complexiteit van een toepassing kan aardig verlagen door de informatie niet alleen in kleine onderdelen (chunks) op te splitsen, maar ze ook stapsgewijs te verstrekken. Een slideshow, 'lees meer'-knop, uitklapmenu's, leerpaden (in bijvoorbeeld e-learningomgevingen), tabbladen... verhogen op die manier eveneens het leerrendement.



e. Nabijheid en similariteit:

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).

Knoppen met een gelijkende functie:

f. Van Restorff-effect:

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. Dat heet het "Van Restorff"-effect. Wil je elementen benadrukken, zorg dan dat ze opvallen. Gebruik eventueel ongewone beelden, speel met het tekstbeeld...

g. Seriële positie-effect:

Gebruikers onthouden items aan het begin van een lijst (of het nu gaat om 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.. Dit *recentheidseffect* geldt vooral bij een auditieve lijst.

h. Signaal-ruisverhouding:

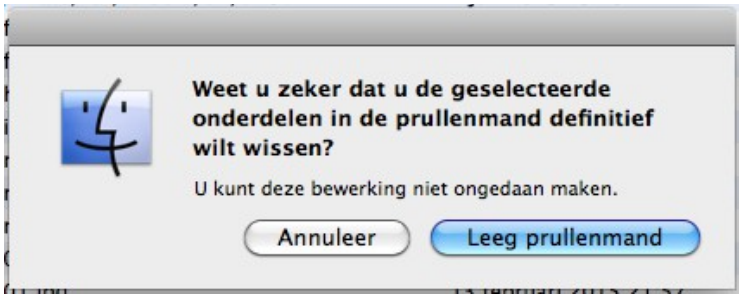
Houd een interface of voorstelling overzichtelijk, en overlaadt ze niet met te veel beeld of animatie. Alles wat de ontvanger te veel afleidt van de eigenlijke boodschap vormt ruis. Een goede interface zorgt voor betere

communicatie met de eindgebruiker. Richt je daarom op het "signaal" en beperk de "ruis".

6. Fouten beperken

Ik bestel om 19 uur's avonds treintickets op de website van de NMBS. Vooraleer ik de bestelling kan uitvoeren, moet ik de gebruiksvoorwaarden, die ik uiteraard niet lees, goedkeuren. Ik wil vooral dat dit snel is afgehandeld. Dat verwacht je ook van de gebruikersinterface. De betaling verloopt via een externe site en even later krijg ik in mijn *mail* de bestelde tickets in PDF-formaat. Wanneer ik ze afdruk, blijkt dat ik de (zo goed als) voorbije dag als afreisdatum heb ingesteld. Er zit niets anders op dan de tickets nogmaals aan te kopen en, inderdaad, nogmaals 65 euro te betalen. Ik meld mijn fout aan de online support van de NMBS, maar krijg nooit antwoord en het geld wordt ook niet terugbetaald. Als gebruiker ben ik bestraft voor mijn onoplettendheid en vooral omdat ik snel wou zijn. Nergens heeft de site me gewaarschuwd dat het wel vreemd is dat ik tickets bestel voor een dag die zo goed als afgelopen is. Nochtans is het nogal waanzinnig om tickets met toegang tot het pretpark Plopsaland de Panne te bestellen als het pretpark al een uur is gesloten. Het foutief betaalde bedrag krijg ik niet terug omdat ik "akkoord" ben gegaan met de gebruiksvoorwaarden. Toegegeven, mijn fout, maar ook: *welkom in de wereld van slecht interface-design en gebrek aan feedback*. De support achter de digitale omgeving hult zich in stilzwijgen. *In de echte fysieke wereld zouden zulke vormen van "bestraffing" als bijzonder onrechtvaardig ervaren worden.*

Een goede UI moet deze vormen van "digitale onrechtvaardigheid" voorkomen door gerichte foutcontroles in te bouwen.



a. Bevestigingsprocedure:

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?"*

b. 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 p*sychologische restricties*:

– *symbolen*: waarschuwing, icoontjes, pictogrammen kunnen gebruikersgedrag beïnvloeden

– *conventies*: rood is gevaar, geel is waarschuwing

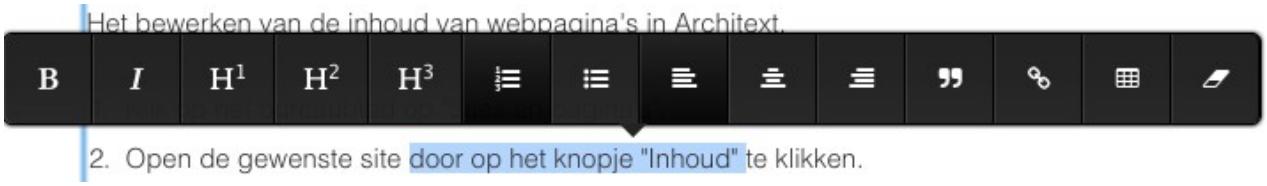
– *mapping*: welke handelingen zijn mogelijk? De zichtbaarheid, het uiterlijk, de plaatsing van de interface-elementen... spelen hier een rol.

c. Garbage in, garbage out:

Als je fouten kan invoeren, krijg je foute output (vb. in veld van postcode: niet invullen of geen cijfer invoeren). Foutcontrole voorkomt foute output. Als UI-ontwerper moet je voorkomen dat *troep* wordt ingevoerd. In webpagina's gebeurt op basis van "reguliere expressies" (zie 5.3.1) een controle op invoer van gebruikers in formuliervelden.

d. De wet van Fitts en inspanningsbelasting

Hoe verder een doel verwijderd is, hoe langer het duurt om het te bereiken. Hoe sneller de beweging 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.

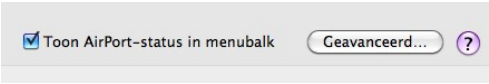


e. Operante conditionering:

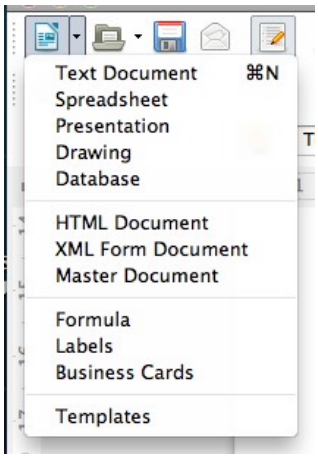
Operante conditionering is een techniek om gewenst gedrag positief te bevestigen/belonen en ongewenst gedrag te bestraffen. Het duikt vaak op in leeromgevingen (e-learning) en games. Men hanteert bij UI-design drie methodes: positieve en negatieve bekrachtiging, bestraffing (vb. volgend level of game over).

7. Functionaliteit

a. Wet van Hick en controle van de 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).



b. 80/20-regel:

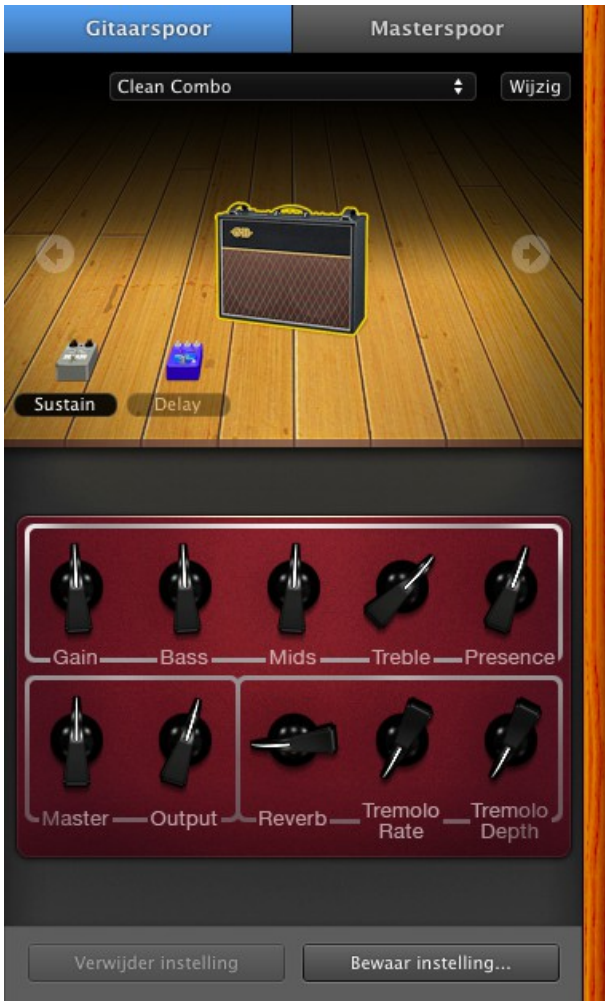
Gebruikers benutten volgens schatting 80% van de werktijd slechts 20% van de functies in een UI. Bij het ontwerp van een interface moet je je dan ook richten op die 20%. Niet-kritische functies die behoren tot de 80%, beperk je tot een minimum of je gooit ze helemaal weg. Ondermeer om die reden zit bij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

8. Herkenbaarheid

Herkenning gaat boven herinnering, Een herkenbare "omgeving" verhoogt de gebruiksvriendelijkheid.

a. Affordance en skeuomorfisme

Afbeelding: Skeuomorfisme in Apple Garageband.



De fysieke eigenschappen van een object of omgeving (bijvoorbeeld een interface) beïnvloeden het functioneren ervan. Dat klinkt nogal ingewikkeld, maar een paar voorbeelden maken veel duidelijk. Een deur nodigt uit om een kamer te betreden, een raam biedt die functie ook, maar is hiervoor minder "uitnodigend" (tenzij je een inbreker zou zijn natuurlijk). Die vorm van "uitnodiging" noemt men **affordance**. Als de affordance van een object om omgeving goed overeenkomt met de bedoelde functie, dan verhoogt het gebruiksgemak en de doelmatigheid van de prestaties. 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. De affordance zit op dat moment zeer goed.

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 andere omgeving, heet skeuomorfisme. Het beperkt zich zeker niet tot software alleen. Het nabootsen van materialen in meubels of bij interieurdesign, is eveneens een vorm van skeuomorfisme. Bij software is het de bedoeling dat zogenaamde "skeuomorphs" de herkenbaarheid en het gebruiksgemak verhogen. 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. Dit verklaart de plotse ommekeer in de Apple-interface van een sterk aanwezig skeuomorfisme naar een meer vlakke stijl in de aard van Microsofts "metro"stijl. Tegenstanders van skeuomorfisme argumenteren dat skeuomorfisme enkel een "overgangsfase" was:

"Now recall what skeumorphism really is: it's a transitional design to aid us over the bridge from old ways of doing something to the new way. But now almost all adults have crossed over that bridge. Future smartphone sales are going to be either replacement phones or to the young just entering the marketplace. And those latter won't have any clue at all about those old ways of doing things. Who currently under the age of 20 will ever see a Rolodex outside a museum (or their grandparents' study)?" (Tim Worstal)[7]

7. WORSTAL, T., "Apple's iOS7, Well, It Was Time For Skeuomorphism To Die", (http://www.forbes.com/sites/timworstall/2013/09/19/apples-ios7-well-it-was-time-for-skeuomorphism-to-die/), Geraadpleegd op 6 oktober 2015.

Toch wil dit geenszins zeggen dat hiermee elke overeenkomst met "fysieke" objecten of omgevingen overboord wordt gegooid.

b. Metaforen

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. Adobe Flash noemt zijn werkgebied de "scene" en objecten kan je bewaren in de "library". Gelijkenissen en metaforen zorgen voor herkenbaarheid en dat is misschien maar goed ook.

c. 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:

1. nabootsing van uiterlijk	het uiterlijk lijkt op iets anders	pictogram van een map
2. nabootsing van gedrag	het ontwerp gedraagt zich zoals iets anders	een robot die zich beweegt zoals een insect
3. nabootsing van functie	het ontwerp werkt zoals iets anders	toetsenbord van een rekenmachine bij een oude Nokia-GSM.

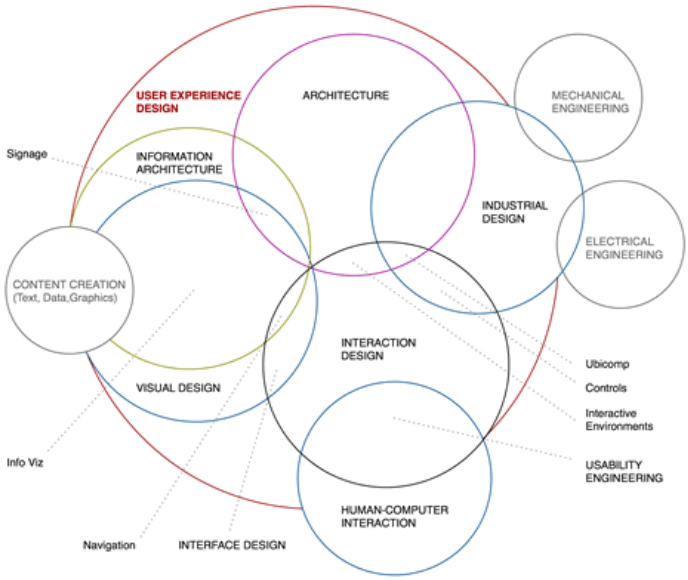
9. Storytelling

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. Bij User Interface- of User Experience-design kan storytelling op nog een andere manier ter hulp schieten.

Het ontwerp bepaalt hoe gebruikers (mensen) het eindproduct ervaren. Dit gebeurt op verschillende niveaus en vertaalt zich in drie vormen van ontwerp: [8]

Visceral design	we houden automatisch van bepaalde dingen (knappe mensen, symmetrie...) en voelen ons afgestoten van andere (spinnen, rot vlees...). Dit niveau bepaalt onze eerste relatie met een ontwerp.	We kiezen er bewust voor om "bang" te zijn als we naar een griezelfilm gaan kijken.
Behavioral Design	Hoe werkt het product? De look and feel, gebruiksvriendelijkheid...	Hoe ervaren we de film?
Reflective Design	Welke waarde kennen we aan het product toe na het eerste gebruik.	We weten dat de vampieren niet echt bestaan en ons (zeker niet vanuit de film) lastig kunnen vallen.

Bij het ontwerp van een product, dienst, interface... moeten we terdege rekening houden met die drie niveaus. 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. Er moet rekening gehouden worden met grafisch ontwerp, technische mogelijkheden en beperkingen, commerciële overwegingen enz. Het is niet eenvoudig om in zo'n samenwerkingsverband de eindgebruiker centraal te stellen. De eindgebruiker is het "Experience Theme" (Cindy Chastain), de hoofdrolspeler van het UX-ontwerp. Zonder zo'n leidraad kan er wel een product afgeleverd worden, maar dan blijft het een verhaal zonder doel, of beter gezegd een verhaal zonder rode draad.



Oef, dit klinkt allemaal nogal ijl en wellicht denk je, wat moet ik hiermee? Nochtans is het niet zo vreemd dat het uiteindelijke doel van een ontwerp een eindgebruiker (en liefst meer dan één is). Ongetwijfeld heb je al een beeld voor ogen van het doelpubliek. Door fictieve personages met al even fictieve, maar potentieel mogelijke, verhalen te verzinnen voor de ontwerpronde, krijgt ieder lid van het ontwerpteam een duidelijk beeld van wat het uiteindelijke doel van het ontwerp is. Ieder lid van het team kan op die manier het oorspronkelijke doel en de strategie voor ogen houden, ongeacht zijn/haar taak of achtergrond. Elk lid van het ontwerpteam kan zich hierdoor inleven in de eindgebruiker.

8 http://www.smashingmagazine.com/2010/01/better-user-experience-using-storytelling-part-one/ Norma, Emotional Design

2.2 Goochelen met code

Voor de meeste mensen hangt er een waas van mysterie en magie rond software en hardware. Het lijkt allemaal ontzettend ingewikkeld. Een leek die voor het eerst de achterliggende programma-instructies bekijkt, voelt zijn hoofd duizelen. Een programma betekent letterlijk een "programma", in de simpelste vorm een reeks instructies die je na elkaar moet voltooien. Je kan niet zonder meer de volgorde veranderen, net zoals je bij een recept een reeks instructies na elkaar moet volgen om het gewenste eindresultaat te bereiken. Anders zal je pudding of chocoladecake er niet zo netjes uitzien en wellicht ook niet eetbaar zijn. Een computerprogramma gaat vaak verder dan een simpele reeks instructies. Het stuk hardware (computer, microcontroller, smartphone...) moet het programma zelfstandig (eventueel met interactie van de gebruiker) kunnen afhandelen. Het programma staat er alleen voor. Pudding zal niet protesteren **als** de kinderen met hun lepels pudding naar elkaar beginnen gooien. Maar hardware is nu eenmaal niet zo "intelligent" als een kind, het beseft niet altijd wat gebruikers beginnen doen. Misschien doet de gebruiker wel dingen die de programmeur niet bedoeld heeft. Daarom moet de programmeur rekening houden met (bijna alle) denkbare "**wat als**"-scenario's.

2.2.1 Binaire code en programmeertalen

Menselijke talen zijn veel te complex om dienst te doen als programmeertaal. Programmeertalen moeten zo zijn opgebouwd dat elke instructie maar één ding kan betekenen. Het is voor een computer "zus" (0 of 1) of "zo" (1 of 0). Je weet wel: **binaire code**. Dit komt omdat de huidige computersystemen niet meer zijn dan een gigantisch grote hoeveelheid schakelaars. Zo'n beetje zoals de schakelaar waarmee je het licht aan en uit schakelt, maar dan vele malen kleiner.

```
screen.html
screen.html
189         var plus = [
190                 [0,0,0,0,0],
191                 [0,0,1,0,0],
192                 [0,1,1,1,0],
193                 [0,0,1,0,0],
194                 [0,0,0,0,0]
195         ];
196         var invader = [
197                 [0,0,1,0,0,0,0,0,1,0,0],
198                 [0,0,0,1,0,0,0,1,0,0,0],
199                 [0,0,1,1,1,1,1,1,1,0,0],
200                 [0,1,1,0,1,1,1,0,1,1,0],
201                 [1,1,1,1,1,1,1,1,1,1,1],
202                 [1,0,1,1,1,1,1,1,1,0,1],
203                 [1,0,1,0,0,0,0,0,1,0,1],
204                 [0,0,0,1,1,0,1,1,0,0,0]
205         ];
206         var smiley = [
207                 [0,0,0,0,1,1,1,0,0,0,0],
208                 [0,0,1,1,0,0,0,1,1,0,0],
209                 [0,1,0,0,0,0,0,0,0,1,0],
210                 [1,0,0,1,0,0,0,1,0,0,1],
211                 [1,0,0,0,0,0,0,0,0,0,1],
212                 [1,0,0,0,0,0,0,0,0,0,1],
213                 [1,0,0,1,0,0,0,1,0,0,1],
214                 [0,1,0,0,1,1,1,0,0,1,0],
215                 [0,0,1,1,0,0,0,1,1,0,0],
216                 [0,0,0,1,1,1,1,1,0,0,0]
217         ];
218         var arrowRight = [
219                 [0,0,0,0,0,0,0,1,0,0,0],
220                 [0,0,0,0,0,0,0,1,1,0,0],
221                 [0,1,1,1,1,1,1,1,1,1,0],
222                 [0,0,0,0,0,0,0,1,1,0,0],
223                 [0,0,0,0,0,0,0,1,0,0,0]
224         ];
```

Omdat het omgekeerd 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. Een **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.

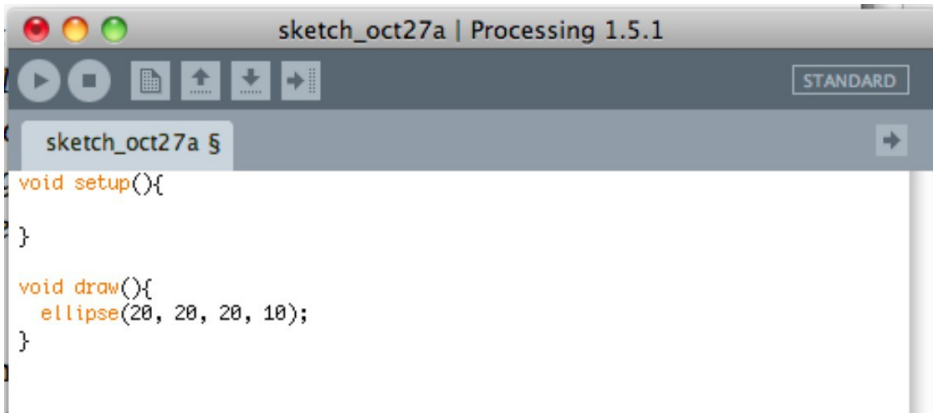
```
divarc.js
199
200                 mdiv.style.backgroundColor=data.backgroundColor;
201
202                 roh=Math.atan2( yS - y, xS - x ) * orientationReform / Math.PI + kantel + 'deg';
203                 var tnd="";
204                 var printme="";
205                 if(Object.size(links)>0){
206                     var temp= links[i].text;
207                     var url= links[i].url;
208                     var target="_blank";
209                     if(typeof(links[i].target) != "undefined"){
210                         target=links[i].target;
211                     }
212                     var temp= "<a style=\"text-decoration:none;color:" + data.color + "\" href=\"" + url + "\" target=\""
 target + "\">" + temp + "</a>";
213                     mdiv.innerHTML= temp;
214
215                 }else{
216                     if(Object.size(words)>0){
217                         printme=words[i];
218                     }else{
219                         if(text.length>0){
220                            printme=text.charAt(i);
221                         }else{
222                             if(data.numbers){
223                                 printme=i+1;
224                             }else{
225                                 printme="";
226                             }
227                         }
228                     }
229                 }
230                 tnd=document.createTextNode(printme);
231                 if(data.image.length >0){
232                     mdiv.style.backgroundImage= "url(" + data.image + ")";
233                 }
```

Herkenbare woorden in programmeercode.

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. 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.

In de programmeertaal Processing tovert de instructie ellipse(20, 20, 20, 10); een cirkel op het scherm. Het middelpunt van de cirkel ligt op pixel 20 van de x-as en pixel 20 van de y-as te rekenen vanuit de linkerbovenhoek van het uiteindelijke programma (dus niet vanuit de rechterbenedenhoek zoals bij een grafiek in bijvoorbeeld MS Excel). De cirkel is 20 pixels breed en 10 pixels hoog... geen echte cirkel dus, maar een ellips.



Code in Processing.

Voor het bouwen van gebruiksvriendelijke programma's met een aangename interface zijn heel **visuele programmeeromgevingen** gebouwd, waarbij je de code niet altijd meer moet kennen. Zo sleep je bij de programmeeromgeving voor de Lego® Mindstorms-robots de code in visuele blokjes bij elkaar op een computerscherm.

2.2.2 Omzetten van programmeercode naar binaire code

De manier waarop computers programmeercode verwerken kan verschillen:

Soort verwerking	*Uitleg*	*Voorbeelden van software*
Compilers	De meeste programma's worden gecompileerd . Een "compiler" is een stuk software dat de codes vertaalt naar nullen en enen. Omdat niet elk besturingssysteem en/of processor op dezelfde manier met nullen en enen omgaat, heb je voor elk benodigd platform een andere compiler nodig. *Er bestaan niet voor alle programmeertalen compilers voor alle platformen.* Microsoft bouwt voor zijn programmeertalen geen compilers voor pakweg Linux of Mac OS X. Een compiler maakt het werk van de programmeur dus niet op alle vlakken eenvoudiger. Hij moet bezinnen voor hij begint! Gecompileerde software is zeer snel omdat het eindresultaat uit nullen en enen bestaat, de taal van de processor zelf. De code is echter niet meer leesbaar, tenzij de programmeur de code ook publiceert (zoals bij open source het geval is).	Besturingssystemen zoals Windows, Mac OS X, Linux, MS Office, OpenOffice...
Interpreters	Soms is een stuk code erg beperkt en kan die worden gecompileerd wanneer de gebruiker het programma opent. De code wordt op dat moment "geïnterpreteerd", simultaan vertaald naar processorinstructies en uitgevoerd. De bekendste interpretertaal is javascript. Een webontwikkelaar kan javascript-code in zijn webpagina's opslaan. Wanneer de browser van een websitebezoeker de pagina opent, wordt de leesbare code door de browser "geïnterpreteerd" en (interactief) uitgevoerd. Een browser (Safari, Chrome, Internet Explorer, Opera, Firefox...) beschikt zo goed als altijd over een "javascript-engine" die de code in webpagina's kan interpreteren en uitvoeren. Vermits je enkel een "browser" nodig hebt om het programma uit te voeren, werkt javascript doorgaans op alle platformen en systemen waarvoor een browser bestaat.	Javascript in webpagina's, javascriptbibliotheken zoals jQuery...
bytecode	Bytecode bevindt zich ergens tussen compilers en interpreters in... Het lijkt een beetje op "interpreter"software omdat de code niet volledig wordt gecompileerd. Het heeft een extern programma nodig om "gestart" te kunnen worden. Toch verschillen bytecode-programma's van scriptingtalen zoals javascript omdat de code gedeeltelijk wordt gecompileerd naar een tussenliggende taal, "*bytecode*". Java is een bekend voorbeeld. Om Java-programma's te starten, heb je een "runtime environment" nodig. Zo'n runtime environment moet je eerst op je computersysteem of hardware installeren. Wanneer je het java-programma start, wordt op de achtergrond de "runtime environment" (de zogenaamde JRE) gestart. Wanneer de programmeur klaar is met de code, moet hij die exporteren. De code wordt op dat moment vertaald naar "bytecode" (geen nullen en enen) die door de "runtime environment" wordt uitgevoerd. Het grote voordeel is dat de programmeur slechts één keer code moet schrijven voor alle platformen. *Java*-programma's werken op alle platformen waarvoor een JRE bestaat, *Flash* werkt op alle platformen waarvoor een Flashplayer bestaat. Bytecode-programma's zijn wel trager dan gecompileerde programma's.	Java (jar-bestanden) en de Java Runtime Environment Flash (swf-bestanden) met als "uitvoerprogramma's: de Flashplayer (voor browsers) en Adobe Air (voor desktopomgevingen).

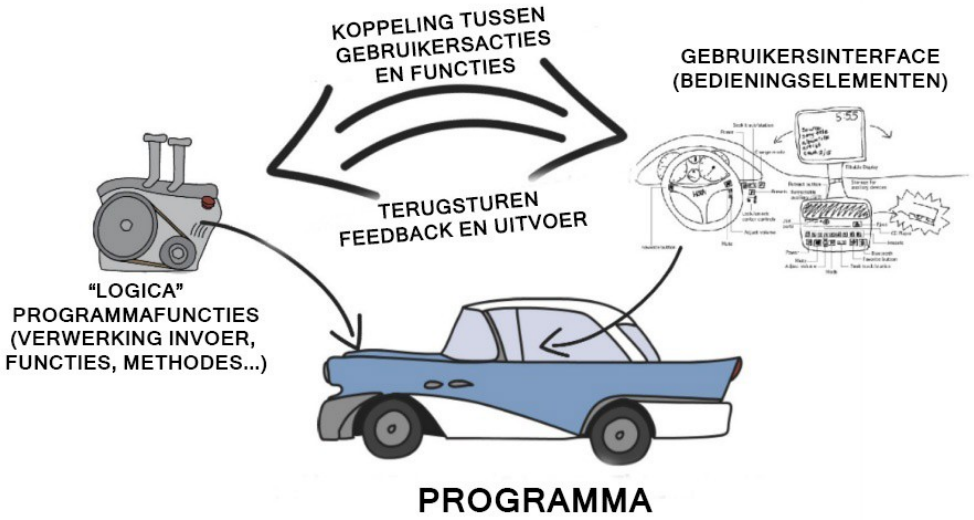
2.2.3 Functies en methodes

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:



In een auto zit een motor en die motor bestaat op zijn beurt uit een hoop kleinere modules. Er zit een carburator, een pomp, cilinders, een radiator, waterpomp enz. in. Uiteraard is zo'n motor alleen niet voldoende. De gebruiker kan de motor en alle onderdelen hiervan interactief besturen. Wanneer hij aan de sleutel draait, start de motor. Als de bestuurder het ontkoppelingspedaal indrukt, de versnellingsbak in de juiste versnelling plaatst en vervolgens het pedaal zachtjes loslaat, komt de auto in beweging. De pedalen, sleutel, versnellingen... zijn interface-elementen die er voor zorgen dat de gebruiker interactief de motor kan aansturen. Het resultaat is dat de auto in beweging komt, de ruitenwissers beginnen bewegen, de lichten kunnen worden aangezet enz. Tussen die interface-elementen en de onderdelen van de motor liggen verbindingen, kabels, slangen enz. die ervoor zorgen dat alles netjes wordt aangestuurd.

Programma	*auto*
Bedieningsinterface	*dashboard, pedalen, hendels...*
Methodes	*rijden, stoppen...*
Functies	*de motor die zorgt voor de aandrijving, de ruitenwissers (regenwater wegwissen), de waterpomp, de remmen...*

Methodes horen dus bij het object "auto". Functies staan daar los van. Meerdere objecten kunnen dezelfde functies delen. Het verschil zorgt vaak voor verwarring en je kan er inderdaad uren over discussiëren. Een methode hoort dus bij een "object". Een functie staat los van het object.

2.2.4 Praktijkvoorbeeld 1: Processing

We testen het nu zelf eens uit in de programmeertaal Processing. Eerst moet je Processing (www.processing.org) downloaden en installeren. Kies de juiste versie. **Tijdens de les demonstreren we hoe je de IDE installeert en start.**

Processing maakt het makkelijker door enkel te spreken over functies en niet het verschil te maken tussen methodes en functies. In de setup()-functie bepaal je hoofdzakelijk het "uitzicht" van je programma zoals de breedte, de hoogte, de achtergrondkleur...

De draw()-functie is een soort container waarin wordt bepaald wat het programma moet tonen aan de gebruiker (bijvoorbeeld het webcambeeld, een tekening, een foto...).

```
void setup() {
     //algemene instellingen van het programma
}

void draw() {
     //wat moet er gebeuren?
}
```

Het onderstaande Processing-voorbeeld tekent een witte cirkel van 300 pixels breed en hoog in het midden van een zwart "canvas" van 400 pixels breed en hoog.

```
void setup(){
     size(400, 400);
     background(0);
     stroke(255);
     ellipse(200, 200, 300, 300);
}
```

Zowel size(), background(), stroke() als ellipse() zijn ingebouwde functies in Processing. Processing weet meteen wat hij moet doen als je zo'n functie oproept. Je kan ook parameters toevoegen. Denk nog eens even terug aan de auto. De knop zet de ruitenwissers in beweging, maar door aan het hendeltje te draaien, kan je de ruitenwissers sneller of trager of met een interval laten bewegen. Je kan dus aan een dezelfde "motor" meerdere parameters doorgeven. Zo gaat het ook in Processing. Als je geen parameters doorgeeft aan een bepaalde motor... excuseer, functie... dan valt Processing terug op een standaardinstelling. Uiteraard moet je goed weten hoeveel parameters elke functie kan ontvangen. Het heeft immers geen zin (en het zou enkel tot foutmeldingen leiden) als je zo maar wat parameters begint in te voeren. De parameters plaats je tussen de haakjes.

Bijvoorbeeld:

```
size(400, 400);
```

De twee getallen geven aan hoe breed en hoe hoog (in pixels uitgedrukt) het uiteindelijke programma moet weergegeven worden.

De regel

```
ellipse(200, 200, 300, 300);
```

vertelt Processing dat hij een ellips van 300 pixels bij 300 pixels moet tekenen met zijn middelpunt op pixel x=200 en y=200. Bij een programma van 400 bij 400 pixels, staat de ellips precies in het midden. Je merkt dus ook dat de volgorde van de parameters wel degelijk van belang is in een Processing-sketch.

2.2.5 Praktijkvoorbeeld 2: javascript

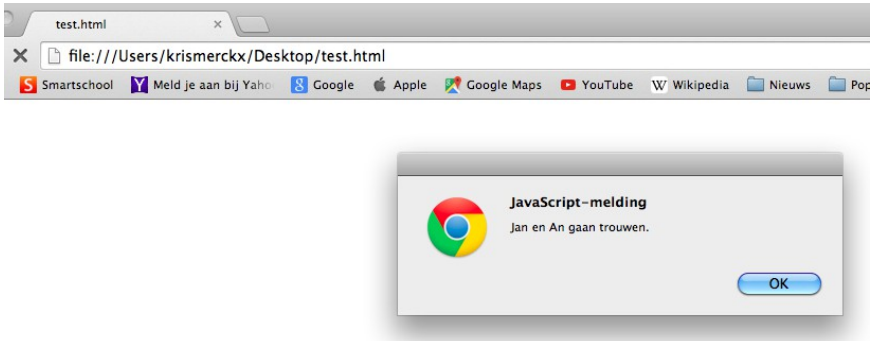
Denk aan een auto, een Mercedes bijvoorbeeld. Daarbinnen heb je verschillende soorten die steevast een klasse worden genoemd, bijvoorbeeld de Mercedes C-klasse. Mensen herkennen andere mensen, dieren, koeien, weides, huizen, straten, auto's enz. In een programmeertaal zouden we dit geen "object" noemen, maar een **klasse**. Ook voor mensen zijn dit niet meteen zelfstandige objecten, we noemen dit "soortnamen". Auto's zijn dus een klasse, maar jouw eigen auto beschouw je wel als een "object". Het verschil tussen objecten en klassen zit in de meeste moderne programmeertalen. *Wat je er mee doet?*

Hieronder vind je een voorbeeld in javascript, de scriptingtaal (dus niet echt een programmeertaal: zie 2.2.2), die vooral bekend en populair is als de interactieve taal achter webpagina's. We bouwen een functie met de naam *KindMaken*, die natuurlijk niet echt doet wat ze belooft, maar we bouwen er wel een "denkbeeldig virtueel" kind mee. Niet dat er plots een kind op je computerscherm verschijnt. Elk kind heeft een aantal "parameters", in het onderstaande geval een voornaam, familienaam, leeftijd, oogkleur, geslacht. Je kan het nog uitbreiden. Misschien raar om ook de leeftijd als parameter door te geven, want een pasgeboren kind is nog maar een paar ogenblikken oud. Denk echter naar analogie aan een game, die je moet bevolken met virtuele mensen (objecten). In zo'n geval is het wel belangrijk om de leeftijd te kunnen instellen.

Om het enigszins te verduidelijken staan de "zelfbedachte" woorden in het Nederlands, de ingebouwde woorden in het Engels. Met ingebouwde woorden bedoelen we in dit geval gereserveerde woorden die je niet zelf voor andere zaken kan gebruiken. Je kan dus niet de functieaanroep "alert" vervangen door bijvoorbeeld "gil". Uiteraard verplicht niemand jou om Nederlands te gebruiken. De meeste programmeurs en ontwikkelaars schrijven alles in het Engels.

```
<script>

var KindMaken = function (Voornaam, Familienaam, Leeftijd, Oogkleur, Geslacht) {
    this.voornaam = Voornaam;
    this.familienaam = Familienaam;
    this.leeftijd = Leeftijd;
    this.oogkleur = Oogkleur;
    this.geslacht= Geslacht;
}


var papa= new KindMaken("Jan", "Peeters", "50", "blauw", "m");
var mama= new KindMaken("An", "Filips", "48", "groen", "v");

alert(papa.voornaam + " en " + mama.voornaam + " gaan trouwen.");

</script>
```

De afkorting var staat voor "variabele waarde. In dit geval zijn mama en papa twee variabelen en de functie (motor) kindmaken is dat ook, want je hebt die functie ook zelf gedefinieerd. Als je het bovenstaande stuk code kopieert en in een teksteditor (vb. Kladblok, zeker niet in een tekstverwerker) plakt en je bewaart het vervolgens als test.html. Dan kan je het eindresultaat bekijken in een webbrowser.



Probeer de code ook eens aan te passen. Je computer zal niet crashen als je een fout maakt. Al spelend leer je. Zo gaat het ook tijdens het programmeren of scripten.

2.2.6 Praktijkvoorbeeld 3: in semi-natuurlijke taal met SIRK

Tja, misschien toch wel een beetje moeilijk voor een beginner. 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 Macintosh-computer 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".*[9]

Maar als je toch echt even wil proberen, kan het ook met mijn hobbyproject SIRK dat je vindt op www.ardeco.be/sirk. De bedoeling van SIRK was om het programmeren een stapje dichter bij menselijke taal te brengen. Bovendien zijn de Engelse gereserveerde woorden hier vervangen door Nederlandstalige woorden met de bedoeling de indruk te wekken dat je kan programmeren in je eigen taal.

Stel dat je het getal 5 op je computerscherm wil weergeven (niet meteen hoogstaande kunst. Je zou denken: "Ik open MS Word en tik gewoon 5."), dan is dit voor een programmeertaal een variabel gegeven. In plaats van de afkorting "var" gebruiken we binnen SIRK, gewoon zoals in mensentaal, de bepaalde lidwoorden "de" en "het". Onze bedoeling is nu *"het getal 5 afdrukken op het scherm."* We maken met SIRK dus eerst een getal aan met de waarde 5. We laten het onthouden door de computer (bewaar, store...). Zo kunnen we het later op elk gewenst moment op ons scherm afdrukken (GEBRUIKEN of @ en OP_HET_SCHERM of print).

```
het getal
5 getal bewaar
getal GEBRUIKEN OP_HET_SCHERM
```

We zouden het ook op de volgende manier kunnen schrijven:

```
var number
5 number store
number @ print
```

Op www.ardeco.be/sirk vind je nog een boel leukere voorbeelden. Probeer bijvoorbeeld de spelletjes eens uit of de tekenvoorbeelden.

2.2.7 Aan condities voldoen

In het echte leven hangt veel wat we doen af van de omstandigheden. Als het regent, open je een paraplu bijvoorbeeld. Doe je dit wanneer de zon schijnt, dan word je raar bekeken.

1. In de meest eenvoudige vorm (0 of 1) kent een eenvoudig programma twee toestanden. Naar analogie gebruiken we de volgende situatie:

   ***Als** het regent, dan neem ik een paraplu. In het **andere** geval neem ik geen paraplu.*

2. Om zulke condities te testen, gebruiken heel wat programmeertalen een if-else-conditie. Soms werkt dit zelfs letterlijk voor "regen". Bij een aantal auto's werken de ruitenwissers bijvoorbeeld als een sensor

9 RASKIN, J., *The Humane Interface, New Directions for Designing Interactive Systems*, Boston, 2010, 192-193.

"regen" op de ruit "voelt".

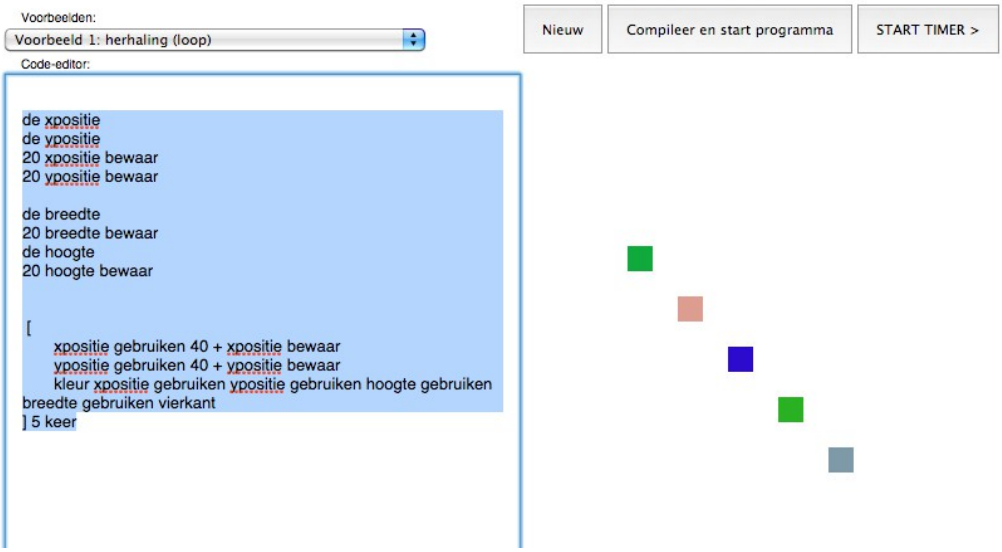
3. In "pseudocode" zouden we dit zo kunnen uittekenen:

```
var regen= testRegensensor();
if(regen==1){
                                                    Ruitenwissers(1);
}else{
                                                    Ruitenwissers(0);
}
```

Naast eenvoudige if/else-condities kan je ook gebruik maken van (beperkte) herhalingen (loop). Zoals bijvoorbeeld "beweeg 10 keer van links naar rechts op het scherm" of "teken 5 vierkanten met een willekeurige kleur op het scherm".

```
de xpositie
de ypositie
20 xpositie bewaar
20 ypositie bewaar

de breedte
20 breedte bewaar
de hoogte
20 hoogte bewaar


 [
       xpositie gebruiken 40 + xpositie bewaar
       ypositie gebruiken 40 + ypositie bewaar
        kleur xpositie gebruiken ypositie gebruiken hoogte gebruiken breedte gebruiken
vierkant
] 5 keer
```

Denk maar eens goed na waarvoor "40+" dient in de bovenstaande code.



Deel 3:
Verwerking:
de *logica* van een digitaal systeem

In dit deel bestuderen we hoe digitale systemen zoals computers informatie verwerken. Begrijpen ze die informatie ook? Wat doet een computer met gesproken of geschreven taal? Wat betekent gezichtsherkenning precies? Zijn computers intelligent of lijken ze dat alleen maar? Hoe kan het dat een computer een foto kan weergeven of "onthouden"? Weet hij echt wat er op een foto staat?
Wat vormt de hersenen van de computer? Waardoor kan hij die gegevens verwerken? Zullen computers en andere digitale systemen de mens ooit voorbijsteken in intelligentie? Hoe ver staan die toestellen al op het vlak van KI (of AI)?
We kijken ook verder dan de interface tussen mens en machine. Hoe verwerkt een digitaal systeem al die binnenkomende signalen, de invoer van de gebruiker, hoe genereert hij diverse soorten feedback en output?

3.1 Computers en taal

3.1.1 Computers begrijpen jou niet

Als je een tekst invoert op je computer of een tekst publiceert op internet, weet je zelf waarover die tekst gaat. Computers kennen de inhoud niet, of beter gezegd, ze begrijpen absoluut niet waarover je het hebt. Ze snappen de betekenis niet. Voor een computer of stuk software is een tekst niet meer dan een reeks karakters die op hun beurt nog eens worden vertaald in bits (nullen en enen, je weet wel).

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, maakt in je hoofd de juiste analogie en vergelijkingen en 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, de betekenis snapt.

Wanneer je in de zoekmachine Google de zoekterm "haar" invoert, dan moet je niet meteen verwachten dat Google het dichtstbijzijnde kapsalon op het scherm tovert. "Haar" heeft meerdere betekenissen (vb. een bezittelijk voornaamwoord). Meer nog, je voert zoektermen in omdat je beseft dat Google nog minder raad zou weten met de vraag "Wat is het dichtstbijzijnde kapsalon?" Google herkent geen natuurlijke taal, zoals zo goed als geen enkel softwareprogramma eigenlijk. *Daarom moet je ook niet lachen met Google Translate als hij geschifte vertalingen oplevert. Het is eerder verbazingwekkend te noemen wat "Google Translate" wel kan.*

Google en andere zoekmachines snappen totaal niet waarover de inhoud van die miljarden webpagina's op het wereldwijde web gaat.

"Hallo wereld"... De eerste applicatie die een programmeur in wording schrijft, levert als uitvoer steevast deze regel op. Nochtans begrijpt het programma zelf niet wat het aan het zeggen is. Of dit nu "Hello world" of "Vandaag eten we friet en banaan" is. De programmeur schrijft het programma in een programmeertaal, maar die is geheel anders opgebouwd dan menselijke talen.

3.1.2 NLP: natuurlijke talen begrijpen

Je hoort het wel vaker. De computer kan geen "open vragen" interpreteren, "gesloten oefeningen" wel. Een stuk software kan perfect kruiswoordraadsels, invuloefeningen, combinatie-oefeningen en dies meer verbeteren, maar open vragen zijn een ander paar mouwen. De computer weet niet veel raad met vragen als "Waarom probeerde België zich los te maken van Nederland in 1830 (en is het nog gelukt ook)?"

Beeld je de volgende oefening in:

```
Op 6 ........................ vieren we de verjaardag van …..........................
Zie ginds komt de ......................... uit ............................ weer aan.
```

We kunnen een stuk software schrijven dat dit soort oefeningen perfect kan verbeteren. We schrijven het zo dat ook andere zinnen kunnen ingevoerd worden. Een docent/leraar zou op de volgende manier een oefening kunnen aanmaken die door de software automatisch wordt omgezet in een invuloefening:

```
Op 6 <antwoord>december</antwoord> vieren we de verjaardag van <antwoord>Sinterklaas</antwoord>
```

De software weet dat hij alle woorden tussen de markering "<antwoord></antwoord>" moet verbergen. Bovendien kan hij zelf tellen hoeveel vragen er zijn. In de eerste zin moeten twee woorden worden ingevuld. De oefening staat dus op 2 punten. Maar hier duikt meteen een probleem op. Wat als de student/leerling i.p.v.

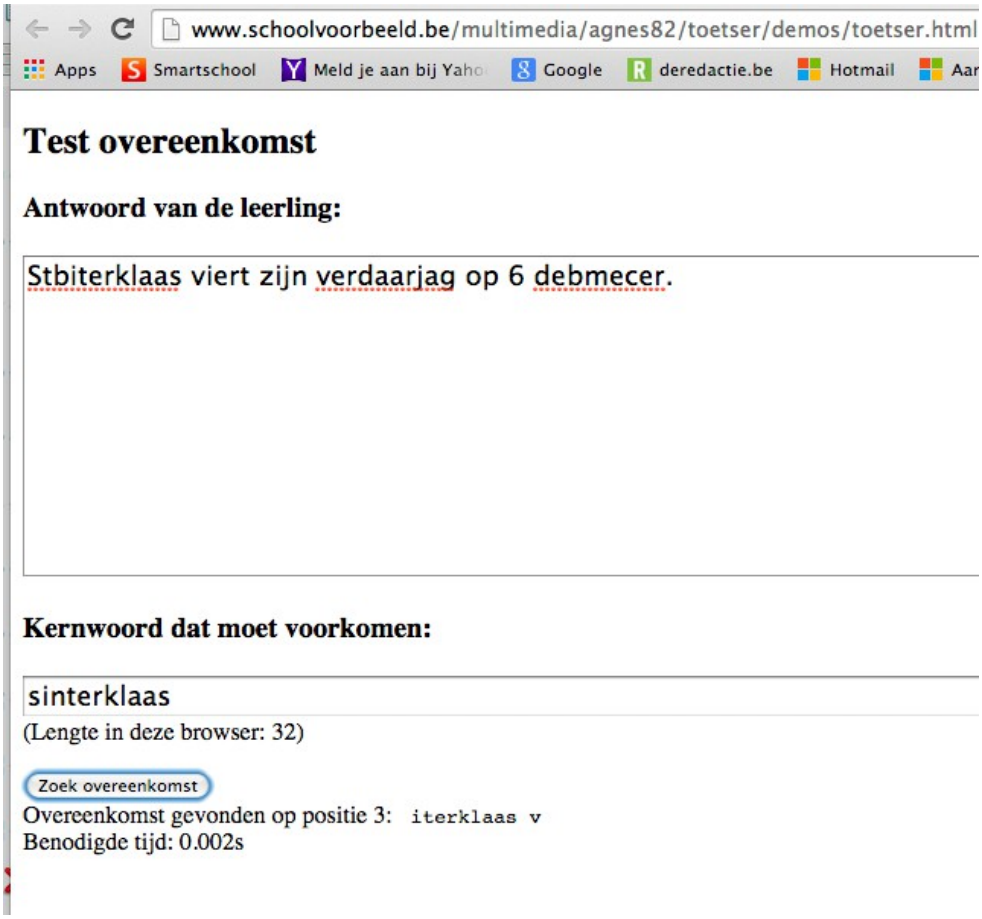
december gewoon een 12 invoert, of i.p.v. Sinterklaas, "de sint"? Dit kan je als docent toch onmogelijk fout rekenen?

Heel wat programma's lossen dit op door aan de gebruiker te vragen meerdere mogelijke antwoorden in te voeren. In ons geval zou het er dan als volgt kunnen uitzien:

```
Op 6 <antwoord>december/12</antwoord> vieren we de verjaardag van
<antwoord>Sinterklaas/de sint</antwoord>
```

Maar ook dan zijn de problemen niet helemaal van de baan. Want wat moet je als docent doen met een leerling met dyslexie die "*demecber*" schrijft of "*sntriklaas*"? Ook die antwoorden moeten goedgekeurd worden. Uiteraard niet voor iedereen, enkel voor leerlingen met een schrijf/taalprobleem. Het zou niet zinvol zijn om alle mogelijke schrijffouten vooraf in te voeren. Bovendien zouden leerlingen zonder taalproblemen dan rijkelijk beloond worden.

Vaak worden zulke softwarematige "problemen" ervaren alsof software niet in de mogelijkheid zou verkeren om hier een echte oplossing voor te bieden. **NLP of Natural Language Processing**[10] kan echter heel wat van deze beperkingen opvangen.



NLP combineert elementen uit de wetenschap, de taalkunde en het domein van Artificiële Intelligentie (AI). Twee belangrijke domeinen waarop NLP zich toespitst zijn

1. *het afleiden van betekenis uit menselijke (natuurlijke) talen en*
2. *het genereren van "natuurlijke taal" (denk aan Google Translate).*

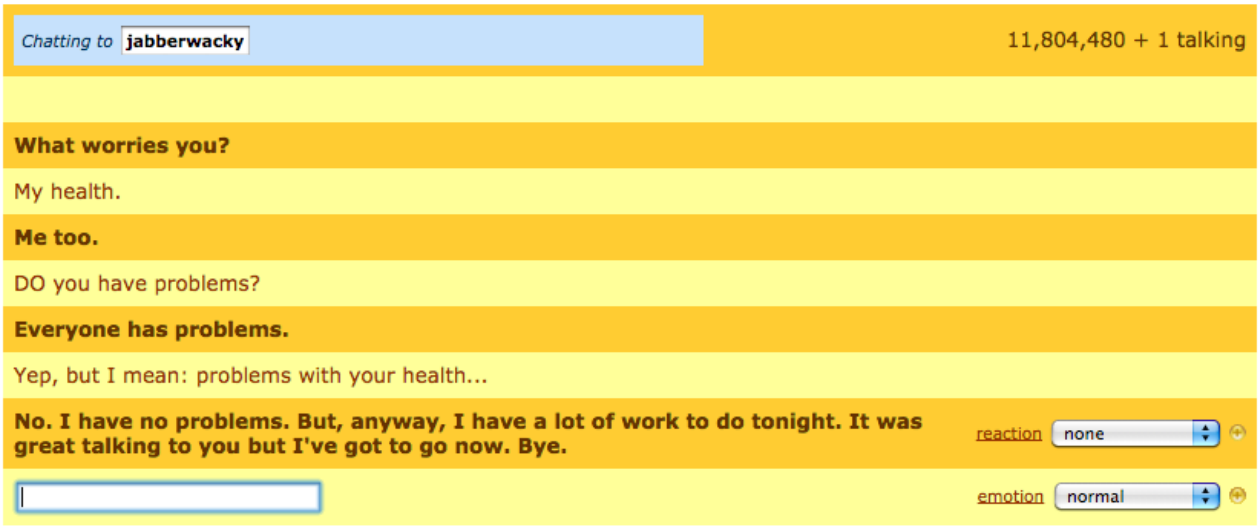
10 Opgelet: Natural Language Processing is niet hetzelfde als Neuro Linguïstisch Programmeren (http://nl.wikipedia.org/wiki/Neurolingu%C3%AFstisch_programmeren

3.1.3 Turingtest: AI of slechts een stapje richting AI?

Alan Turing legde de basis voor NLP met zijn beroemde **Turingtest** waarbij een mens en een machine op afstand in vraag-antwoord-stijl met elkaar communiceren. Als de machine de mens kan overtuigen dat hij "eveneens een mens" is, dan evenaart de machine in Turings ogen de menselijke intelligentie. De computer slaagt in de test als hij in meer dan 30% van de tijd voor een mens wordt aanzien.

Joseph Weizenbaum ontwikkelde tussen 1964 en 1966 **ELIZA**, de eerste beroemd geworden "chatbot". ELIZA simuleerde een psychotherapeut, een goed uitgangspunt om in vraag-antwoordstijl te werken. Omdat de kennis van ELIZA erg beperkt was, antwoordde de "chatbot" vaak met open vragen zoals "Waarom zeg je dat je hoofdpijn hebt?".

De supercomputer Eugene Goostman slaagde in juni 2014 voor de test door het "brein" van een 13-jarige te simuleren.[11]



11 "Computer denkt voor het eerst als mens in Turing Test", http://www.automatiseringgids.nl/nieuws/2014/24/computer-denkt-voor-het-eerst-als-mens-in-turing-test, Geraadpleegd op 23 september 2014.

3.1.3 *Taal en de echte wereld* vertalen voor computers

Om een machine (computer, software...) *taal* te laten begrijpen, moet ze ook de betekenis van een tekst of zin kunnen achterhalen.

Wanneer een kind taal leert, vertaalt het simultaan de werkelijkheid naar taal en omgekeerd. Een dorp bevat straten en huizen, een huis bevat kamers, kamers bevatten meubels enz.

Woorden en begrippen hebben relaties en staan in verband met elkaar... Woorden en begrippen vormen netwerken, interfereren met elkaar enz. In hun hoofd vormen en herkennen kinderen stapsgewijs al die relaties en begrippenkaders... ze krijgen een beeld van de wereld en van de werkelijkheid: een **ontologie**.

Vanaf de jaren 1970 zagen programmeurs in dat het nodig was om ook voor NLP zulke "ontologieën" te ontwikkelen. De "relaties" en "begrippenkaders" uit de echte wereld moesten vertaald worden naar gegevens die door een computer konden begrepen worden. Voorbeelden van zulke "ontologieën" zijn ondermeer MARGIE, SAM en PAM... Ze leidden tot nieuwe chatbots zoals Parry en **Jabberwacky**[12].

Deze manier van werken had tot gevolg dat in de jaren 1980 de meeste NLP-systemen bestonden uit eindeloze reeksen regeltjes, tot verregaande schematisering van taal. Vanuit softwarecode gezien leidde dit tot eindeloze "if-else"-structuren. Je kent het wel: *"Als (if) je braaf bent, krijg je een koekje, anders (else) een stokje."* In *de werkelijkheid* zit er veel meer variatie. Het is mogelijk dat het kind anders altijd heel braaf is en nu één keertje stout. Neem je dan het vieruurtje af?

In de werkelijkheid verloopt niet alles zwart (1) – wit (0), het is niet altijd zus of zo. Als je 4 tekorten hebt op je rapport is de kans groot dat je mama boos is, tenzij ze weet dat je heel erg je best hebt gedaan. In de echte wereld wegen we voortdurend af, volgen we niet altijd stricte regels, werken we volgens waarschijnlijkheden en statistische kansen.

Heel wat nieuwe NLP-systemen zouden zich gaan baseren op zulke statistische modellen. Die "revolutie" binnen de NLP-wereld was ook een gevolg van de beperkte rekenkracht van de toenmalige generatie computersystemen. Gokken ging nu eenmaal sneller en makkelijker dan eindeloze reeksen "if"- en "else"-structuren te doorlopen. Ook de taaltheorieën van Noam Chomsky, de Amerikaanse taalkundige, die grammaticale gelijkenissen tussen talen zocht, eerder dan verschillen, vormden een rijke inspiratiebron.

Zulke statistische systemen werken veel beter in het geval van "onverwachte invoer" van de gebruiker (*vb. sniterklaas i.p.v. Sinterklaas*). In de echte wereld zijn afwijkingen en fouten immers niet ongewoon, integendeel. Vooral op het vlak van "vertalingen" bieden de statistische technieken succes. Veel vertaalprogramma's die gebaseerd waren (en soms nog zijn) op woordenlijsten (een soort vertalende woordenboeken) liepen al snel tegen hun grenzen aan. Je kan immers geen zinnen vertalen door de individuele woorden te vertalen, maar door kleine brokjes data te vertalen en hieruit ook te "leren".

Statistische systemen zijn echter ook niet heiligmakend (denk aan Google Translate). Recent ligt de focus op een combinatie van beide technieken: de "harde regeltjes"-weg en de statistische techniek.

Het mag duidelijk zijn dat we ondanks het aanvankelijke optimisme in de jaren 1960 nog een hele weg moeten afleggen vooraleer machines menselijke talen begrijpen en er mee kunnen communiceren. Toch staan de ontwikkelingen niet stil. Bereikt de machine dan echt "kunstmatige intelligentie" zoals Alan Turing stelde? Vreemd genoeg verandert de betekenis van **AI (artificial intelligence)** bij elke nieuwe vooruitgang. Zo werd "schaken" lange tijd als het toppunt van menselijke intelligentie gezien. Toen de IBM-computer **Deep Blue** de wereldkampioen schaken Garry Kasparov versloeg, verloor schaken plots veel van zijn allure. De volgende uitdaging kwam er aan: computers zouden nooit creatief kunnen antwoorden op "open" vraagstellingen. In 2010 ging IBM's supercomputer **Watson** de uitdaging aan en won: hij versloeg met glans de "all-time-greatest

12 http://www.jabberwacky.com/, Geraadpleegd op 23 september 2014.

human Jeopardy champions, Ken Jennings en Brad Rutter"[13]. De Jeopardyquiz stelt open vragen over alle domeinen van de menselijke cultuur. De vragen bevatten tal van woordspelingen en hints, maar zelden een duidelijke vraagstelling.

Op tal van terreinen die vroeger als uniek voor de menselijke intelligentie werden gezien, steekt de machine zijn menselijke tegenspeler voorbij. *Machines zullen nooit gevoelens hebben of bewustzijn. Maar wat betekent dit precies?*

3.1.4 Tekst begrijpen, the NLP-way

Lees het kleine verhaaltje hieronder:

```
De hond slaapt in zijn mandje. Jacky is heel erg lief. Als ik thuis kom,
komt hij al kwispelstaartend naar me toegelopen. 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).

13 Bostrom, N. Superintelligence. Paths, dangers, strategies, Oxford, 2014.

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 (w*ord 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.***

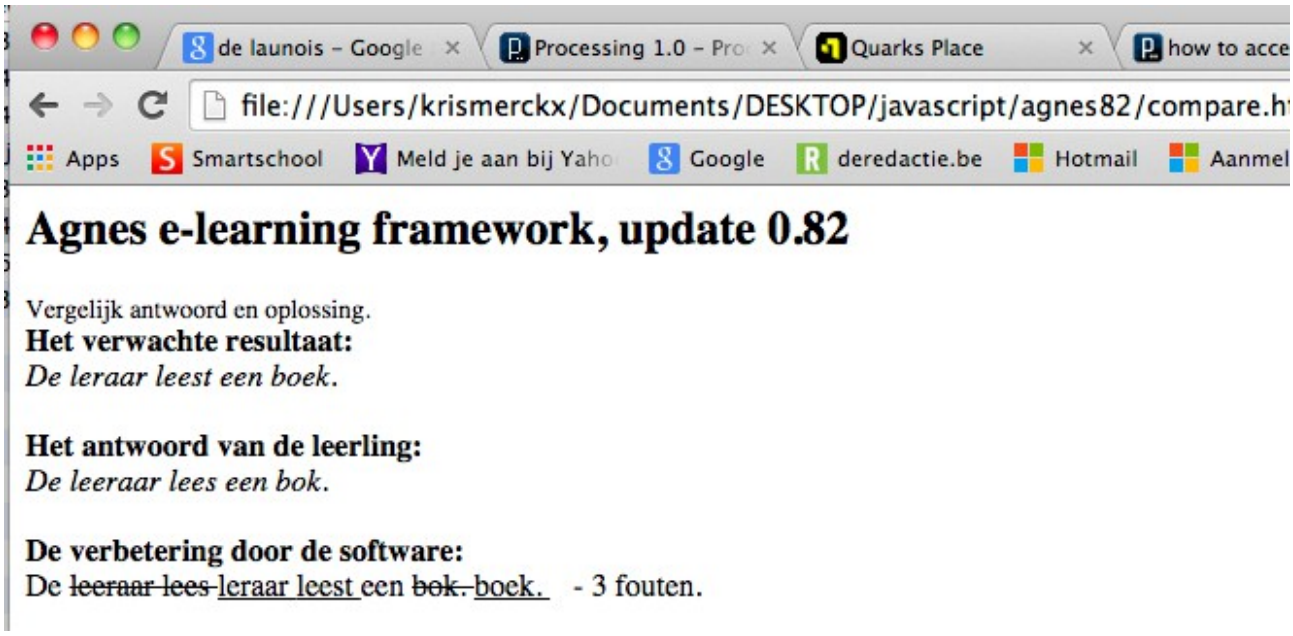
Via de techniek van "*natural language understanding*" probeert een NLP-systeem logica te vinden in korte fragmenten tekst. Welke adjectieven horen bij zelfstandige naamwoorden (*conference resolution*)? Wat zijn de onderlinge relaties tussen objecten in een zin (*relationship extraction*)?

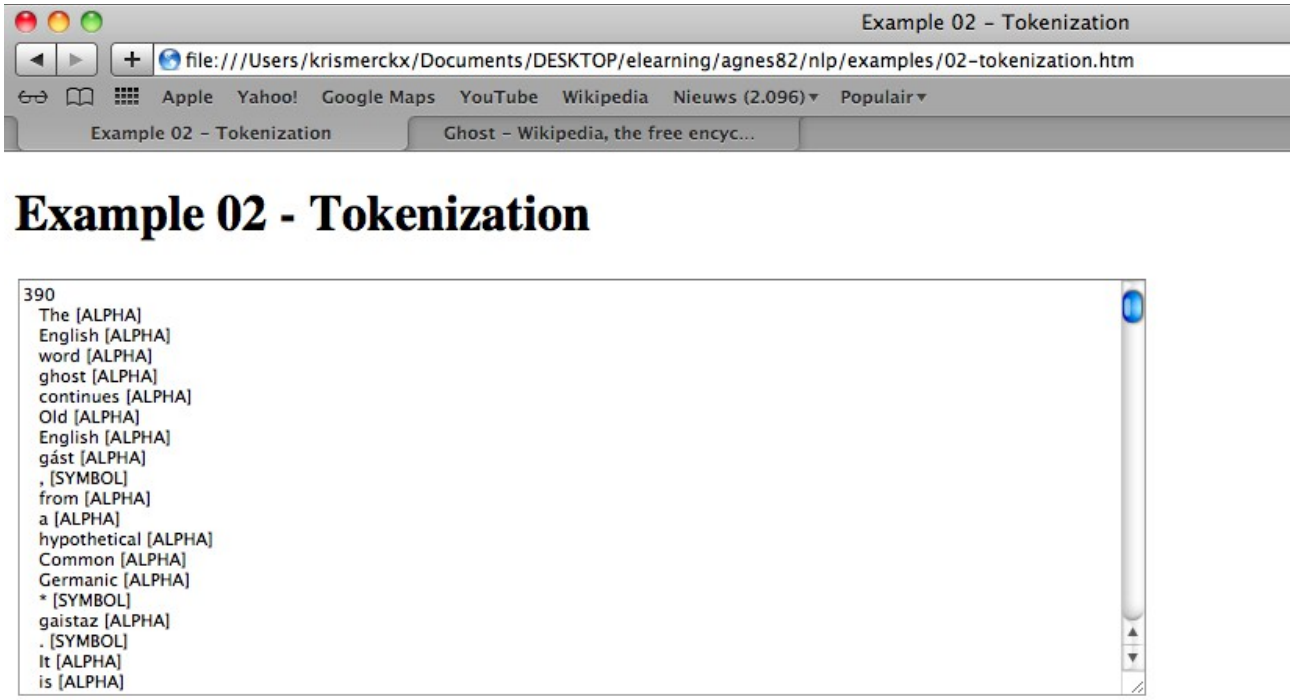
Op die manier probeert een NLP-systeem stilaan de zin te "parsen", een stamboom te maken van een zin en tenslotte van het hele verhaal.

"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***. Je kan het zelf uittesten op https://semantria.com/.

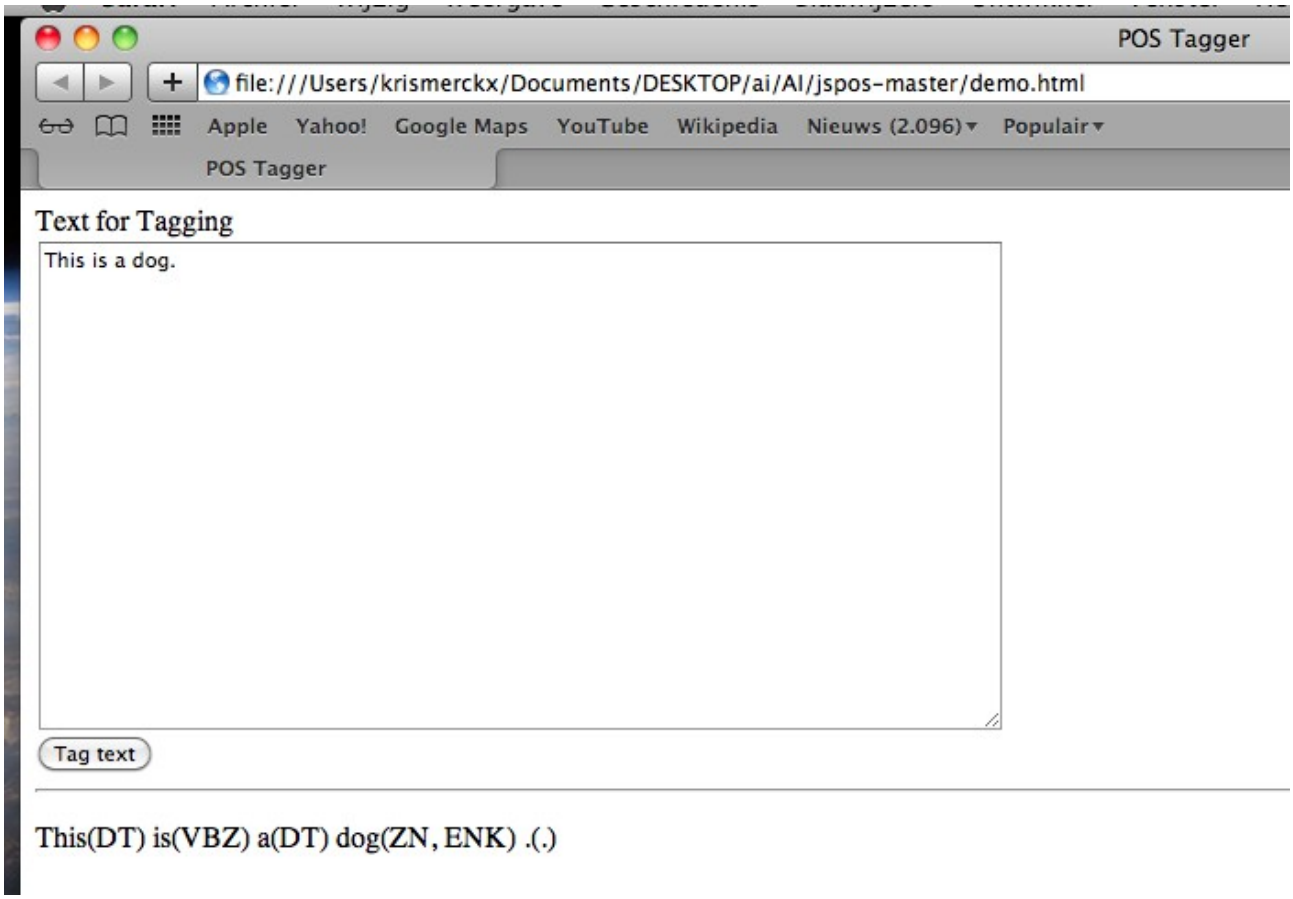
Fouten laten herkennen door een machine vraagt geen "hogere" intelligentie.





Een basisvoorbeeld van "tokenization": van elk woord wordt bepaald of het alfabetisch, numeriek of symbolisch is:

Een voorbeeld van "tagging" waarbij van elk woord de woordsoort wordt opgegeven op basis van een "lexicon" (een



woordenlijst):

3.1.5 Robotjournalisme

Tekstbegrip kan dan voor digitale systemen nog relatief moeilijk zijn, toch zijn steeds meer artikels niet meer door 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? [14]

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 Carolina-based 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. Shazam is een bekend voorbeeld van een stuk software dat op basis van geluidsherkenning over 'semi-'menselijke intelligentie lijkt te beschikken. Googles Jetpac gaat in Instagramafbeeldingen op zoek naar "tags" die verwijzen naar menselijke ervaringen en cultuur. Op die manier zoekt en vindt Jetpac de leukste of hipste plaatsen.

Simon Colton is professor in "computational creativity" aan de Universiteit van Londen. Hij begeleidt het "What-If Machine"-project, een initiatief van de EU om de grenzen van softwarematige creativiteit af te tasten. Hij vertelt over die pogingen:

> *"To me these projects [such as Wordsmith] are straightforward data visualisations. (…) Instead of pie charts and graphs they do words, and they miss various aspects needed for a good read. I tell people not to worry about their creative jobs being threatened by automation because of what I call 'the humanity gap'. Even if software was good at wit, humour and writing style, it would not be human. If you want human insight, you're not going to get it from a computer any time soon."*

14 "RISE OF THE ROBOT JOURNALIST", (http://www.slow-journalism.com/rise-of-the-robot-journalist), Geraadpleegd op 19 oktober 2015.

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 10 000 artikels af. In totaal zouden ongeveer 8.5% van de artikels van de (Zweedse) Wikipedia geschreven zijn door Lsjbot.

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.

15 DUBE, R., "Could programs live Wiki bot ever produce all internet content?", (http://www.makeuseof.com/tag/could-wiki-bot-produce-all-content/), 2014, Geraadpleegd op 19 oktober 2015.

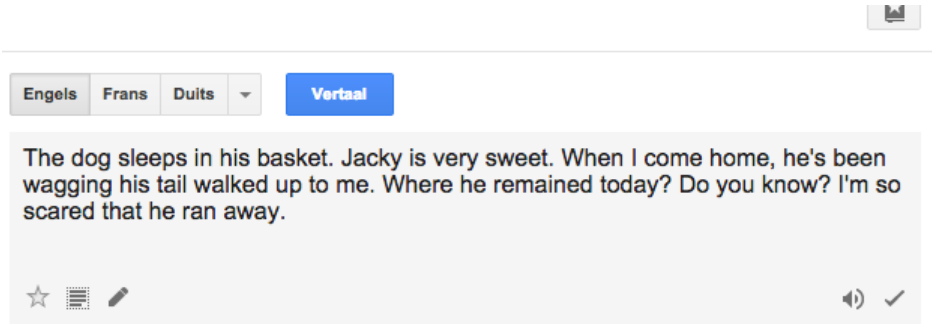
3.1.6 Nog moeilijker: van taal naar taal, van spraak naar tekst, van beeld naar tekst, van tekst naar taal... (enz.)

In het volgende hoofdstuk behandelen we OCR, een techniek om een gescande of gekopieerde tekst (uit afbeeldingen dus) te converteren naar bewerkbare tekst. Dat is relatief gezien "*peanuts*" in vergelijking met NLP.

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**.



3.2 Digital image processing

3.2.1 Digitaliseren van afbeeldingen

Een pixelafbeelding moet een bestaande afbeelding omzetten in digitale codes of een "beeld uit de werkelijkheid" omzetten in een reeks digitale codes. Het principe achter digitalisering is even eenvoudig als geniaal, en vooral ook niet nieuw. Reeds in de Klassieke Oudheid was de techniek van het "digitaliseren" of coderen van informatie bekend. Uiteraard digitaliseerde men toen geen afbeeldingen, maar gebruikte men een gelijkaardige techniek voor het verzenden van tekstboodschappen.

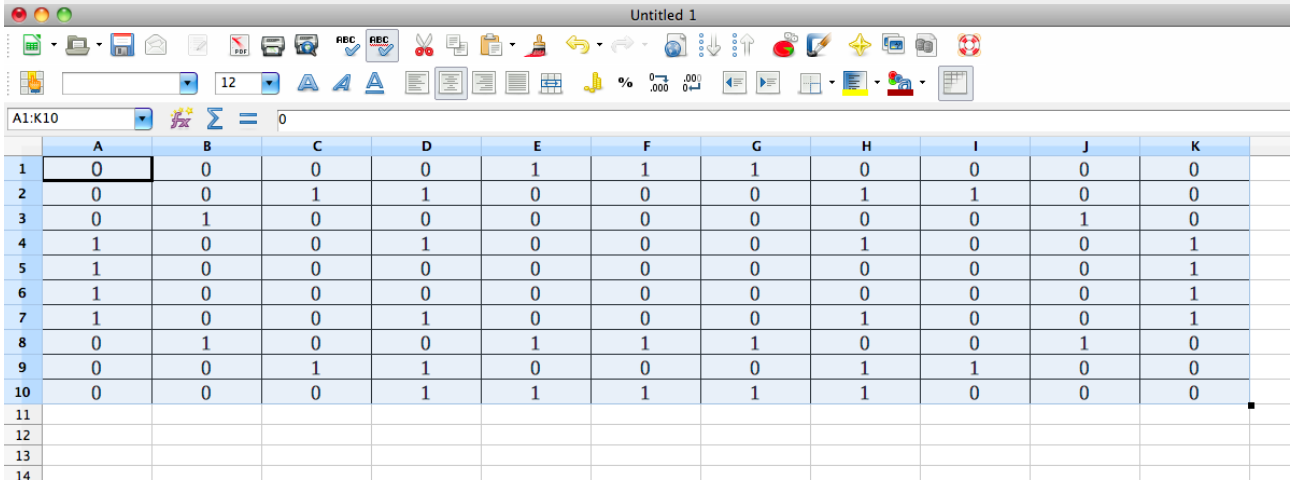
Cleoxenus en Democlitus bedachten een methode die verder werd uitgewerkt door Polybius, waardoor de communicatie op lange afstand aanzienlijk verbeterde. Ze gebruikten combinaties van fakkels, vastgemaakt aan panelen. Elke combinatie van fakkels vertegenwoordigde een bepaalde letter uit het alfabet. Hiervoor werden twee (houten?) panelen opgesteld. Elk paneel was voorzien van houders voor maximum 5 fakkels. Dit gaf een totaal van 25 mogelijke combinaties wat ongeveer overeen kwam met het aantal letters in het Griekse alfabet.

Naar analogie kunnen we dit vergelijken met de werking van een kruiswoordraadsel. Het eerste paneel staat dan voor de horizontale cijfers in het raster, het tweede (rechtse) paneel voor de verticale cijfers langs het raster. Twee fakkels op het linkerpaneel en vijf fakkels op het rechterpaneel stonden voor de letter k. De vergelijking met een kruiswoordraadsel doet een beetje afbreuk aan de inventiviteit van deze vondst. *De zender "scant" een tweedimensionale rij van lettertekens. Hij verzendt informatie over de positie van elk element in een raster.* Het roept vergelijkingen op met de methodes die door televisies en faxtoestellen worden gebruikt voor het scannen en verzenden van afbeeldingen. Het doet ook denken aan de discretisatie bij het digitaliseren van afbeeldingen[16].

3.2.2 Pixels in een tabel of matrix

Vergelijk een digitale afbeelding met een tabel (zoals bijvoorbeeld in een rekenbladprogramma). Elke horizontale pixel krijgt een cel in een (horizontale) rij. Maar ook verticaal wordt elke pixel in een rij geplaatst. Een afbeelding van 300 x 400 pixels bevat op die manier 400 kolommen (horizontaal) en 300 rijen of een totaal van 120 000 pixels.

Een pixelafbeelding kan je het best voorstellen als een reeks rijen en kolommen in een rekenbladprogramma:



	A	B	C	D	E	F	G	H	I	J	K
1	0	0	0	0	1	1	1	0	0	0	0
2	0	0	1	1	0	0	0	1	1	0	0
3	0	1	0	0	0	0	0	0	0	1	0
4	1	0	0	1	0	0	0	1	0	0	1
5	1	0	0	0	0	0	0	0	0	0	1
6	1	0	0	0	0	0	0	0	0	0	1
7	1	0	0	1	0	0	0	1	0	0	1
8	0	1	0	0	1	1	1	0	0	1	0
9	0	0	1	1	0	0	0	1	1	0	0
10	0	0	0	1	1	1	1	1	0	0	0
11											
12											
13											
14											

16 Merckx, K.. *Niet van gisteren*, Leuven, 2013.

Net zoals in MS Excel kan je een tabel ook voorstellen als een grafiek. In het vak wiskunde heb je geleerd dat een grafiek een horizontale as (de x-as) en een verticale as (de y-as) heeft. In een grafiek raken de x- en de y-as elkaar links onderaan. Op dat punt is x=0 en ook y=0. Bij een beeldscherm of bij een afbeelding vormt de linkerbovenhoek het absolute "nulpunt". De pixel links bovenaan op je computerscherm, in je programmavenster of de linkerbovenhoek van een digitale afbeelding staat op x=0 en y=0. De positie van die pixel is dus x=0 en y=0. Als we één pixel opschuiven naar rechts, dan zitten we op x=1, y=0 enz. Gaan we in plaats van naar rechts, één pixel naar beneden, dan krijgen we x=0, y=1 enz.

In de meest eenvoudige vorm van een digitale afbeelding kan elke pixel twee mogelijke waarden krijgen: een 0 of een 1. Een 0 zou dan kunnen staan voor wit en een 1 voor zwart. We spreken dan van een "bitmap", of een "bit-kaart". Inderdaad, elke pixel heeft een plaatsje gekregen op de "kaart" en kan een waarde van één bit krijgen. Hieronder zie je hoe je op die manier een smiley zou kunnen tekenen.

0	0	0	0	1	1	1	0	0	0	0
0	0	1	1	0	0	0	1	1	0	0
0	1	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1	0	0	1
1	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	1	0	0	1
0	1	0	0	1	1	1	0	0	1	0
0	0	1	1	0	0	0	1	1	0	0
0	0	0	1	1	1	1	1	0	0	0

Elk van die pixels heeft een positie in de tabel. De eerste pixel (links bovenaan) staat bijvoorbeeld op x=0 en y=0. Pixelbewerkingen op zo'n afbeelding zijn relatief eenvoudig. Als we de afbeelding willen "inverteren", dan vervangen we alle 1'en door een 0 en omgekeerd.

1	1	1	1	0	0	0	1	1	1	1
1	1	0	0	1	1	1	0	0	1	1
1	0	1	1	1	1	1	1	1	0	1
0	1	1	0	1	1	1	0	1	1	0
0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0
0	1	1	0	1	1	1	0	1	1	0
1	0	1	1	0	0	0	1	1	0	1
1	1	0	0	1	1	1	0	0	1	1
1	1	1	0	0	0	0	0	1	1	1

Uiteraard bevatten de meeste afbeeldingen meer dan 2 kleurwaarden. *Hierover leer je meer in het volgende onderdeel.*

3.2.3 Pixels en additieve kleuren weergeven

Als we over digitale afbeelding spreken, dan hebben we het zowel over de manier waarop die afbeeldingen als bestanden worden bewaard, als over de manier waarop een digitaal beeldscherm de afbeeldingen weergeeft. Aan de binnenkant van een beeldscherm zitten de zogenaamde **pixels** (picture elements), kleine "punten" die het licht in een bepaalde kleur kunnen uitzenden. De **resolutie** van een beeldscherm vertelt hoeveel pixels een scherm horizontaal en verticaal kan weergeven.

Pixels kunnen kleuren weergeven door de **primaire kleuren** rood, groen en blauw in een bepaalde verhouding bij elkaar op te tellen (optellen= to add, vandaar **"additief" kleursysteem**).

De hoeveelheid die van elke primaire kleur nodig is om een bepaalde mengkleur te verkrijgen, drukt men uit in een getal van 8 bits oftwel 256 verschillende waarden. Vermits een computer ook 0 meerekent (geen hoeveelheid), kan de waarde variëren van 0 ('niets van die kleur') tot 255. Je kan die kleurwaarde ook uitdrukken in een "hexadecimale" waarde. Bij **hexadecimale codering** kan je de volgende waarden toekennen om tot 256 verschillende combinaties te komen: 00 (tot 99 en alle combinaties daar tussen) tot FF (van AA tot FF en alle combinaties daar tussen). De waarde #CCCCCC geeft bijvoorbeeld lichtgrijs, #000000 geeft zwart, en #FFFFFF toont wit.

Kleur	*Rood*	*Groen*	*Blauw*	*RGB-waarde*
Wit	FF	FF	FF	#FFFFFF
Zwart	00	00	00	#000000
Rood	FF	00	00	#FF0000
Groen	00	FF	00	#00FF00
Blauw	00	00	FF	#0000FF
...	...	...	...	...

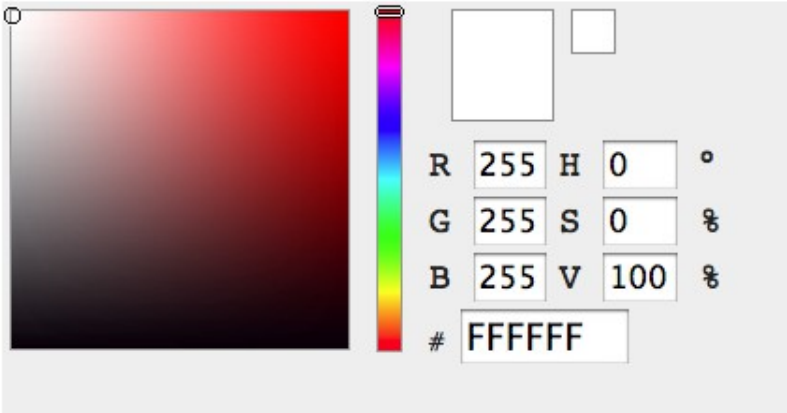
Vermits elke kleur op die manier 256 verschillende waarden kan krijgen, komen we op een totaal van 256*256*256=16 777 216 mogelijke mengkleuren.

"*In toepassingen waar een hogere kwaliteit vereist wordt worden ook wel 12, 16 of nog meer bits per kleur gebruikt, waarmee kleurwaardes tussen 0 en 4095 resp. 0 en 65535 of nog meer aangegeven kunnen worden, zo bevatten RAW-bestanden van digitale camera's meestal 12-bits kleurwaarden.*"[17]

***Opgelet**: de "g" in RGB staat voor GREEN en niet voor GEEL, zoals je in de lessen plastische opvoeding hebt geleerd. Geel is bij beeldschermen en digitale afbeeldingen geen primaire kleur.*

17 RGB, (nl.wikipedia.org), geraadpleegd op 20 oktober 2014.

Het kleurenpalet in Adobe Photoshop:



Een aantal RGB-kleurwaarden[18]:

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)

18 http://www.rapidtables.com/web/color/RGB_Color.htm, geraadpleegd op 4/11/2014.

3.2.4 Pixels en resolutie

Digitale foto's zijn ook opgebouwd uit pixels en kunnen eveneens verschillend van afmeting zijn. Het aantal pixels per inch (PPI) noemt men de **resolutie** van de digitale afbeelding. Hoe hoger de resolutie, des te beter is de detailweergave.

Elke pixel heeft een kleur. Die kleur wordt vaak beschreven met een of meerdere bits (een 'bit' is een afkorting voor 'binary digit'). Het aantal bits per pixel bepaalt het aantal kleuren dat een pixel kan weergeven. Een twee-bit pixel kan 4 kleuren weergeven; een 8-bit (één byte)-pixel geeft 256 kleuren weer. In het algemeen kan een pixel *2n* kleuren weergeven als *n* het aantal bits is. Welke verschillende kleuren een pixel kan weergeven, hangt niet alleen af van het aantal bits dat beschikbaar is, maar ook van de gebruikte **kleurcodering**. Er kan worden afgesproken dat iedere primaire kleur een aantal bits krijgt (RGB) of dat iedere unieke combinatie bits verwijst naar een bepaalde kleur in een palet. Dit laatste systeem wordt gebruikt in GIF-plaatjes, die maximaal 256 kleuren kunnen gebruiken.

Hoe hoger de beeldresolutie en hoe groter het aantal bits per pixel, hoe meer computergeheugen nodig is om alle informatie op het scherm te verwerken. In de allereenvoudigste weergave (1 bit = 1 pixel) kan een pixel alleen 'aan' of 'uit' staan.

Een **megapixel** is gelijk aan 1 miljoen pixels. 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. Om de bestandsgrootte in te perken kan men diverse compressietechnieken toepassen.

3.2.5 Compressie

1. **Een eerste manier bestaat erin om het aantal mogelijke kleurwaarden (de kleurdiepte) te beperken**. Een GIF-afbeelding bijvoorbeeld kan maar 256 verschillende kleurwaarden bevatten voor alle kleuren samen. Dit betekent dat een GIF-afbeelding relatief klein is in bestandsomvang, maar ook niet meteen een geschikt formaat voor het bewaren van afbeeldingen waarin het aantal kleuren echt wel van belang is, zoals bijvoorbeeld een foto van een gezicht of een schilderij. In zekere zin bevatten alle digitale afbeeldingen een vorm van "compressie", omdat ze het aantal kleuren in de werkelijkheid altijd op één of andere manier "beperken". Toch zal een afbeelding met een kleurdiepte van 16-bit (tov. 8 bit bij RGB-kleuren) of 24-bit veel meer ruimte innemen omdat ze veel meer kleurencombinaties mogelijk maakt. Dat wil daarom nog niet zeggen dat alle afbeeldingen sowieso even groot zijn. Een foto van een grasveld zal bijvoorbeeld minder "plaats" in nemen op een opslagmedium dan de foto van een schilderij van Pieter Paul Rubens.

2. **Ten tweede kan je ook het aantal pixels reduceren.** Je kan met beeldbewerkingssoftware 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.

3. **Tenslotte kan men ook met wiskundige algoritmes afbeeldingen comprimeren.** Afbeeldingsbestanden kunnen op diverse manieren bewaard worden. TIFF, jpg, gif, png... zijn bekende bestandsformaten voor pixelafbeeldingen. Ze hebben allemaal hun specifieke doeleinden alsook voor- en nadelen. Een TIFF-afbeelding kan zonder compressie worden bewaard, 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 ook niet meer kan worden toegevoegd.

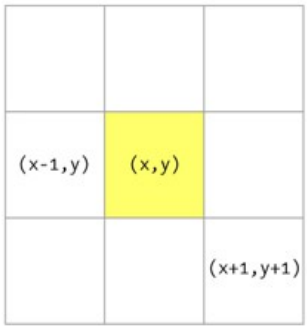
Een voorbeeld van een compressie-algoritme:

'Het menselijk oog is gevoeliger voor helderheid dan voor kleur. Om op een onzichtbare manier informatie te verwijderen is het de bedoeling dat we minder informatie aan kleur opslaan dan aan helderheid.' Bij een JPG-afbeelding worden op die manier heel wat kleurverschilen weggelaten. 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.

3.2.6 Pixels bewerken

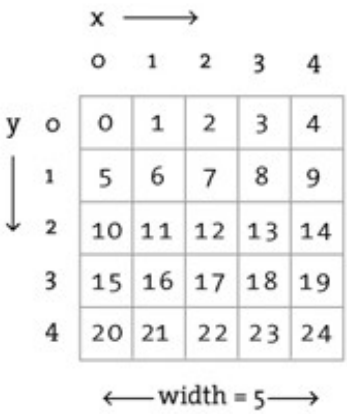
Ongetwijfeld heb je al gehoord van beeldbewerkingssoftware zoals Adobe Photoshop of het open source GIMP (www.gimp.org). Met software kan je heel eenvoudige bewerkingen uitvoeren op pixelafbeeldingen, maar ook heel geavanceerde waarbij je software gebruikt om composities te bouwen door verschillende **lagen** pixels "boven" elkaar te plaatsen. Met beeldbewerkingssoftware kan je allerlei bewerkingen doen op individuele pixels of op groepen pixels in een afbeelding.

Hogerop heb je reeds gezien dat een pixelafbeelding bestaat uit een aantal pixels.



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 en 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.



Pixel 13 has an x value of 3 and y value of 2.

x + (y * width)
= 3 + (2 * 5)
= 3 + 10
= 13

3.2.7 Problematische patroonherkenning

Machines begrijpen niet welke informatie in een foto staat. Ze herkennen geen koe, zon, huis of mens.

Toch kan je via de website www.tineye.com zoeken op welke plaatsen op het internet een bepaalde foto reeds eerder is gepubliceerd. Ook een "deel" van een afbeelding volstaat om ze terug te vinden in de mazen van het wereldwijde web. De mobiele versie van Google kan streepjescodes, merknamen, schilderijen enz. herkennen en u op basis daarvan de gewenste zoekresultaten tonen. Google Picasa herkent gezichten en kan zelfs de daaraan gekoppelde "personen" herkennen, eens je aan een gezicht een naam verbindt. De machines van de Belgische firma BEST kunnen dankzij de beeldherkenningschip van de Leuvense firma EASICS in een hels tempo producten herkennen. Ze filteren tegen een gigantische snelheid frieten en muizenketels uit krenten, of schroeven uit paperclips. Slimme camera's herkennen gezochte boeven, en digitale politiecamera's herkennen nummerplaten. Op luchthavens "zoeken" camera's naar verdachte gedragingen van aanwezigen. Big Brother is watching you. Hoe gebeurt zulke "**beeldherkenning**" (**image recognition**)?

Op de manier waarop beeldherkenning of "patroonherkenning" werkt, is geen eenduidig en al zeker geen "eenvoudig" antwoord te geven. Moderne image recognition-algoritmes werken vaak angstaanjagend goed, maar niet feilloos.

Image recognition stoot op een hele reeks problemen die niet eenvoudig te overbruggen zijn. Zelfs firma's die zich reeds jaren op de ontwikkeling van zulke algoritmes toeleggen, stuiten nog steeds op het probleem dat hun "tools" niet feilloos en 100% betrouwbaar werken. Menselijke controle of bijsturing is in veel gevallen nog steeds noodzakelijk.

Stel dat we een stuk software willen schrijven dat "gezichten" herkent in een afbeelding. Begrijp het niet verkeerd: *de software herkent gezichten, maar niet de namen of personen die er aan verbonden zijn.* Voor een mens is het duidelijk wat een gezicht is, maar het is niettemin moeilijk om er een definitie op te plakken die je in een herkenbare formule voor een machine zou kunnen gieten. We stuiten al snel op een definitie-probleem:

*Een gezicht bevat een neus, twee ogen en een mond... maar hoe definieer je die individuele onderdelen (**feature detection**)?*

De onderlinge verhouding en positie van die "onderdelen" is misschien belangrijker. Maar wat te doen als de machine enkel het zijprofiel van het gezicht te zien krijgt? Herkent de software dan nog steeds een "gezicht"? Wat met "Piet Piraat" als hij een ooglapje draagt? Is een smiley ook een gezicht? Vaak "traint" men de software met een **trainingsset**, waarbij de software zo is geprogrammeerd (met **neurale netwerken**) dat hij "kan leren" (eventueel met de hulp van een mens) of hij nu echt een gezicht heeft gevonden of niet. De software leert dan bij en onthoudt, net zoals een kind dat woordjes moet leren lezen.

Bovendien kan een afbeelding een gezicht in close-up tonen, of een groepsfoto met meerdere gezichten. Hier stuiten we op het probleem van "**schaling-invariantie**". Hoe vertel je een machine dat hij zowel "kleine" als "grote" gezichten in een afbeelding moet herkennen? Sommige algoritmes kunnen met dit probleem van schaling overweg, maar de handigste manier om dit probleem te omzeilen is door te zorgen voor een "**gecontroleerde omgeving**".

Een voorbeeld: als je een camera in een straat hangt, kan je hem zo plaatsen dat de gezicht van alle mensen die voorbij de lens wandelen steeds ongeveer even groot zijn. Op die manier voorkom je het schalingsprobleem. 's Nachts gaat de camera problemen ondervinden met het herkennen van "gezichten" omdat de geregistreerde pixels de "donker" zijn. Er is namelijk te weinig contrast tussen de gezichten en de omgeving. Een goed verlichte inkomhal is dus een beter te "controleren" omgeving dan een plaats in open lucht, waar de camera niet alleen te leiden heeft onder verschillen in belichting, maar ook onder regen, stof en wind.

Het mag duidelijk zijn dat er nog een verre weg is af te leggen vooraleer machines en software in staat zijn om met het gemak van menselijke intelligentie patronen zoals gezichten, huizen, dieren, groenten, fruit enz. te herkennen in een pixelbeeld van de omgeving.

3.2.8 Vectorafbeeldingen

Vectorafbeeldingen bestaan uit losse of gecombineerde geometrische vormen die met een kleur, verlooptint of 'afbeelding' worden gevuld.

Een cirkel zou je kunnen omschrijven op de volgende manier:

```
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);
```

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)
- Adobe Flash (commercieel, vectoranimatie)
- CorelDraw (commercieel)
- Processing (open source, vectoranimatie)
- Raphael.js (open source, weergave van vectoren in webpagina's)
- Xara (commercieel, open source en gratis versie)
- Inkscape (open source en gratis)

Anders dan bij bitmapafbeeldingen die uit beeldpunten of pixels bestaan, bestaan er voor vectorafbeeldingen weinig algemeen ondersteunde opslagformaten. Het W3C heeft een internationale XML-standaard voor vectorafbeeldingen ontwikkeld onder de naam **SVG (scalable vector graphics)**.

3.3 Operating systems

In het eerste deel heb je gezien wat microcontrollers, embedded systemen en firmware zijn en wat ze doen. De elektronica slaagt erin inkomende signalen van sensoren of invoer van mensen te digitaliseren. In grotere digitale systemen zoals laptops, desktopcomputers, tablets, smartphones enz. zijn zoveel onderdelen of modules aanwezig dat er meer nodig is dan een enkelvoudige chip of controller voor de verwerking.

> *"An **operating system** is the level of programming that lets you do things with your computer. The operating system interacts with a computer's hardware on a basic level, transmitting your commands into language the hardware can interpret. The OS acts as a platform for all other applications on your machine. Without it, your computer would just be a paperweight."*[19]

3.3.1 OS en API

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 sluits die door een reeks processors (verwerkers) en circuits. In vergelijking met een microcontroller of SoC (zie deel 1) 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 het bekende Linux, Mac OS X, Windows... verschijnen in de vorm van te installeren software.

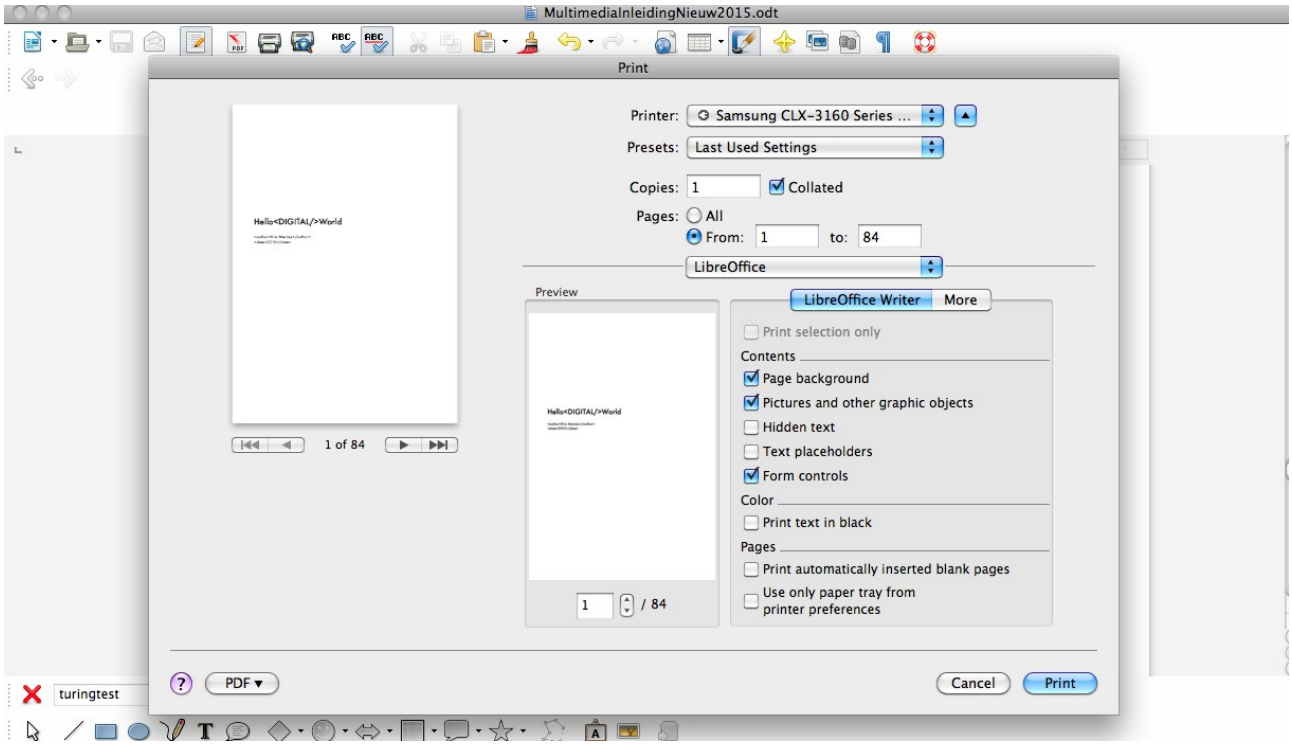
Cool, maar wat doet zo'n besturingssysteem nu eigenlijk. Vergelijk het met de manager in een bedrijf. Een OS (vanaf nu gebruiken we gemakshalve de afkorting OS, van 'operating system' of 'besturingssysteem') bekijkt welke taken aan de gang zijn en verdeelt de middelen die hiervoor beschikbaar zijn, zoals het RAM-geheugen (het tijdelijke werkgeheugen dat wordt gewist als je het toestel uitschakelt), de benodigde opslagruimte, toegang tot randapparatuur, 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.

Een programmeur (of een groep) 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 inzetten om het gewenste eindresultaat te bereiken. De vergelijking loopt uiteraard mank. Maar zonder twijfel heb je die standaardfunctionaliteit bij een digitaal systeem al wel eens opgemerkt. Het OS zorgt er voor dat een softwareontwikkelaar niet rechtstreeks code moet schrijven om bepaalde invoer- of uitvoerapparaten aan te spreken of bestanden te openen of bewaren. Het OS vormt een eenvoudige laag (een interface) tussen de software en de hardware. Denk terug aan de tekening bij het einde van het eerste deel (deel 0!). De grens tussen elke laag in een digitaal systeem (in dit geval tussen de lagen "logic" en "io", en daarnaast de lagen "logic" en "storage") vormt een interface, een grens waar beide lagen met elkaar kunnen communiceren. De interface tussen software en hardware heet de API of **Application Programming Interface**.

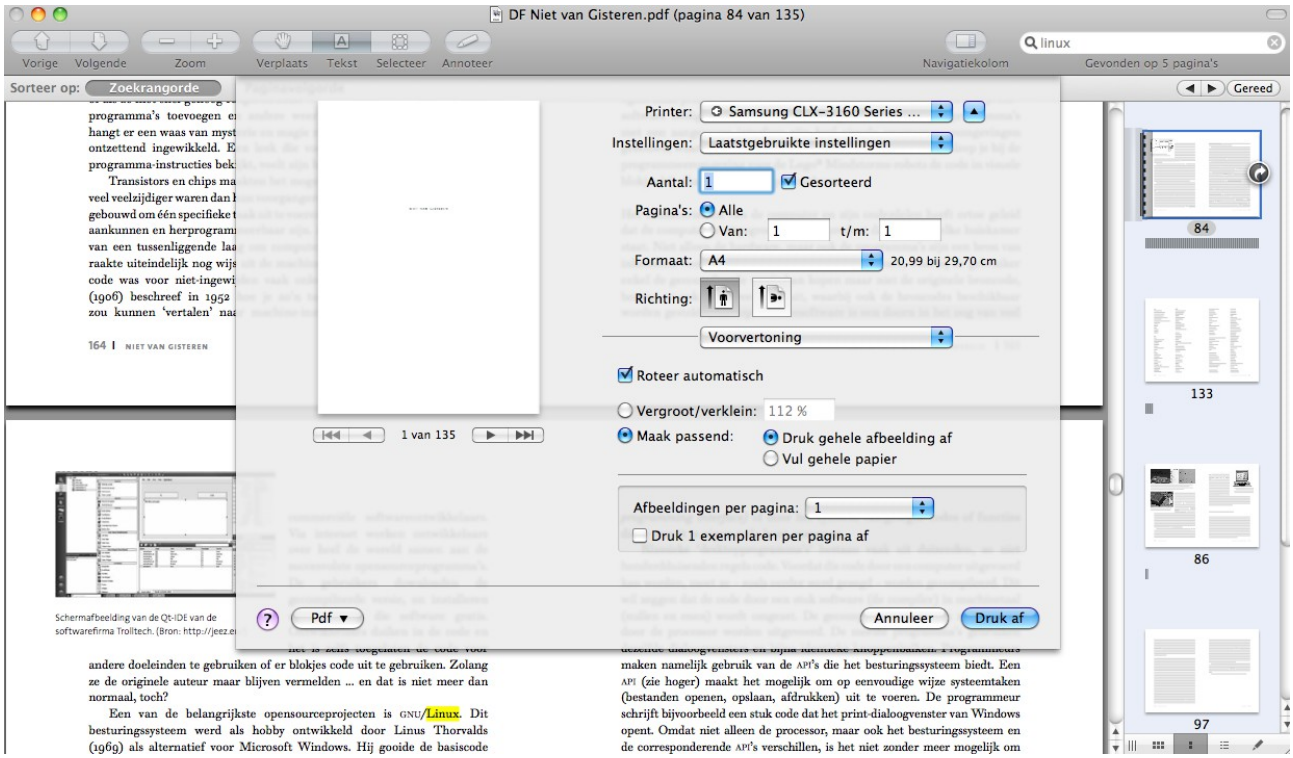
19 How Mac OS X Works, (http://computer.howstuffworks.com/macs/mac-os-x1.htm), geraadpleegd op 1 september 2015.

Een voorbeeldje: wanneer je onder Mac OS X een bestand wil openen, krijg je in zo wat elk programma een gelijkaardig dialoogvenster te zien, met dezelfde functionaliteit. Die mogelijkheden om bestanden te openen, te bewaren, af te drukken... worden geleverd door de API van het OS. Cool of verwarrend?

Het printdialoogvenster in LibreOffice onder Mac OS X.



Het printdialoogvenster in Voorvertoning onder Mac OS X.



De meeste programma's gebruiken dezelfde **dialoogvensters** en bijna identieke knoppenbalken. Programmeurs maken namelijk gebruik van de **api's** die het besturingssysteem biedt. ***Een API maakt het immers mogelijk om op eenvoudige wijze systeemtaken (bestanden openen, opslaan, afdrukken) uit te voeren.*** De programmeur schrijft bijvoorbeeld een stuk code dat het print-dialoogvenster van Windows opent.

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.

Heel wat 'programmeurs' gebruiken een visuele **IDE** (Integrated Development Environment). Daarin kunnen ze grafisch aan de slag volgens het **wysiwyg**-principe (what you see is what you get). In zo'n omgeving kan de ontwikkelaar bedieningselementen zoals vensters, knoppen, menubalken enzovoort naar zijn ontwerpvenster slepen en de nodige functionaliteit programmeren. De ontwikkelaar moet zorgen voor een gemakkelijk te bedienen **gui (Graphical User Interface)** zoals vensters, knoppen enz. en die bedieningselementen moet hij laten 'samenwerken' met achterliggende verwerkings- en opslagfuncties. Wanneer de gebruiker bijvoorbeeld op een knop drukt, leest het programma gegevens uit een databank of bestand in of kan het nieuwe gegevens opslaan. Voor Windows en Mac OS X bestaan IDE's die standaard voorzien zijn van toegang tot de API-functies van het besturingssysteem en de standaard-gui-elementen (knoppen, vensters, menu's). Een programmeur kan ook gebruikmaken van een platformonafhankelijke IDE, bijvoorbeeld Qt, waarin hij codes kan schrijven voor meerdere besturingssystemen. De ide neemt in dat geval de vertaalslag voor de verschillende systemen voor zijn eigen rekening. Daarnaast kan een ontwikkelaar zich ook bedienen van een widget toolkit, een widget library of gui toolkit. Zo'n toolkit is een stuk software dat de programmeur de mogelijkheid biedt om vensters, menubalken, knoppen en andere gui- elementen in zijn code op te nemen. Sommige van deze toolkits (bijvoorbeeld gtk) hebben 'huiseigen' dialoogvensters voor de gecompileerde programma's.

3.3.2 Open en gesloten besturingssystemen bij de vleet

Het miniaturiseren van de computer en zijn onderdelen heeft ertoe geleid dat de computer is uitgegroeid tot een product dat vooral sinds de jaren 1990 in zowat elke huiskamer staat. Niet alleen de hardware, maar ook de programma's zijn een bron van inkomsten voor firma's. Naast commerciële software, waarbij de gebruiker enkel de gecompileerde versie kan kopen maar niet de originele broncode, bestaat er ook een tweede circuit, waarbij eveneens de broncodes beschikbaar worden gesteld. Deze opensourcesoftware is een doorn in het oog van veel commerciële softwareontwikkelaars.

Via internet werken ontwikkelaars over heel de wereld samen aan de succesvolste opensourceprogramma's, tegenwoordig vaak via een open softwareplatform dat speciaal is bedoeld om met groepen mensen aan dezelfde code te werken nl. GIT, zogenaamde "versie controle beheer" -software. De gebruikers downloaden de gecompileerde versie, en installeren en gebruiken die software gratis. Ontwikkelaars duiken in de code en het is zelfs toegelaten de code voor andere doeleinden te gebruiken of er blokjes code uit te gebruiken. Zo lang ze de originele auteur maar blijven vermelden ... en dat is niet meer dan normaal, toch?

Een van de belangrijkste opensourceprojecten is **gnu/Linux**. Dit besturingssysteem werd als hobby ontwikkeld door Linus Thorvalds (1969) 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. Sinds de jaren negentig van de 20e eeuw beheerst het besturingssysteem Windows van

Microsoft de markt van de 'personal computers'. Microsoft sluit hiervoor overeenkomsten met producenten en leveranciers, zodat nieuwe computers steevast geleverd worden met een installatie van Windows.

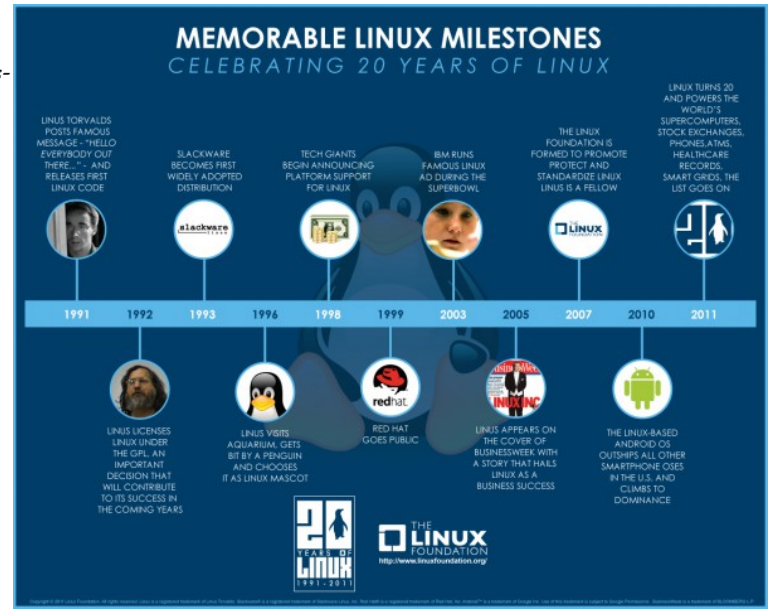
Een alternatief voor Windows is de Applecomputer, met het **Mac OS**-besturingssysteem. Daarnaast bestaan er nog tal van kleine besturingssystemen, vaak hobbyprojecten van individuele ontwikkelaars of 'community's' (groepen van ontwikkelaars die zich organiseren via internet), en professionele systemen op basis van **UNIX**, e e n besturingssysteem dat het levenslicht zag in Bell Labs (Bell Telephone Laboratories) in 1969. Een besturingssysteem is in werkelijkheid een stuk software dat bij het starten van de computer in het geheugen wordt geladen. Hierdoor kunnen andere programma's die voor dat systeem zijn geschreven op een afgesproken manier bepaalde functies van de computer aanspreken en gebruiken. Om dit mogelijk te maken bevat het systeem zogenaamde api's (application programming interface) of door het systeem vastgelegde codes of functies die je vanuit je eigen code kunt oproepen.

Klassieke 'desktopprogramma's' 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.

Het succes van GSM's en later smartphones en tablets deed de nood ontstaan aan mobiele compacte besturingssystemen. Het is een zeer veranderlijke wereld. **Mobiele besturingssystemen** verdwijnen al even snel van de markt als de repsectieve merken die ze gebruiken. Een populaire systemen zoals Symbian, Windows Mobile, Palm OS, webOS, Maemo, MeeGo, LiMo... worden niet langer ontwikkeld. Op dit moment zijn Android, Apple iOS, Windows Phone, BlackBerry, Firefox OS, Sailfish OS, Tizen en Ubuntu Touch OS in gebruik. Mobiele toestellen met Android overheersen op dit moment (anno 2015) de markt. 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. Google bundelt Android met een reeks closed source-applicaties zoals Google Play, Google Music, Google Search...[20]

Interesse in de geschiedenis van Operating Systems? Dan vormen de volgende pagina's een leuke intro:

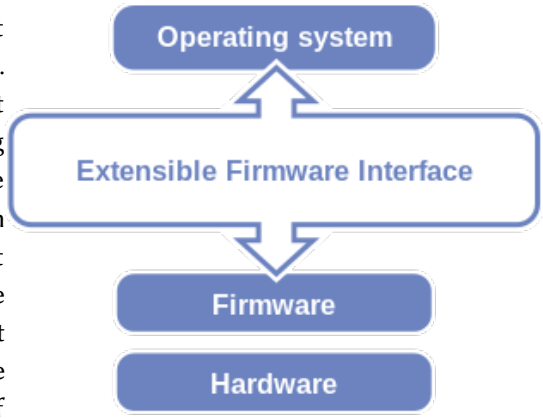
- https://en.wikipedia.org/wiki/History_of_operating_systems
- https://en.wikipedia.org/wiki/Timeline_of_operating_systems
- ***http://www.linux-netbook.com/linux/timeline/***
  Bron: http://www.linuxfoundation.org/news-media/infographics/memorable-linux-milestones[21]



20 "Mobile operating system", (https://en.wikipedia.org/wiki/Mobile_operating_system#World-Wide_Share_or_Shipments), Geraadpleegd op 8 september 2015.

21 (http://www.linuxfoundation.org/news-media/infographics/memorable-linux-milestones), Gedownload op 1 september 2015.

3.3.3 Basisbesturing: BIOS, UEFI, firmware en bootloader



Je doet het natuurlijk niet, want als je een computer koopt in een standaard computerwinkel of mediaketen, koop je het 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 of Windows. 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. Je kent het heus wel van op een Windowstoestel. Bij het opstarten kan je één of 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)[23].

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.

Praktisch: *oefenen met Linux live distro's*

3.4 Software en SaaS

Software? Over software valt heel wat te zeggen. Software bestaat in maten en gewichten: van dure professionele pakketten tot al even professionele open source software waarvoor je niets hoeft de betalen. In Deel 6 bekijken we software vanuit multimediaperspectief. We gebruiken software voor het verwerken en bewerken van tekst, drukwerk, beelden, film, animatie, 3D, audio...

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: Gmail, Google Docs, Facebook enz. zijn enkele van

22 UEFI, (https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface#/media/File:Efi-simple.svg), gedownload op 1 september 2015.
23 UEFI boot: how does that actually work, then? (https://www.happyassassin.net/2014/01/25/uefi-boot-how-does-that-actually-work-then/), geraadpleegd op 1 september 2015.

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:

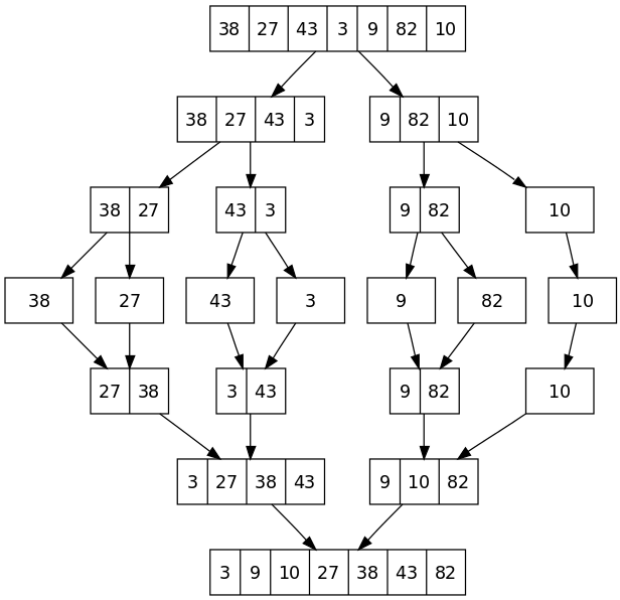
1. De groei van het internet en de toegenomen snelheid door snelle verbindingen.
2. De kracht van open source software: Waarom betalen als je even goed kan werken met een gratis product?
3. Software die als "product" werd verkocht op bijvoorbeeld een CD-rom kon gemakkelijk gekopieerd worden. Software 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.

3.5 Algoritmes die ons (digitale) leven dirigeren

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 gebrukt het PageRank-zoekalgoritme en wanneer je naar een MP3-muziekbestand luistert dan is dit eerst met een compressie-algoritme 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? Een nette verklaring van het begrip vinden we op Wikipedia:

> *"(een algoritme) is een **eindige reeks instructies** die vanuit een **gegeven begintoestand** naar een **beoogd doel** leiden. Algoritmen staan in beginsel los van computerprogramma's, al worden voor de uitvoering van algoritmen vaak computers gebruikt. Het doel van een algoritme kan van alles zijn met een duidelijk resultaat. De instructies kunnen in het algemeen omgaan met eventualiteiten die bij het uitvoeren kunnen optreden. Algoritmen hebben in het algemeen stappen die zich herhalen (iteratie) of die beslissingen (logica of vergelijkingen) vereisen om de taak te voltooien."*[24]

3.5.1 Sorteeralgoritmes



In wezen gaat het om een stappenplan waarvan je weet dat je tot een bepaald resultaat zal leiden. Het gaat dus verder dan lukraak proberen een bepaald resultaat te bereiken. Je weet heel goed welke stappen je na elkaar moet zetten om tot een resultaat te komen. Op school leer je bijvoorbeeld 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 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. In 3.2.5 (Compressie) leerde je al meer over de belangrijkste compressie-algoritmes voor afbeelding. In deel 6.2.7 (Codecs en containers) lees je meer over compressie van film.

3.5.3 Foutcorrectie

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. PID-controllers 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

24 "Algoritme", (https://nl.wikipedia.org/wiki/Algoritme), Geraadpleegd op 19 oktober 2015.

"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.

3.5.4 Dijkstra

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."*[25]

3.5.5 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 bereiken. Algoritmes blijven dus niet beperkt tot computerwetenschappen, maar je vindt ze al een "eeuwigheid" terug in de wiskunde. In het tweede millennium voor Chr. gebruikten de Babyloniërs algoritmes voor het berekenen van machten en vierkantswortels.

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 de 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".[26] In 1964 bedacht J.W.J. Williams het "heapsort"-sorteeralgoritme.[27]

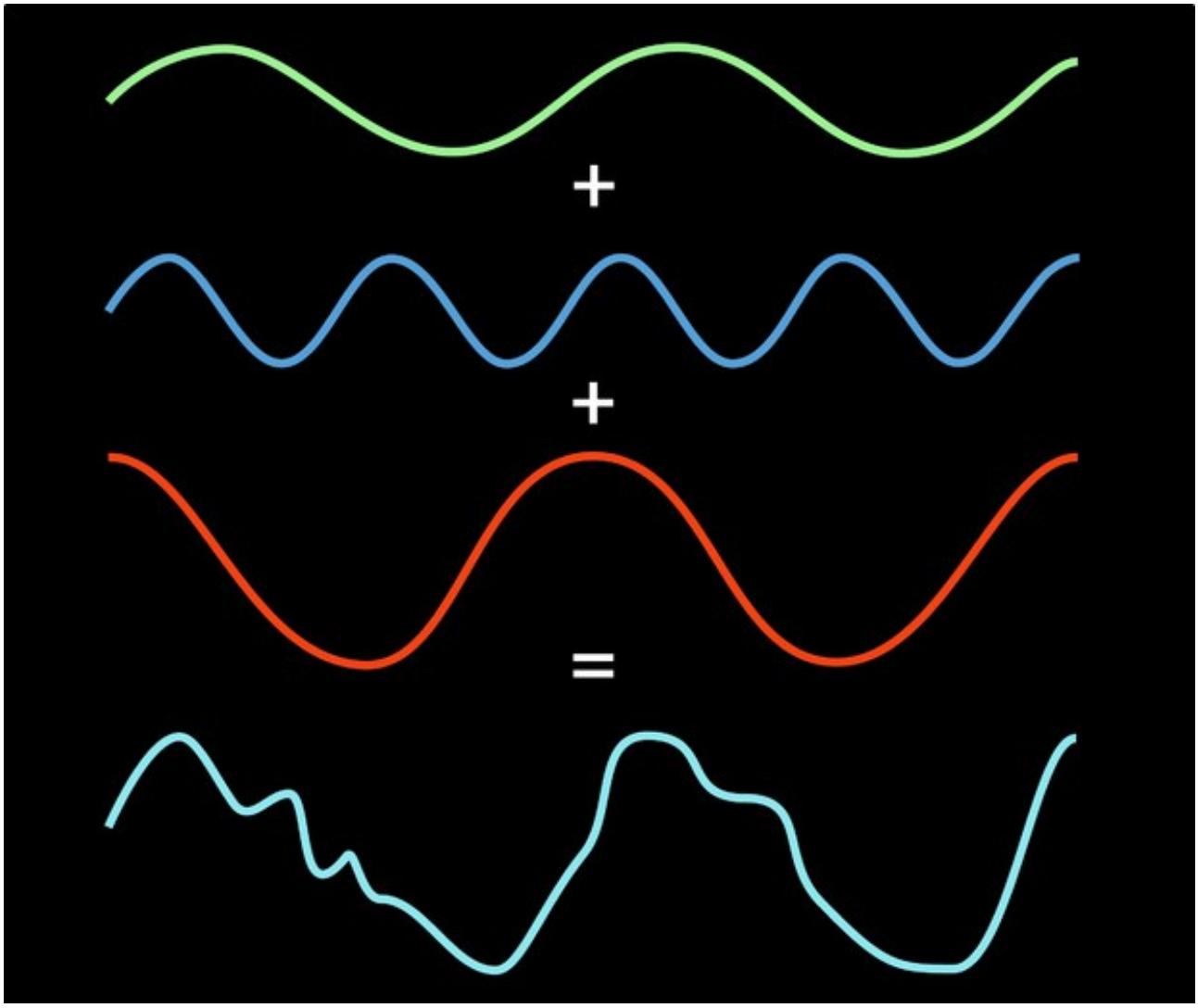
Zonder dit soort "sorteeralgoritmes" zouden heel wat moderne computertechnieken zoals data mining, AI, linkanalyse... ondenkbaar zijn.

25 GUNNINK, M., "Padvinder vindt pad: over Pathfinding", (http://kninnug.nl/padvinder/index.html), Geraadpleegd op 20 oktober 2015.
26 "Quick sort", (https://en.wikipedia.org/wiki/Quicksort), Geraadpleegd op 19 oktober 2015.
27 "Heapsort", (https://en.wikipedia.org/wiki/Heapsort), Geraadpleegd op 19 oktober 2015.

3.5.2 Conversie en compressie

De 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 het 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 bijvoorbeed 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."[28]



28 DE MOOR, W., "Sneller algoritme voor Fourier-transformaties ontwikkeld", (http://tweakers.net/nieuws/79444/sneller-algoritme-voor-fourier-transformaties-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. In 3.2.5 (Compressie) leerde je al meer over de belangrijkste compressie-algoritmes voor afbeelding. In deel 6.2.7 (Codecs en containers) lees je meer over compressie van film.

3.5.3 Foutcorrectie

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. PID-controllers 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.

3.5.4 Dijkstra

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."*[29]

3.5.5 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*

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

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)."[30]

3.5.6 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.

3.5.7 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**.

3.5.8 Controle en voorspelling

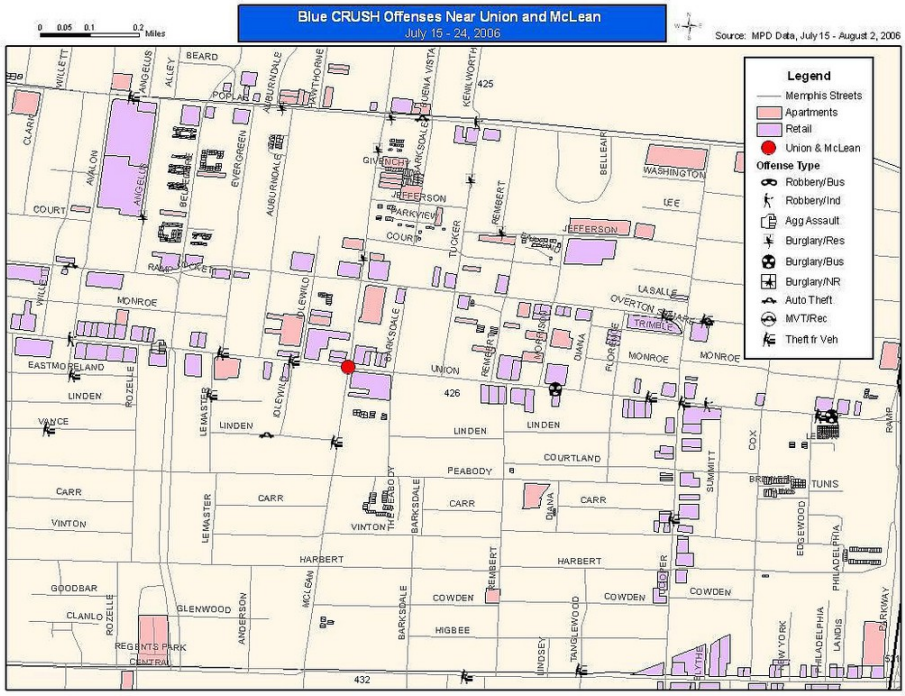
In 1948 schreef George Orwell het boek "1984" waarin hij scherpe kritiek leverde aan het adres van de USSR. Hij schetste

"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')."*[31]*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*

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

31 "Big Brother (George Orwell)", (https://nl.wikipedia.org/wiki/Big_Brother_(George_Orwell)). Geraadpleegd op 20 oktober 2015.

zijn internationale partners (Five Eyes: US, Australië, Canada, Nieuw-Zeeland, Groot-Brittannië) wereldwijd miljoenen mensen in de gaten. Ze monitoren telefoongesprekken, SMS-berichten, e-mailberichten, 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 misdaden 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."[32]

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.

3.5.9 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.

32 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.

3.6 Mechanische, elektronische en digitale computers

Rekenen kost veel te veel tijd, tijd die je aan aangenamer bezigheden kan besteden. Dat merkte men ook bij de Amerikaanse volkstelling van 1880: 500 ambtenaren hadden 7 jaar lang de handen vol om alle gegevens te verwerken. Herman Hollerith (1860–1929) was ambtenaar bij het landelijk bureau voor statistiek. Hij zag in dat je met de automatische verwerking van ponskaarten veel sneller zou kunnen werken. In 1890 introduceerde hij zijn Hollerithmachine. Gegevens zoals geslacht, leeftijd en nationaliteit van elke inwoner werden op ponskaarten opgeslagen. De machine telde en sorteerde! Met behulp van 42 machines slaagden hetzelfde aantal ambtenaren erin om de klus op een maand te klaren. In 1939 startte IBM met de ontwikkeling van een rekenmachine ,die vijf jaar later onder de naam Automatic Sequence Calculator (Mark I) het levenslicht zou zien. Het toestel werd samen met Harvard University ontwikkeld en was maar eventjes 15 meter lang en 2,5 meter hoog. De Duitser Konrad Zuse (1910– 1995) kende het werk van Babbage niet toen hij van start ging met zijn eigen rekenmachine, die volgens het **binaire talstelsel van Leibniz** werkte. Zijn eerste model (Z1) gebruikte nog mechanische schakelingen, maar beschikte wel al over een geheugen. Stilaan begon hij te experimenteren met elektromechanische verbindingen, wat resulteerde in de Z3, de eerste computer waarbij zowel het geheugen als de processor elektromagnetisch functioneerde. Het binaire talstelsel volstond niet voor de verwerking van gegevens. Een elektronische schakeling levert een 1 wanneer er een stroomstoot doorgaat en een 0 wanneer dit niet het geval is.

Bij computers is het van belang meerdere schakelingen te kunnen combineren. Soms mag een stroomstoot pas worden doorgegeven wanneer meerdere schakelaars op 1 staan, in andere gevallen volstaat het dat een van beide schakelaars een stroomstoot doorgeeft. De benodigde wiskunde werd geleverd door het werk van de Schotse wiskundige George Boole (1815–1864). Hij schreef in 1847 een boek waarin hij het binaire talstelsel combineerde met de logische verbindingen and, or, not. Voor de rekeneenheden van computers leverde dit de gepaste oplossing om meerdere verbindingen te kunnen combineren.

De Brit Alan Turing (1912–1954) ontwikkelde een theoretisch model voor een 'computer' onder de naam **turingmachine**. *Deze Logical Computing Machine bleef bij een gedachte-experiment.* Turing werd vooral beroemd omdat hij tijdens de Tweede Wereldoorlog de geheime Enigma-code kon kraken. Hiervoor ontwierp hij samen met Tommy Flowers (1905– 1998) de Colossus-computers, die voldoende rekencapaciteit hadden om de Duitse codes te kraken. In 1952 werd hij gearresteerd op verdenking van homoseksualiteit en tot een experimentele chemische castratie veroordeeld. Op 7 juni 1954 werd hij levenloos teruggevonden. Hij zou zelfmoord hebben gepleegd door te bijten in een met cyanide vergiftigde appel. Volgens de legende staat deze appel symbool voor het logo van de Amerikaanse computerfirma Apple. Over Turings dood deden al snel wilde verhalen de ronde. Volgens één complottheorie was Turing om het leven gebracht door de Britse geheime dienst omdat hij door zijn werk tijdens de oorlog op de hoogte was van veel staatsgeheimen.

De Hongaar **John von Neumann** (1903–1957) tekende kort na de oorlog de architectuur uit voor computers. Een computer moest beschikken over een invoereenheid, een processor voor de verwerking van de invoer, een uitvoereenheid en een geheugen voor de opslag van data. De gegevens en de programma's moesten binair (op basis van het talstelsel van Leibniz) worden opgeslagen. In 1945 bouwde hij zijn eerste edvac-computer, die echter duidelijk was afgekeken van een vroeger model van de Amerikaan John Atanasoff (1903–1995). Omdat het patent niet goed was geregeld, konden anderen met zijn ideeën aan de haal gaan. Hierdoor stond zijn toestel ook ongewild model voor de eniac (Electronic Numerical Integrator and Calculator) van John Mauchly (1907–1980) en John Eckert (1919– 1995). Zij presenteerden vol trots hun toestel als de eerste elektronische digitale computer. Dankzij de elektronenbuizen was de rekencapaciteit enorm vergroot, maar de buizen maakten het toestel ook erg kwetsbaar. Een elektronenbuis of vacuümbuis kan elektrische signalen versterken of nullen en enen doorgeven. De Mark I maakte voor zijn schakelingen gebruik van elektrische relais, die toelaten een grote stroom aan of uit te zetten met een kleine stroom. Hierdoor waren ze erg geschikt voor gebruik in computers. Maar het resulteerde wel in een zeer groot toestel met meer dan 700.000 onderdelen en

80 kilometer elektrische draden. In 1879 had Thomas Edison de elektrische gloeilamp uitgevonden. Hij leidde de stroom door een stukje verkoold katoen, waardoor het begon te gloeien en licht afgaf. Om te voorkomen dat de draad volledig verbrandde onder invloed van de lucht, plaatste hij het in een luchtledige glazen bol. Veel onderzoekers begonnen te experimenteren met de mogelijkheden van de lamp en dit leidde tot onder meer de radiobuis en de elektronenbuis, die ook perfect bruikbaar bleek als schakelaar in computers. Een relais bevatte veel bewegende onderdelen en zorgde voor relatief lange schakeltijden. Omdat een kleine wijziging in de elektronen een grote verandering kan veroorzaken in de doorgelaten stroom, was een elektronenbuis perfect bruikbaar als versterker. De ENIAC bevatte 18.000 buizen en was in alle opzichten sneller dan de Mark I. Toch had een elektronenbuis ook heel wat nadelen. Ze was niet alleen duur en erg breekbaar, maar slorpte ook massa's energie op.

De komst van de **transistor** in 1948 loste alle problemen op. De eerste patenten op een transistor dateren al uit 1928, maar lijken nooit in de praktijk te zijn omgezet. De transistor van William Bradford Shockley (1910–1989), John Bardeen (1908–1991), en Walter House Brattain (1902–1987) zou echter al snel de elektronenbuis verdringen. De transistor was veel kleiner, goedkoper en betrouwbaarder dan de elektronenbuis. Bovendien verbruikte hij minder elektriciteit en produceerde minder warmte. Net zoals een elektronenbuis is de transistor (transferresistor) een versterker, maar kan hij ook nullen en enen doorgeven. Jack Kilby (1923–2005) van Texas Instruments voegde in 1958 een tiental transistors samen in één geïntegreerde schakeling. Tegelijkertijd kwam Robert Noyce (1927–1990) van Fairchild Semiconductor met een soortgelijk circuit op de proppen. Het kreeg al snel de naam 'ic' (integrated circuit), maar is bij het grote publiek vooral bekend geworden onder de naam 'chip'.

Sindsdien is **de chip uitgegroeid tot het basisonderdeel van elektronische apparaten zoals de computer.** Het aantal transistors op een chip zou steeds maar toenemen. Bovendien werden de transistors alsmaar kleiner, waardoor een chip in onze tijd al snel miljoenen tot zelfs meer dan een miljard transistors telt. Het groeiende aantal transistors heeft ertoe geleid dat ook de rekenkracht van computers enorm is toegenomen. Volgens de befaamde **wet van Moore** (Gordon Earle Moore, 1929) zou de benodigde oppervlakte voor één transistor om de twee jaar halveren. Elke twee jaar kwam er dus een nieuwe 'chiptechnologiegeneratie' met transistors die slechts half zo groot zijn als die van de vorige generatie. **Uiteraard kan dit niet oneindig doorgaan. Men kan immers niet kleiner gaan dan de grootte van een atoom.** Lange tijd daalde ook de prijs van nieuwe transistors, maar ook daaraan komt stilaan een einde. De initiële investeringen voor het ontwerp van een chip zijn door de miniaturisatie zo hoog geworden dat de prijzen enkel nog zakken bij immense productievolumes. Van grote invloed op de **miniaturisatie** is de consumentenelektronica met steeds lagere prijzen. De schaling leidt er ook toe dat we elektronische toestellen steeds sneller als voorbijgestreefd beschouwen, wat de wegwerpmaatschappij in de hand werkt.

3.7 DNA- en quantumcomputers

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? 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? En 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.

3.8 Artificiële intelligentie

Wanneer kan je zeggen dat een computer "intelligent" is? Als hij slaagt voor de Turingtest? als hij kan schaken? Op het moment dat hij een quiz met open vragen wint? Dat niveau heeft een computer ondertussen al bereikt. Diverse computerdeskundigen waarschuwen voor het moment waarop de computer de menselijke intelligentie bereikt. Dat kantelmoment is op zich niet het probleem. Grotere vraagtekens kunnen we stellen bij het moment vlak *daarna.* 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... ze waarschuwen voor dit schakelmoment, dan men soms benoemt als de "singulariteit". Kunstmatige intelligentie zal zich nadien in razend tempo blijven verder ontwikkelen en zo intelligent worden dat AI door geen mens meer kan begrepen worden. Menselijke intelligentie is dus geen eindpunt, maar een kantelmoment waarna menselijke intelligentie snel overbodig lijkt te worden, als we de pessimisten mogelijk 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. Vermelde computerwetenschappers 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. Nochtans gaat de ontwikkeling op dat vlak in razend tempo. Zelfs Google waagt zich op dit terrein zoals blijkt uit de reeks roboticabedrijven die ze hebben overgenomen. Het massale gebruik van drones bij de oorlog in Syrië kan een eerste stap zijn naar Star Wars-achtige oorlogvoering.

Deel 4:
Storage: Langetermijngeheugen – data

Toen ik in 1991-1992 mijn thesis intikte, schreef ik eerst nog alles uit met de hand, met balpen op een massa velletjes papier. Een vriend beschikte toen reeds over een prille Applecomputer, tikte mijn teksten in (hij moet wel vaak hebben gegromd toen hij twee maanden lang stapels handgeschreven en moeilijke leesbare teksten onder de neus geduwd kreeg) en drukte de 200 A4-pagina's van mijn eindwerk af. Foto's moest ik er echter nog manueel aan toevoegen: foto kopiëren op een witzwartkopieermachine, uitknippen, plakken en het geheel nogmaals kopiëren. Niet veel anders gebeurde het bij de kranten. The times they are a-changin'.
Het grote voordeel van digitalisering is dat je de data ook achteraf kan bewerken. We bewaren al onze bestanden en gooien steeds minder weg, ook al is het dan virtueel. Seagate, de producent van harde schijven, verkocht in 2011 een totale capaciteit van 330 exabyte aan harde schijfruimte. Behoorlijk wat. Het world wide web zou in 2013 meer dan 3 zettabyte[33] aan data hebben bevat en dat is sindsdien exponentieel toegenomen.
In dit deel bekijken we hoe die overvloed aan data wordt bewaard. Hoe bewaart een digitaal systeem bestanden?

33 1 ZB = 1000^7 bytes = 10^{21} bytes = 1000000000000000000000 bytes = 1000exabytes = 1 billion terabytes = 1 trillion gigabytes. (Bron: https://en.wikipedia.org/wiki/Zettabyte)

4.1 Tekstbestanden of binaire bestanden?

Je tikt een tekst op je computer. Ongetwijfeld gebruik je hiervoor een tekstverwerker. Nadien bewaar je het als een bestand, want je wil het in de toekomst nog gebruiken of eventueel aanpassen. Hetzelfde doe je met die digitale foto waarvan je net de rode ogen hebt weggewerkt. Zowel je tekstdocument als de foto krijgen een plaats op je computer als een bestand op de harde schijf.

4.1.1 ASCII

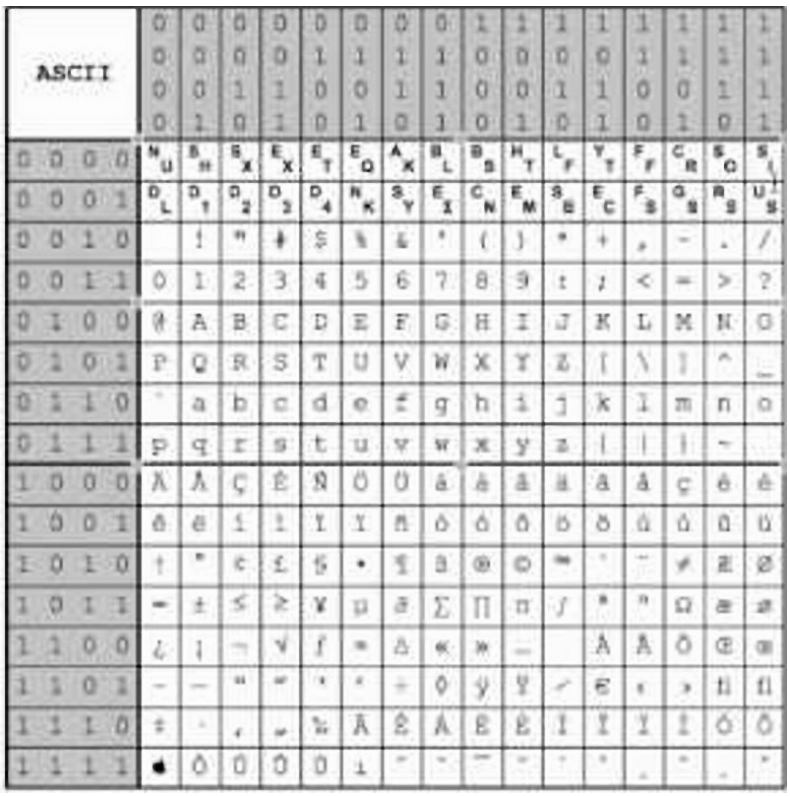
De computer bewaart beide bestanden binair, als een reeks nullen en enen, want dat is de enige code die een computer verstaat. Elk bestand bestaat uit een vaak erg lange, maar toch eindige reeks nullen en enen op het opslagmedium van je computer. We staan er niet vaak bij stil, maar het gebeurt wel verbazingwekkend snel. 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 2^8 combinaties (2 tot de achtste macht) want elke bit kan ofwel 1 ofwel 0 zijn. In totaal zijn op die manier 256 combinaties te vormen. Op die manier gebruiken we alle combinaties tussen

```
00000000 = 0
```

en

```
11111111 = 255
```

Hiervoor zijn er natuurlijk afspraken gemaakt. In principe kan je om het even welk toetsenbord op om het even welk digitaal systeem dat tekstinvoer toelaat, aansluiten, en de toetsaanslagen worden 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[34] 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.*

34 Stuurcode: Tot in de jaren 1970 werden digitale bestanden bewaard op ponskaarten, kaarten waarin een reeks gaten die énen of nullen representeerden konden worden aangebracht. Soms werd er per ongeluk een gat gemaakt waardoor er een fout in het “bestand” sloop. Men kon het gat natuurlijk niet meer dichtmaken. Daarom drukte men achter zo'n fout de stuurcode DEL die uit een reeks van 8 gaten bestond. Hierdoor werd het voor het systeem duidelijk dat het vorige teken mocht worden genegeerd.

4.1.2 UNICODE en UTF-8

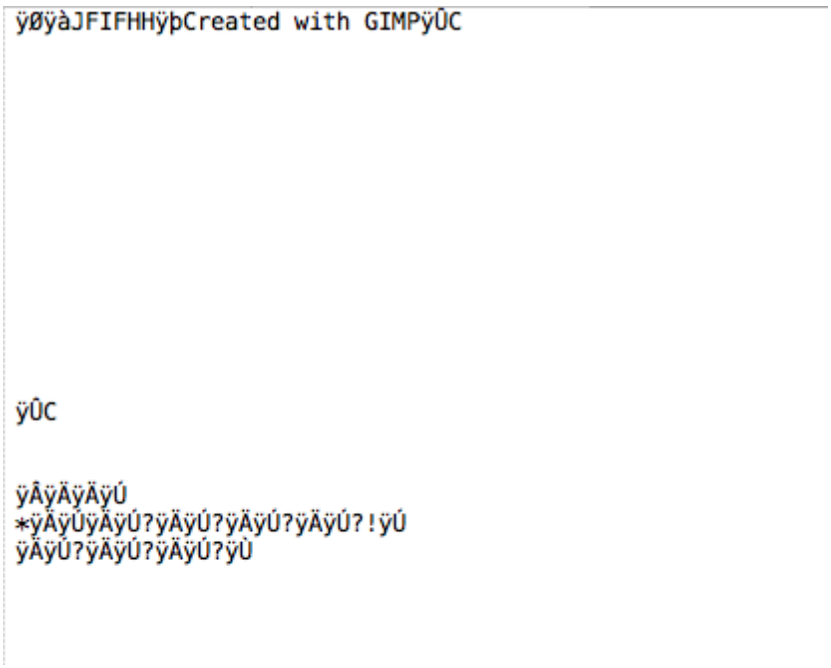
ASCII werkt zoals je merkt perfect, maar pakweg Arabieren, Chinezen en Japanners, kortom iedereen die geen standaard Europees georiënteerde taal schrijft, bleef met ASCII wel aardig op zijn honger zitten. Er bestonden geen binaire codes voor het Arabisch of Chinees (een Chinese Jan met de pet kent ongeveer 7000 verschillende karakters)[35]. 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[36].

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.

4.1.3 Platte tekst en/of binaire bestanden

Alle bestanden zijn binair (alle bestanden bestaan uit nullen en enen), maar toch maakt men vaak het onderscheid tussen PLAIN TEXT (platte tekst)-bestand en BINAIRE bestanden. Een platte tekst bevat enkel tekst (in ASCII) en regeleindes, maar geen informatie over opmaak, lettertypes enz. Een PLAIN TEXT-bestand kan je openen in een teksteditor zoals Kladblok (je merkt dat het nog ergens goed voor is) en de inhoud ervan zal leesbaar zijn voor ons mensen. Webpagina's, CSV-bestanden, XML enz. zijn leesbaar vermits opgesteld met ASCII- of UTF-8-tekensets. Wanneer je een foto zou openen in een teksteditor, zal dit ook zonder problemen lukken, maar veel zal je er niet uit opmaken. Hetzelfde geldt voor bijvoorbeeld PDF, ZIP-bestanden of uitvoerbare bestanden (exe).



Afbeelding: een foto geopend in een teksteditor.

35 China has always been “larger than life” when it comes to numbers and quantities. After all, it has the largest population out of any other country. It is no wonder then, that its language has become just as extensive! But exactly how big is it? The Chart of Generally Utilized Characters of Modern Chinese defines the existence of 7000 characters! If you think this number is high, you'll be shocked to hear that according to the Great Compendium of Chinese Characters or “Hanyu Da Zidian” (汉语大字典; Hànyǔ dà zìdiǎn), the number of existing characters is actually 54,678! But if you’re the kind of person that loves a challenge then there's the Dictionary of Chinese Variant Form (中华字海; Zhōnghuá zì hǎi). This work, also called the “Yìtǐzì zìdiǎn” (异体字字典), contains definitions for 106,230 Chinese characters! GARZA, J., “How many Chinese characters are there”, (http://www.hutong-school.com/how-many-chinese-characters-are-there), Geraadpleegd op 6 september 2015.

36 De Internationale Organisatie voor Standaardisatie (ISO) is een internationale organisatie die normen vaststelt. De organisatie is een samenwerkingsverband van nationale standaardisatieorganisaties in 164 landen. (https://nl.wikipedia.org/wiki/Internationale_Organisatie_voor_Standaardisatie)

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.

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.

Zoals eerder gezegd kan één enkele byte 256 verschillende waardes bevatten. Gaan we naar 4 bytes dan krijgen we 4 miljard mogelijke combinaties (2^{32}). 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.[37]

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. 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.

Toch hebben PLAIN TEXT-bestanden één bijzonder groot voordeel.

1. Ze zijn heel makkelijk uitwisselbaar
2. en niet strict gebonden aan een bepaald soort software.
3. Het is zelfs mogelijk om er gestructureerde gegevens in te bewaren.

37 AZAD, K., "A little diddy about binary file formats", (http://betterexplained.com/articles/a-little-diddy-about-binary-file-formats/), Geraadpleegd op 6 september 2015.

4.2 Bestandssystemen

4.2.1 Opslagmedium in blokjes

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. In het vorige onderdeel hebben we bekeken hoe digitale bestanden worden bewaard. 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.

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.

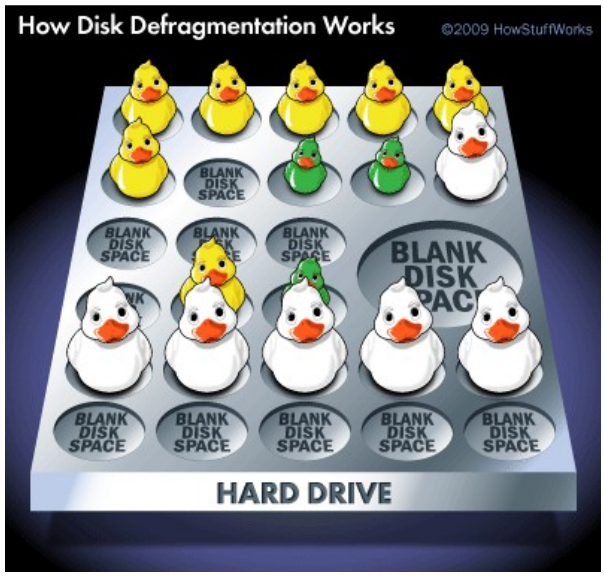
Wie een Windowscomputer gebruikt en het meemaakte dat de computer crashte, heeft wellicht ooit al eens gehoord over het begrip **FORMATTEREN**. Formatteren is een proces waarbij het besturingssysteem of een stuk andere software, de harde schijf of het opslagmedium gaat indelen. Zo'n beetje zoals een 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.

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. 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.[38]



Afbeelding [39]

4.2.2 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 over 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)
…	...

38 MAY, P., "How to Defrag Your Hard Drive", (http://electronics.howstuffworks.com/how-to-tech/how-to-defrag-hard-drive2.htm), Geraadpleegd op 7 september 2015.
39 Ibid.

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 e e n **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 ook relatief makkelijk herstellen (naar een vorige staat) na een crash.[41]

4.2.3 Soorten bestandssystemen

Bestandssysteem	*Kenmerken*
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.*[42]

40 LAYTON, J.B., "What's an inode?", (http://www.linux-mag.com/id/8658/), Geraadpleegd op 7 september 2015.
41 MILBERG, K., "How does a journaling filesystem work?", (http://searchenterpriselinux.techtarget.com/answer/How-does-a-journaling-filesystem-work), Geraadpleegd op 7 september 2015.
42 "Usage statistics and market share of Linux for websites", (http://w3techs.com/technologies/details/os-linux/all/all), Geraadpleegd op 7

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.*

4.3 Databanken

4.3.1 Tabellen met relaties

Sorry als je dit altijd hebt gedacht, maar een spreadsheetprogramma zoals MS Excel is geen databanktoepassing, ook al gebruiken heel wat mensen zo'n software nu net met dat doel. In een databank bewaar je gestructureerde informatie door ze te bewaren in een of meerdere tabellen.

Ga je regelmatig winkelen in een zaak waar men ook klantenkaarten uitdeelt? Heel vaak zal men je bij een aankoop eerst vragen naar je postcode. Hieronder kom je te weten waarom dit zo is.

Een eenvoudige adressenlijst zouden de meeste mensen meteen in één tabel bewaren. Die zou er dan als volgt kunnen uitzien:

Naam	*Voornaam*	*Straat*	*Huisnummer*	*Gemeente*	*Postcode*
Jan	Pieters	Bondgenotenlaan	900	Leuven	3000
An	Janssens	Naamsesteenweg	1234	Heverlee	3001
...	...	...	...	...	...

Vaak volstaat zo'n simpele databank niet, en ook in het geval van het bovenstaande voorbeeld kan het beter. Immers: als je adressenlijst aangroeit, moet je wellicht vaak dezelfde gemeente en postcode en dezelfde straatnaam invoeren. We zouden het bovenstaande voorbeeld op die manier in 3 tabellen kunnen onderbrengen.

Een tabel voor de gemeentes (tbGemeentes)

ID	*Gemeente*	*Postcode*
1	Leuven	3000
2	Heverlee	3001
...	...	...

september 2015.

Een tabel voor de straten (tbStraten):

ID	*Straatnaam*	*Gemeente_ID*
1	Bondgenotenlaan	1
2	Naamsesteenweg	2
…	…	…

Een tabel met de personen (tbAdressen):

Naam	*Voornaam*	*Straat_ID*	*Huisnummer*
Jan	Pieters	1	900
An	Janssens	2	1234
…	…	…	…

In deze tabellen gaan we ervan uit dat één persoon slechts op één huisnummer woont. De straatnaam hoeven we niet in te vullen. We kunnen enkel verwijzen naar een unieke "id" (identiteit) die de straat krijgt in de tabel "straten". Elke straat kan slechts in één gemeente liggen. Ik hoor je al denken dat meerdere gemeentes een "Dorpsplein" hebben, maar dat doet in deze databank niet terzake. Vanaf het moment dat je *Straat_ID=1* kiest, weet de databank dat *Straat_ID=1* in *Gemeente_ID=1* ligt. Op die manier kom je automatisch bij de juiste gemeente terecht.

In de gebruikersinterface vraagt men eers om bijvoorbeeld de postcode. Eens je die heb ingevoerd of geselecteerd, kies je de gemeente. Vervolgens krijg je een lijst van alle straten in die gemeente. Wanneer je de straat kiest, krijg je een overzicht van al je adressen in die straat. Een bijkomend voordeel is dat wanneer een straat van naam verandert, de straatnaam automatisch een update krijgt in alle adressen.

Omdat databanken beschikken over technieken om verschillende tabellen aan elkaar te koppelen of in relatie te brengen (denk aan *Gemeente_ID, Straat_ID*…) spreet men in dit geval over **RELATIONELE DATABANKEN**.

Relationele databanken worden in bedrijven, webshops, diensten, websites, elektronische leeromgevingen… massaal gebruikt. Zulke databanken maken het mogelijk om de inhouden die voor de gebruikers worden gepresenteerd, zinvol te bewaren. Lijsten met studenten, lijsten met punten in een digitale leeromgeving. Verkoopsartikelen, merknamen, kleuren en maten in een webshop. Facturen, betalingen en offertes in een boekhoudprogramma… Ook sociaal netwerksites zoals Facebook en Twitter en zoekmachines zoals Google bewaren hun resultaten en index in grote databanken.

4.3.2 CRUD

Een databank moet aan een aantal voorwaarden (de zogenaamde CRUD-voorwaarde) voldoen om een databank genoemd te worden. Meteen zal je duidelijk worden waarom een spreadsheetprogramma geen "databank" kan/mag genoemd worden:

C	*create*	Duurzame opslag van gegevens. Gegevens mogen niet dubbel opgeslagen worden. Een ID moet bijvoorbeeld altijd behouden blijven. Als je een rij verwijdert, behouden de andere rijen hun oorspronkelijke ID. Anders zijn de "relaties" om zeep.
R	*read*	Eenvoudig opzoeken en filteren van gegevens.
U	*update*	Gegevens moeten eenvoudig kunnen aangepast worden
D	*delete*	Gegevens moeten verwijderd kunnen worden.

Uiteraard moeten er op een databank **gebruiksrechten** worden ingesteld. Niet iedereen mag het recht hebben om gegevens te verwijderen of aan te passen, of zelfs toe te voegen. Toepassingen waar dit wel kan, zoals Wikipedia, bewaren nog steeds een bewerkingsgeschiedenis en hebben een controlemechanisme ingebouwd om misbruik of fouten te voorkomen.

Om CRUD-commando's uit te voeren, ondersteunen de meeste databanken de taal SQL (structured query language = gestructureerde opvraagtaal).

Indien we de gegevens uit onze adressenlijst zouden filteren op straat, dan zou dit in SQL kunnen klinken als

```
SELECT * FROM tbAdressen WHERE Straat_ID=1
```

of

```
DELETE * FROM tbStraten WHERE Gemeente_ID=2
```

4.3.3 Kleine en grote databanken

Databanksoftware voor huis-, tuin- en keukengebruik zoals bijvoorbeeld MS Access of Filemaker Pro, bieden een gebruiksvriendelijke interface voor CRUD-commando's. Accessdatabanken kunnen echter ook gebruikt worden voor databankopslag achter bedrijfssoftware die bijvoorbeeld geprogrammeerd is in Visual Basic of Visual C++.

Grote toepassingen die heel veel data moeten kunnen verwerken, vereisen natuurlijk navenante databanksystemen. Op websites is de databank **MySQL** de grote nummer één. Het databanksysteem is open source en gratis te gebruiken.

Zeer grote toepassingen zoals ERP-software[43] kunnen gebruik maken van MySQL of **ORACLE DATABASE**.

Databanken kunnen ook geautomatiseerd informatie verzamelen. In een aantal gevallen roept dit ernstige vragen op over schendingen van de privacy. De USA, Groot-Brittannië, Canada, Australië en Nieuw-Zeeland luisteren wereldwijd communicatiekanalen af via het spionagenetwerk **ECHELON**. Het marketingbedrijf

43 Het begrip ERP staat voor enterprise resource planning, waarmee in de regel een computerprogramma ofwel software wordt bedoeld. Dit soort computerprogramma's wordt voornamelijk binnen organisaties gebruikt ter ondersteuning van alle processen binnen het bedrijf. Een ERP-programma bestaat meestal uit kleine deelprogramma's (modules) die allemaal een specifieke taak ondersteunen. Denk bijvoorbeeld aan computerprogramma's die worden gebruikt om de financiële administratie te voeren of de voorraden bij te houden. Waarbij voorheen veel bedrijven gebruikmaakten van verschillende programma's om hun bedrijfsactiviteiten te ondersteunen, kunnen in een ERP-programma een heel groot aantal processen (afdelingen) worden ondergebracht. Een gevolg van een dergelijk ERP-systeem is dat gegevens van bijvoorbeeld klanten en producten (denk aan adresgegevens van klanten) binnen de organisatie slechts eenmaal ingevoerd hoeven te worden, omdat een enkele database wordt gehanteerd. Naast dat de kans op inconsistente data wordt verkleind kan een ERP-systeem veel andere voordelen bieden zoals een efficiëntere manier van werken, betere communicatie en soms zelfs kostenreductie.
"Enterprise resource planning", (https://nl.wikipedia.org/wiki/Enterprise_resource_planning), Geraadpleegd op 7 september 2015).

Doubleclick probeert te achterhalen welke e-mailadressen achter IP-adressen verborgen zitten. Voor bedrijven zijn zulke data veel geld waard.

4.4 Gestructureerde of semantische data voor informatie-uitwisseling

4.4.1 HTML en XML

Bekijk dit stukje HTML:

```
<div>
      <h1>Niet van gisteren</h1>
      <p>
      Techniek en technologie omringen ons, van de wastafel 's morgens tot de televisie
      's avonds, van de auto die ons naar het werk brengt tot de wekker die we zetten
      voor het slapengaan. Sommige uitvindingen lijken alsof ze nog maar recent in ons
      dagelijks leven zijn binnengekomen. Toch is het vaak net anders. Veel technieken
      en technologische concepten zijn ouder dan je denkt, zelfs eeuwenoud, of kennen
      hun voorgangers eeuwen of zelfs millennia geleden.
      </p>
      <p>Kris Merckx</p>
      <p>Prijs: &euro; 27</p>
      <p>Leuven, 2013</p>
</div>
```

Bekijk dit stukje XML:

```
<book>
      <title>Niet van gisteren</title>
      <description>
            Techniek en technologie omringen ons, van de wastafel 's morgens tot de
            televisie 's avonds, van de auto die ons naar het werk brengt tot de
            wekker die we zetten voor het slapengaan. Sommige uitvindingen lijken
            alsof ze nog maar recent in ons dagelijks leven zijn binnengekomen. Toch
            is het vaak net anders. Veel technieken en technologische concepten zijn
            ouder dan je denkt, zelfs eeuwenoud, of kennen hun voorgangers eeuwen of
            zelfs millennia geleden.
      </description>
      <author>Kris Merckx</author>
      <price currency="euro">27</price>
      <edition>
            <date>2013</date>
            <location>Leuven</location>
      </edition>
</book>
```

Misschien klap je al dicht of begint je hoofd al te tollen, maar doe even de moeite en je bemerkt wel een verschil.

Laat ons eerst even stilstaan bij de **gelijkenissen**. Als je goed kijkt, merk je dat de onderdelen van de tekst in beide gevallen zijn afgesloten door markeringen. Daarvoor staat ook de ML in HTML en XML. ML betekent MARKUP LANGUAGE of markeertaal. Je markeert het begin en het einde van elk onderdeel. Men spreekt in dit geval ook wel van "tagging". Net zoals je in Facebook aanduidt of tagt welke personen op een foto staan, zo markeer je hier het begin en einde van een element met code. Het opzet is vrij eenvoudig. Je begint met <tagnaam> en je eindigt met </tagnaam>. Zoals je merkt, kan je markeringen/tags ook in elkaar stoppen, alsof je kleinere dozen in grotere dozen stopt.

De bedoeling van **markeertalen** is dan ook inhoud die voor mensen relatief duidelijk is te structureren voor computers. Een computer weet immers niet waar een bepaald tekstonderdeel zoals een titel of alinea eindigt. Daarom vertellen we dit aan het toestel met markeringen.

Beide voorbeelden bevatten in wezen dezelfde inhoud, maar toch is er voor computers een belangrijk **verschil**.

Beeld je in dat beide voorbeelden zouden bekeken worden door iemand die geen Nederlands spreekt, dan merk je het meteen. Bij het eerste voorbeeld zou de Engelstalige misschien door hebben dat de <h1> staat voor een titel en de <p> voor een paragraaf (dat is ook de betekenis), maar hij/zij zou geen idee hebben over de inhoud. Het XML-voorbeeld structureerd niet alleen de inhoud, maar vertelt ook iets over de inhoud. Het gaat over een ... (ja!) boek.

Vanuit het standpunt van informatie-uitwisseling is **XML** dus veel handiger, maar voor computers is het eigenlijk één pot nat, want dat toestel "begrijpt" al evenmin wat een boek is. Gestructureerde semantische[44] markeringen zijn dus vooral zinvol bij informatie-uitwisseling tussen computers, met die voorwaarde dat er steeds mensen komen bij kijken die zulke programmeercode schrijven dat ze dit soort uitwisselingsformaten handig kunnen gebruiken.

Om informatie uit databanken te ontsluiten zijn XML-gebaseerde talen zeer handig. Er is echter nog een groot verschil tussen HTML en XML. De markeringen voor HTML liggen vast, ze zijn internationaal afgesproken. Elke webontwikkelaar weet waarvoor <h1> en <p> staat (en zo zijn er nog een hoop tags). Een XML-taal kan door de ontwikkelaar zelf bedacht worden. In plaats van <book> had hij evengoed <dog> of <product> kunnen gebruiken. De X in XML staat immers voor Extensible of Extended.

4.4.2 Met goedkeuring en zonder goedkeuring stelen

Moest alle informatie over boeken op een website als Amazon of BOL.com opgebouwd zijn in het XML-formaat zoals hierboven aangeduid, dan zouden andere ontwikkelaars heel snel die informatie kunnen overnemen, filteren en gebruiken. Onzichtbaar voor de gewone gebruiker gebeurt dit ook. Heel wat online diensten bieden een **REST-service** aan waardoor ze anderen toegang geven tot hun data.

REST is een bijzonder krachtig instrument omdat het je toelaat niet alles te moeten gaan zoeken, filteren, kopiëren en plakken in je eigen databanken. Met REST kan een webontwikkelaar geautomatiseerd zijn eigen pagina's opvullen met informatie die afkomstig is van andere websites en internetbronnen.

Wanneer een interessante online dienst geen REST-service of REST-api aanbiedt, dan kan de ontwikkelaar nog steeds krachtige **SCRAPING**-methodes programmeren. Dat zijn programmeertechnieken die een beetje lijken op reguliere expressies, waardoor geautomatiseerd inhouden van andere webpagina's kunnen gekopieerd en bewaard worden.

Bekende uitwisselingsformaten op basis van XML zijn ondermeer RSS (voor nieuws), Podcasts (voor muziek) en Vodcasts (film). Ook voor afspeellijsten gebruikt software vaak een XML-gebaseerd formaat. Een ander bekend formaat dat veelvuldig in de code van webpagina's wordt gebruikt, is **JSON** (Javascript Object Notation). Als je het onderstaande voorbeeld bekijkt, zal je al snel vaststellen dat XML vlotter leesbaar is:

```
{
    "book": {
        "title": "Niet van gisteren",
           "description": "\nTechniek en technologie omringen ons, van de wastafel s
morgens tot de televisie s avonds, van de auto die ons naar het werk brengt tot de
wekker die we zetten voor het slapengaan. Sommige uitvindingen lijken alsof ze nog maar
recent in ons dagelijks leven zijn binnengekomen. Toch is het vaak net anders. Veel
technieken en technologische concepten zijn ouder dan je denkt, zelfs eeuwenoud, of
kennen hun voorgangers eeuwen of zelfs millennia geleden.\n\t",
        "author": "Kris Merckx",
        "price": {
            "_currency": "euro",
            "__text": "27"
        },
        "edition": {
            "date": "2013",
            "location": "Leuven"
        }
    }
}
```

44 "Semantiek", (https://nl.wikipedia.org/wiki/Semantiek), Geraadpleegd op 7 september 2015.

4.5 Big data

De term “big data” duikt veelvuldig op in de pers. Het is geen computertechniek, maar slaat op de gigantische hoeveelheid digitale informatie die wereldwijd beschikbaar is. In 2016 zullen datacenters (plaatsen waar internetservers staan opgesteld) wereldwijd 16000 ha oppervlak innemen. De gezamenlijke datacapaciteit is dus enorm en naar schatting is op dit moment slechts 0.5% van die gegevens geanalyseerd. “Men” hoopt hiervan binnen enkele jaren tijd tot 33% te taggen en analyseren.

Waarom is zo'n analyse “nodig” of beter gezegd: Wie vindt zo'n analyse nodig?

De redenering lijkt eenvoudig: hoe meer data, hoe nauwkeuriger de analyses, hoe efficiënter de beslissingen van bedrijfsmensen en overheden. Hierdoor kan men kosten drukken en risico's reduceren.

“Big data” is een modewoord en vaak gebruikt men het ten onrechte of in een verkeerde context. Drie factoren zijn cruciaal volgens industrieel analyst Doug Laney[45]:

Volume	De hoeveelheid gegevens.
Snelheid	De snelheid waarmee informatie kan opgevraagd en ingevoerd kan worden. Hoe sneller, hoe sneller en efficiënter kan gereageerd worden.
Diversiteit	Diverse soorten gegevens worden geanalyseerd: gestructureerde data uit bijvoorbeeld databanken, sensoren, informatie uit sociale media, video's van YouTube, financiële transacties, beursgegevens enz. Diverse data kunnen elkaar ook tegenspreken waardoor een grondige analyse erg complex wordt. Hoe bepaal je welke bron de meest correcte informatie bevat? Hoe kan je bovendien op een zinvolle manier al die verschillen datatypes zinvol analyseren en combineren?

Om te kunnen spreken over “Big data” moeten minstens twee van deze factoren zijn voldaan.

Bij big data gaat het dus niet zo zeer over de hoeveelheid data die je verzamelt, maar eerder over wat je met die data doet.

Analyse van big data kan leiden tot

1. kostenbesparing
2. tijdsbesparing
3. nieuws producten of productoptimalisatie
4. betere besluitvorming
5. optimalisatie van producten, transport enz.
6. frauduleuze transacties ontdekken[46]
7. ...

De hoeveelheden data zijn zo groot dat ze met reguliere databanken niet kunnen opgeslagen en geanalyseerd worden. Bovendien groeit de hoeveelheid beschikbare data exponentieel, ondermeer door de sociale media zoals Facebook en YouTube. Daarnaast verzamelen steeds meer toestellen zelf data en wisselen die ook uit

45 META Group. "3D Data Management: Controlling Data Volume, Velocity, and Variety." February 2001.
46 "Big Data What it is & why it matters", (http://www.sas.com/en_us/insights/big-data/what-is-big-data.html), Geraadpleegd op 7 september 2015.

(denk aan het Internet of things). RFID-chips zijn de intelligente opvolgers van de traditionele streepjescodes en kunnen eveneens in dit proces van "**datamining**" worden ingezet. **Al die gegevens vormen een schat aan informatie voor marketingdoeleinden.** Datamining is het gericht zoeken naar statistische verbanden tussen gegevens. Men probeert op die manier profielen op te stellen zoals het aankoopgedrag van consumenten, verpreiding van ziektes enz. Dankzij datamining en profilering kan men het gedrag van mensen, netwerken, groepen of systemen "voorspellen". Maar soms slaat men de bal toch wel aardig mis.

De analyse en verwerking van big data vraagt niet alleen om nieuwe software, databanken en bestandssystemen, maar ook om gespecialiseerde beroepen zoals dat van "**datascientist**". Met nieuwe algoritmes probeert men onbekende verbanden tussen data te ontdekken. Omdat de data uit zeer diverse bronnen komen en in zeer verschillende formaten, volstaan traditionele algoritmes en programmeertechnieken niet meer. De software moet zelf leren uit de gevonden informatie (**machine learning**) en besluiten trekken (lees: voorspellingen doen). In de programmeercode maakt men ondermeer gebruik van **neurale netwerken** waarbij men de manier waarop menselijke hersenen leren, probeert na te bootsen.

Wie anders dan Google is erg bedreven in het verzamelen en analyseren van gigantische hoeveelheden informatie. Zij introduceerden **MapReduce**, een programmeerframework voor het uitvoeren van analyses op grote hoveeheden gegevens. MapReduce splits een zoek- of analyseopdracht op in meerdere kleinere taken en verdeelt die over meerdere computers. Als één van de computers tijdens dit werk de "geest geeft", neemt een andere computer automatisch die openstaande taak over. De **Apache Hadoop**-omgeving volgt met enige fantasie ook het IPOS-model:

1. de invoer bestaat uit "big data",
2. het **Hadoop Distributed File System** (HDFS) staat in voor de opslag van de gegevens (op meerdere servers, blockgrootte van 64 mb),
3. MapReduce zorgt voor de "verwerking" en het
4. resultaat is een analyse.

Soms loopt het ook wat "mis":

Voorbeelden:

"De Amerikaanse winkelketen Target kwam in het nieuws, doordat ze folders over babybenodigdheden naar een vrouw hadden gestuurd, nog voordat haar familie wist dat ze zwanger was. Het bedrijf wilde zich bewust op zwangeren richten omdat zwangerschap een van de weinige periodes in iemands leven is waarin de vastgeroeste winkelpatronen doorbroken kunnen worden. Via statistische analyse was het bedrijf in staat zwangeren te herkennen aan hun veranderde aankoopgedrag. Critici stelden dat het bedrijf weliswaar babyfolders naar zwangeren stuurde, maar mogelijk ook veel van die folders naar niet-zwangeren had gestuurd.

Toen in 2009 de Mexicaanse griep toesloeg, kon Google de verspreiding volgen door te zien in welke regio's naar griepgerelateerde klachten werd gezocht. Deze analyse werd wel vertroebeld doordat veel mensen die via het nieuws over de ziekte hadden gehoord, al gingen zoeken terwijl ze (nog) niet ziek waren."[47]

47 "Big data", (https://nl.wikipedia.org/wiki/Big_data), Geraadpleegd op 7 december 2015.

4.6 Gegevens kleiner maken

We kunnen ondertussen zo goed als alle data digitaliseren en bewaren op optische of magnetische opslagmedia. Geluid, bewegend en stilstaand beeld, tekst, 3D-gegevens ... alles kan door de digitale mallemolen en voor de 'eeuwigheid' bewaard worden. Via het wereldwijde netwerk van computers dat we het internet noemen, kunnen we die digitale data met elkaar uitwisselen. Enkel de opslagcapaciteit en de snelheid van de datanetwerken vormde lange tijd een probleem. Hiervoor ontwikkelden programmeurs **compressietechnieken**, waardoor digitale gegevens minder ruimte innemen op een medium, maar toch geen merkbare informatie verliezen.

Test: *Als je over een computer met tekstverwerker beschikt, kun je zelf een compressietechniek bedenken door de meest voorkomende woorden in een tekstdocument met een 'zoek-en-vervang'-opdracht te vervangen door een getal (waarvan het aantal tekens wel korter moet zijn dan die van het woord zelf) of de eerste letters. Wanneer je twee versies van het document bewaart, zul je merken dat de tekst met compressie minder opslagruimte in beslag neemt. Je moet natuurlijk een decoderingsmechanisme voorzien zodat de gebruiker weet voor welk woord elk getal staat.*

Uiteraard werken moderne compressietechnieken volgens zorgvuldig uitgekiende wiskundige algoritmes. Tot de bekendste compressiealgoritmes behoren mp3 (geluid), jpeg (beelden) en zip.

Karlheinz Brandenburg (1954) kan terecht de vader van het **mp3**-formaat worden genoemd, maar het succes van mp3 is zeker niet aan hem te danken. Een Australische hacker 'kocht' de demo-software voor het coderen van mp3 met een gestolen creditcardnummer en schreef er een nieuwe interface voor. Vervolgens publiceerde hij zijn programma op het internet, waar iedereen het gratis mocht downloaden. Het gaf duizenden mensen de mogelijkheid om muziek van audio-cd's te halen en te converteren naar het mp3-formaat. Al snel kwam er software beschikbaar waarmee mensen overal ter wereld hun muziekcollectie konden 'delen'. Muziek van internet halen werd een fluitje van een cent – tot groot ongenoegen van de muziekmaatschappijen. Rechtszaken volgden en daarbij werden vooral de producenten van de verdeelsoftware geviseerd. Er er vielen rake klappen: hoe succesvol software als Napster ook was, het kon niet op tegen de gerechtelijke machine.

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. Bij beelden gebeurt een gelijksoortige compressie. Aangezien het menselijk oog veel gevoeliger is voor verschillen in helderheid dan voor kleurverschillen, worden bij **jpeg-compressie** veel kleurverschillen weggelaten. Hoe meer verschillen worden weggelaten, hoe groter de compressie.

Bij analoge film 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. Compressietechnieken worden eveneens gebruikt wanneer in een massa digitale gegevens moet worden gezocht naar bepaalde patronen. Bij computer vision is een machine in staat om informatie uit afbeeldingen of beelden af te leiden. Implementaties zijn niet moeilijk te bedenken:

het sturen of besturen van robots, tellen van voetgangers, afbeeldingen indexeren en organiseren, interactie met de omgeving en vele andere.

4.7 Rotte gegevens en dataverlies

Nauwelijks twintig 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) en de moed niet te 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 dagelijks 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 geopend worden. 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 en kunnen hanteren. 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 een paar 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. Het moderne 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?

Deel 5:
Netwerken

Een digitaal systeem staat zelden alleen. Net zoals ieder van ons rond zich een sociaal netwerk van vrienden, kennissen, collega's, bekenden verzamelt, zo werkt een computer zelden in zijn "eentje".
Computers kunnen in een netwerk samenwerken en uitgroeien tot één supercomputer door hun rekenkracht te combineren zoals bij het renderen van een 3D-film gebeurt. Ze kunnen ook hun opslagruimte combineren zoals bij big data of zoekmachines het geval is.
Het meest bekende netwerk is het internet, waarvan het web (http), e-mail (mailto), chat... de bekendste toepassingen vormen. Het internet of things moet in de nabije toekomst alle digitale apparaten via het internet informatie laten uitwisselen.
In dit deel leer je niet meteen hoe je netwerken opstelt, maar leer je iets dieper en vooral grondiger graven in het internet. Je leert sneller en beter informatie verzamelen en je leert hoe zoekmachines functioneren.

Waarom gaat zoeken in Google zoveel maal sneller dan een simpel bestand terugvinden op je harde schijf? Dat zou een fijne examenvraag zijn waarbij je de informatie uit het hoofdstuk over bestandssystemen en uit dit deel moet vergelijken.

5.1 De werking van het internet

5.1.1 Clients en servers: de klanten en hun obers

Het internet kan je niet zo maar vastpakken. **Waaruit bestaat het** precies, welke onderdelen maken deel uit van het "internet"? In de eerste plaats bestaat het uit een **bonte verzameling hardware**: routers, kabels, servers, masten voor mobiele telefonie, satellieten in een baan om de aarde, radio's, mobiele telefoons, computers.... Ze vormen van een wereldwijd netwerk van netwerken. Het internet is flexibel: als je onderdelen verwijdert of toevoegt, ondervindt het naar nauwelijks hinder van of aanpassingsmoeilijkheden. Een router verwijderen of een kabel uittrekken legt niet het ganse internet plat. Het lijkt in dat geval meer op een speldenprik in je vingertopje.

Bepaalde onderdelen zijn relatief statisch en vormen de ruggengraat van het internet. Zo'n onderdelen kan je niet zonder problemen overboord gooien. Je computer, mobiele telefoon... vormen eindpunten in het internet. Men spreekt in dit geval vaak van "**clients**" die door "**servers**" worden bediend, zoals een restaurantbezoeker door een ober wordt bediend. Een klant (client) bestelt eten en de ober levert wat later de gevraagde schotel. Je browser op je computer vormt een client die een webpagina opvraagt en de webserver stuurt even later de gevraagde webpagina naar de browser van de "bezoeker".

Als je een pakje bestelt bij een postorderbedrijf, dan legt je pakje een hele weg af via het bedrijf dat het artikel prduceert, de transportmaatschappij, het postkantoor, de postbode... vooraleer het pakje bij de "client" is. Al die tussenschakels vindt je ook terug op het internet, men noemt het "**nodes**". Tussen een server en een client liggen op die manier één of meerdere nodes die het verzenden en uitwisselen van "data"**pakketjes** mogelijk maken. **De server(s), nodes en clients zijn met elkaar verbonden**. Die verbindingen kunnen fysiek (tastbaar) zijn in de vorm van "kabels", maar ook draadloos (via satellieten, radioverbindingen, GSM-masten...)

5.1.2 Goede afspraken maken goede vrienden: protocollen

Om heel het verkeer vlot te laten verlopen zijn er verkeersregels nodig, afspraken tussen alle gebruikers, servers, nodes, clients. Enkel duidelijke afspraken maken een vlot verkeer mogelijk. Die regels en afspraken zijn vastgelegd in een reeks **protocollen**. Zoals het verkeersreglement uitlegt dat je pas mag doorrijden met je auto als het licht op groen springt en je moet rekening houden met voorrangsregels, zo leggen **communicatieprotocollen** vast op welke manier een vlotte uitwisseling van informatie tussen servers, nodes en clients mogelijk wordt. Protocollen bepalen niet enkel regels, maar ook een reeks te volgen stappen om bepaalde taken uit te voeren. Alle toestellen die op het internet zijn aangesloten, moeten die protocollen volgen. Als iedereen zo maar zijn zin doet, alle toestellen hun eigen regels volgen en hun eigen taaltje spreken, dan is een vlot verkeer niet mogelijk. Een protocol bepaalt dus ook de code (taal) waarin toestellen met elkaar communiceren, net zoals iedere weggebruiker de pictogrammen op de verkeersborden moet begrijpen om te weten welke regels hij in elke verkeerssituatie dient te volgen.

Huizen hebben een adres. Dit zorgt ervoor dat de postbode de brieven makkelijk bij de juiste personen kan afleveren. Op het internet hebben alle clients, servers en nodes eveneens een adres. Indien dit niet het geval zou zijn, zou het uitwisselen van informatie tussen een client en een server behoorlijk rampzalig worden. Een server

zou niet meer weten aan welk toestel hij de gevraagde informatie moet sturen. Het protocol dat zorgt voor de verdeling van die adressen, heet het **Internet Protocol** of kortweg **IP**, wat je zeker kent van de veelgebruikte term IP-adres. Het protocol dat zorgt voor de verzending van informatie tussen diverse adressen het het **Transmission Control Protocol of TCP**. Beide protocollen worden vaak in één adem uitgesproken als TCP/IP. Sommige servers (DNS-servers, of **Domain Name Servers**) nemen de rol van adressenlijst en richtingaanwijzer op zich. Ze sturen elke vraag die binnenkomt door naar het gepaste adres. Hoe meer toestellen op het internet worden aangesloten, hoe meer adressen nodig zijn. Net dat vormt een probleem. Elk adres bestaat uit een reeks cijfers en cijfers worden door digitale systemen binair gecodeerd. De vierde versie van IP gebruikte hiervoor een systeem van 32 bits. Dat voldeed geruime tijd en kon voldoende verschillende adressen leveren voor elk toestel. Door het toenemende aantal op het internet aangesloten toestellen (mobiele toestellen, webservers, internet of things...) raakte het aantal beschikbare adressen uitgeput. Het Internet Engineering Task Force (stel het je niet voor als een reeks Amerikaanse superhelden) besliste reeds in 1991 om IP te vernieuwen naar een **adresseringssysteem** van 128 bits: dit vernieuwde IP-protocol kreeg de welluidende naam IPv6 (versie 6).

Bovenop de TCP/IP-laag stoeit men met nog een reeks andere protocollen. Ongetwijfeld kan je **HTTP**, het **hypertext transfer protocol** dat een reeks regeltjes bevat voor het uitwisselen van webpagina's en data tussen webservers en browsers. Het world wide web (www) maakt gebruik van dit protocol. Het wereldwijde web bestaat uit een gigantische reeks websites en webpagina's. Websites vraag je op door een **domeinnaam** of **URL** (uniform resource locator) in te voeren in je browser. Een URL bestaat uit een reeks onderdelen:

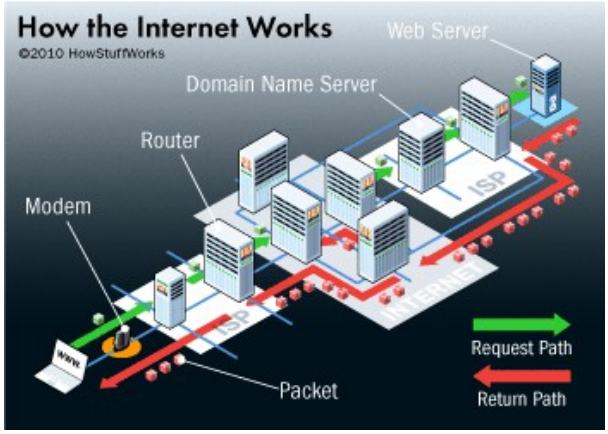
Protocol	*domein*	*domeinnaam*	*extensie*	*Map, bestand (eventueel)*	*Map, bestand (eventueel)*
http://	www. , users. , ...	blabla	.com, .be ...	/mapnaam /index.html	/bestandsnaam.html /nogEenMap

Op één server (met één enkel IP-adres) kunnen zich meerdere websites bevinden. Op één IP-adres kunnen zicht dus meerdere domeinen bevinden, net zoals in één appartementsgebouw meerdere gezinnen (personen met “namen”) kunnen wonen.

Een webontwikkelaar die op zijn computer een website bouwt, verstuurt alle gemaakte webpagina's naar de webserver. Hij doet dit met een stuk software dat snel bestanden kan uitwisselen met een webserver via het **FTP**-protocol (**File Transfer Protocol**). Uiteraard kan niet iedereen zo maar een website aanpassen. De webontwikkelaar heeft hiervoor een gebruikersnaam en wachtwoord nodig, net zoals bij het MAILTO-protocol (je weet wel: e-mail).

5.1.3 Packet switching: uitwisselen van informatie tussen computers

Wanneer je http://www.schoolvoorbeeld.be/Multimedia_28.html invoert in je browser, stuurt je computer een aanvraag (request) naar je Internet Service Provider of ISP (Telenet, Belgacom...). De ISP stuurt die aanvraag door naar bijvoorbeeld een DNS (domain name server) die nakijkt bij welk IP-adres de domeinnaam thuishoort. Als hij een overeenkomst vindt (er bestaan ook domeinnamen die nog geen "plaatsje" hebben op een IP-adres, je zou in dit geval van dakloze domeinnamen kunnen spreken), stuurt hij de aanvraag door naar de webserver die zich op dat IP-adres bevindt. De webserver stuurt het gevraagde bestand (en eventueel gekoppelde bestanden zoals afbeeldingen, stijldocumenten enz.) terug naar de computer (browser) die erom gevraagd heeft. Die pakketjes volgen niet noodzakelijk dezelfde route als de "request". Een response (antwoord) kan een totaal andere weg volgen, afhankelijk van welke verbindingen (telefoonlijnen of glasvezelkabels) er op dat moment vrij zijn (de weg met de minste file). Soms wordt een pakketje zelfs opgesplitst in meerdere brokjes om bij de eindbestemming weer te worden samengevoegd. Het lijkt wel alsof de ontvanger alle puzzelstukjes weer moet samenvoegen. Die techniek van **packet switching** werd reeds in de jaren 1960 ontwikkeld. Tom Igoe legt het in zijn boek "Making things Talk" als volgt uit:



"Vergelijk het met het opsturen van een fiets naar een ander adres. De fiets is veel te groot om in één pakje te verzenden via de post. Daarom haal je hem eerst uit elkaar zodat elk onderdeel in een kleine doos past. Op een netwerk (...) wordt ieder bericht in kleine stukjes gebroken van min of meer dezelfde grootte. Elk stuk krijgt een pakketnummer. Daarna schrijf je het adres van de bestemmeling en de afzender op elke doos. Daarna verzend je het. De koeriersdienst verspreidt je pakjes misschien over meerdere vrachtwagens als dat zo beter past. Op het internet gebeurt dit in de transportlaag. Die laag is verantwoordelijk voor het verzenden van alle pakjes naar hun bestemming. (...) Elke router stuurt de pakjes één voor één naar de routers waarmee hij verbonden is. Als hij met meerdere routers verbonden is, kiest hij de minst drukke. Elk pakje kan een andere weg naar zijn bestemming volgen. De ontvanger leest de header-gegevens en voegt alle pakjes weer samen."

5.2 Van hypertekst tot het web

Toen ik zelf nog op de schoolbanken zat, raakte ik snel afgeleid door het simpelweg lezen van een woord in een oefening. Ik herinner me dat ik in het vak Nederlands ooit een zin over een astronaut moest verdelen in zinsdelen. In mijn hoofd zweefde ik ondertussen al ergens halfweg tussen de aarde en de maan. Dat is exact wat we in ons hoofd doen: we leggen analogieën en volgens sommige onderzoekers is dat ook de manier waarop onze hersenen informatie opslaan.

5.2.1 Wat is hypertekst en hypermedia?

Veel leraars zouden in dat geval zeggen dat hun leerlingen te snel zijn afgeleid. Het leggen van verbanden tussen inhouden, teksten, fragmenten van teksten... is wat iedereen helemaal gewend is door de komst van het

web. Op webpagina's vind je niet simpelweg “tekst”, maar “**hypertekst**”, **tekst die verbanden legt met andere teksten of *media*.**

In veel gevallen legt de webontwikkelaar de links zelf, maar het kan ook geautomatiseerd gebeuren zoals bijvoorbeeld in een zoekmachine zoals Google of in de online encyclopedie Wikipedia (zie afbeelding hieronder).

Enkele begrippen [bewerken]

Hypertekst [bewerken]

Een hypertekst is een welbepaalde (maar meestal niet onveranderlijke) verzameling hypertekst. Bekende voorbeelden:

- Wikipedia
- de Internet Movie Database
- Amazon, eBay, of een willekeurige andere webapplicatie of website
- Windows Help-handleidingen van Microsoft Windows-applicaties
- GNU info-handleidingen van GNU-applicaties

Hypertekstsysteem [bewerken]

Een hypertekstsysteem is een softwaresysteem voor het maken en gebruiken van hypertekst. Bekende voorbeelden:

- Mediawiki (waar Wikipedia in wordt gemaakt)
- Windows Help met bijvoorbeeld de Microsoft Windows Help Workshop
- GNU info met bijvoorbeeld texinfo
- softwaredocumentatie-hypertekstgeneratoren zoals doxygen, Javadoc en NDoc

Hypertekststandaard [bewerken]

Een hypertekststandaard is een verzameling technische afspraken voor het representeren en gebruiken van hypertekst. De met grote afstand meest bekende standaard op dit gebied is het World Wide Web, dat bestaat uit drie onderdelen:

- HyperText Markup Language (HTML), een opmaaktaal voor hypertekst;
- de Uniform Resource Locator (URL), een documentadresseermethode voor het Internet;
- het HyperText Transfer Protocol (HTTP), een protocol voor het opvragen en oversturen van documenten op het Internet.

Ooit zei iemand me:

> *“Heel die multimedia en dat internet, het is eigenlijk niet meer dan 'klikken' en nog eens 'klikken'.”*

Gedeeltelijk klopt zo'n uitspraak ook wel. Hypertekst en bij uitbreiding “hypermedia” vormt de basis van multimedia. Tekst, afbeeldingen, film, geluid... **wordt aan elkaar gelinkt al dan niet via voor de gebruiker zichtbare links**.

Oneerbiedig zouden we onszelf de “zap”generatie kunnen noemen. *Eén belangrijk punt van kritiek* moeten we meenemen in het multimediaverhaal: **te veel koppelingen leiden te erg af van de kern van de zaak**. Leg enkel zichtbare koppelingen als die bijdragen tot een betere multimedia-ervaring.

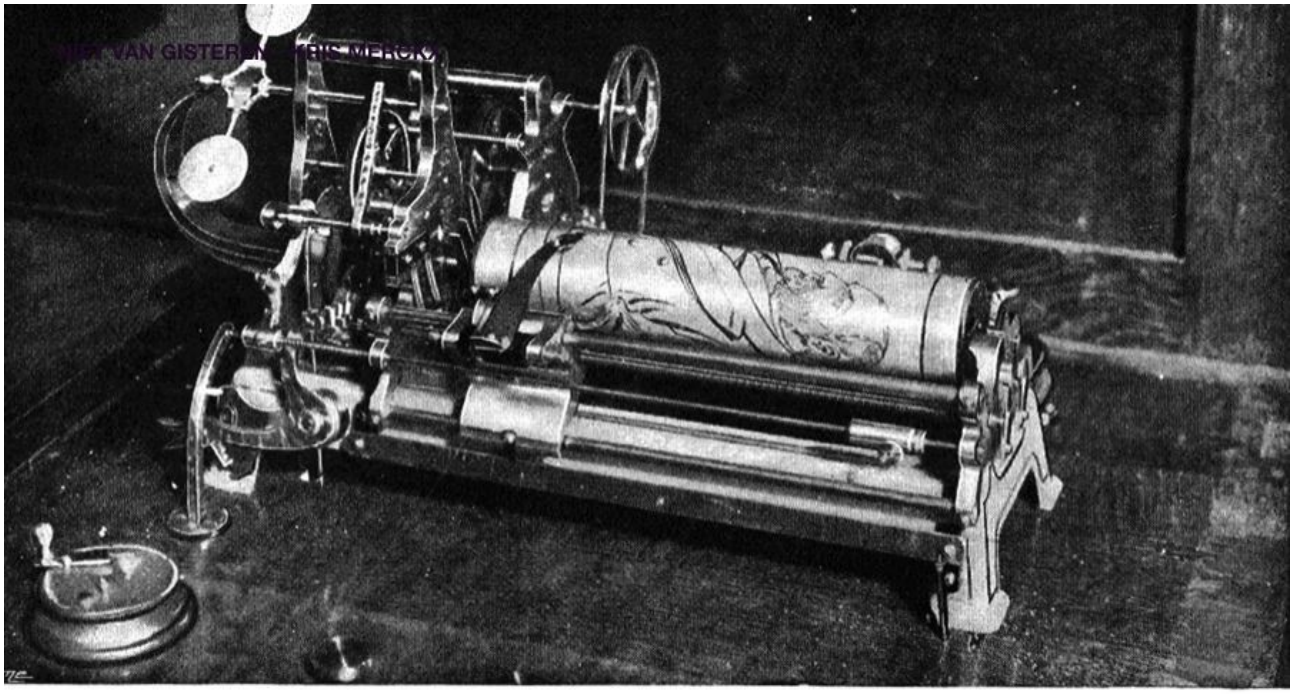
Voorbeeld

Een voorbeeld van de HTML-code van een webpagina: alle media (afbeeldingen, films, audio, stijlbestanden, scripts...) zijn gelinkt. **Ze zitten niet in het bestand, maar zijn extern bewaard.**

```
<!doctype html>
<html lang="nl-be">
        <head>
        <title>Wobbel</title>
        <meta name="author" content="Kris Merckx"/>
        <meta charset='utf-8'>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
        <link href='http://fonts.googleapis.com/css?family=Open+Sans+Condensed:300,700,300italic' rel='stylesheet' type='text/css'>
        <link rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.11.1/themes/smoothness/jquery-ui.css" />
        <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.11.1/jquery-ui.min.js"></script>
        <link rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.11.1/themes/smoothness/jquery-ui.css" />
        <link rel="stylesheet" href="wobbel/basis_links.css" />
        <script src="wobbel/basis_links.js"></script>
<head>
<body>
<h1>Een link</h1>
<a href="http://www.blabla.be" title="een link">Blabla</a>

<h1>Een afbeelding</h1>
<img src="wobbel/architext.png" title="foto bij het artikel">

<h1>Een video</h1>
<video width="320" height="240" controls="">
  <source src="http://www.fotografiejoel.be/film.mp4" type="video/mp4">
        Uw browser ondersteunt de weergave van mp4-film niet.
</video>
<h1>Een audiofragment</h1>
<audio width="320" height="100" controls="">
        <source src="http://www.hetklikt.be/songs/files/beatit2.mp3" type="audio/mp3">
        Uw browser ondersteunt de weergave van mp3 niet.
</audio>
```

5.2.2 Teksten mondiaal verspreiden
Paul Otlet, de Belgische uitvinder van het analoge internet
<RANDINFO>

De telegraaf was het SMS- en mailsysteem van de 19e eeuw. Zowel qua succes als qua impact op de samenleving kende de telegraaf vergelijkbare successen. Met als belangrijkste verschilpunt dat niet iedereen een telegraaf in huis had en "mobiele" telegrafie bestond nog niet. Voor het eerst werd het dankzij de telegraaf (die al 'optische' voorlopers kende in de 17e en 18e eeuw) mogelijk om flitsend snel berichten uit te wisselen over de ganse wereld. Visionaire denkers zagen meteen de toekomstperspectieven die het uitwisselen van informatie via elektrische signalen mogelijk maakte: ooit zou je niet alleen tekst, maar ook beeld en geluid via een "telegraaf" kunnen verzenden.



The transmitter in operation at the *New York Herald* office, showing Croker being telegraphed to the *Chicago Times Herald*, 1000 miles away.

Otlet, één van die Belgen die de hemel ingeprezen wordt in de US en de UK. Hij wordt niet zonder reden de vader van het "analoge" internet genoemd. Alex Wright, professor aan de universiteit van Oxford wijdde in 2014 nog een boek aan Otlet.

De moeder van Paul Marie Ghislain Otlet (1868 – 1944) stierf op zijn derde en tot zijn twaalfde kreeg hij thuisonderwijs waar hij opgroeide als een eenzame boekenwurm. De bibliotheek zou levenslang zijn favoriete omgeving zijn. Niet voor niets ontwierp hij de Universele Decimale Classificatie (UDC) voor het klasseren van boeken. Samen met nobelprijswinnaar Henri La Fontaine (1854 – 1943) vatte hij in 1895 het plan op een bibliografie te schrijven van alle wereldwijd gebubliceerde kennis die hij vervolgens ook publiek beschikbaar wilde maken. Ze beperkten zich niet alleen tot boeken, maar namen ook artikels, magazines, afbeeldingen, pamfletten... media die door bibliotheken in die tijd genegeerd werden, in hun lijsten op. Op die manier bouwden ze een databank op met meer dan 12 miljoen steekkaarten. Van de overheid kregen ze toelating om de databank onder te brengen in overheidsgebouwen te Brussel. Dankzij de overheidssteun konden ze ook

personeel in dienst nemen. Via brief of een telegraafverbinding konden mensen van waar ook ter wereld informatie opvragen. Deze analoge zoekmachine kreeg meer dan 1500 vragen per jaar. Deze analoge zoekmachine kreeg de naam "**Mundaneum**". In zijn boeken "Traité de documentation (1934) en "Monde: Essai d'universalisme" uit 1935 schreef hij over de mogelijkheid van een mechanisch collectief geheugen dat alle informatie zou bevatten en toegankelijk moest zijn via telecommunicatie. Hij droomde van een netwerk ("réseau") van "computers" (hij noemde het "elektrische telescopen") waarin mensen informatie konden zoeken en bladeren door miljoenen gelinkte documenten, afbeeldingen, geluiden en filmfragmenten. In 1934 zei de overheid haar medewerking en steun op. De doodsteek voor dit visionaire Mundaneum kwam door de Duitse invasie. Duitsers gebruikten de originele ruimte als tentoonstellingsruimte voor kunst van het "Derde Rijk" en vernietigden duizenden steekkaarten. Paul Otlet stierf als een gebroken man.

Paul Otlet stond niet alleen met zijn ideeën. De tijd was er rijp voor. De Amerikaanse wetenschapper **Vannevar Bush** (1890- 1974) liep met vergelijkbare ideeën rond. In zijn essay "As We May Think" beschreef hij de **Memex (Memory Extender)**, een toestel dat op basis van microfilms alle belangrijke gegevens zou opslaan. In zijn concept moest de informatie zo worden georganiseerd dat informatie snel zou kunnen worden opgeroepen en geraadpleegd. Het zou leiden tot nieuwe encyclopediesystemen met een "vlechtwerk van associatieve sporen" die je met de Memex zou kunnen uitlezen. Ted Nelson (1937) startte in 1960 met een gelijkaardig concept op basis van computertechnologie dat hij **Project Xanadu** doopte, maar veel verder dat een plan is het nooit gekomen.

</EINDE RANDINFO>

5.2.3 De komst van het digitale internet

De voorganger van het wereldwijde web dat we nu kennen was ARPAnet of het "Advanced Research Projects Agency Networks". ARPAnet was opgezet door het Amerikaanse leger dat in volle Koude Oorlog op zoek was naar een manier om te kunnen blijven communiceren in geval van een nucleaire oorlog. Met dit doel voor ogen werd de TCP/IP-communicatiestandaard ontwikkeld die bepaald hoe informatie binnen een computernetwerk moet worden uitgewisseld. Internet is in feite niet meer dan een verzameling computers die via telefoonkabels informatie uitwisselen. Gaandeweg werden steeds meer 'netwerken' van computers met elkaar verbonden. Hierdoor is het internet een netwerk van duizenden netwerken geworden. Om die computers en netwerken met elkaar te kunnen verbinden, moesten ze ook softwarematig met elkaar kunnen spreken. Hiervoor werd het TCP/IP-protocol ontwikkeld. Elk netwerk en elke computer binnen dat netwerk krijgt een IP-adres (een internet protocol-'adres' bestaande uit een aantal cijfers). Netwerken of computers met een vast (statisch) IP- adres zijn voor alle computers van het internet te bereiken. Zij dienen als server. Binnen een netwerk krijgen computers vaak een dynamisch (= op regelmatige tijdstippen wijzigend) IP-adres. Zulke toestellen dienen als client.

5.2.4 Weven op het web

Door de opkomst van het internet sinds de zestiger jaren van de vorige eeuw, ontstonden allerlei mogelijkheden voor het uitwisselen van informatie zoals e-mail, IRC (internet relay chat), nieuwsgroepen, bulletin boards waar vooral professionele computergebruikers en computernerds en hobby'isten gebruik van maakten. De doorbraak van een wereldwijd informatiesysteem kwam tussen 1989 en 1991. Tim Berners Lee was actief aan het CERN in Geneve. Op zoek naar een manier om eenvoudig informatie te kunnen delen met andere academici en wetenschappelijke instituren bedacht hij het wereldwijde web. **De hypertekstaal HTML (hypertext markup language) moest een standaard worden voor documenten**. Een muisklik op een link in een document, leidt de gebruiker naar een ander HTML-document. Voor het uitwisselen van HTML-documenten tussen computers werd het **HTTP-protocol (hypertext transfer protocol)** uitgewerkt. Tim ontwierp ook de eerste webbrowser, een programma waarmee je webpagina's online kan bekijken. Hij doopte zijn systeem **World Wide Web, kortweg www**. Al snel sprongen ook gewone mensen en bedrijven op de trein door hun eigen websites (een verzameling webpagina's) te publiceren.

5.3 Computers begrijpen jou niet

Als je een tekst invoert op je computer of een tekst publiceert op internet, weet je zelf waarover die tekst gaat. Computers kennen de inhoud niet, of beter gezegd, ze begrijpen absoluut niet waarover je het hebt. Ze snappen de betekenis niet. Voor een computer of stuk software is een tekst niet meer dan een reeks karakters die op hun beurt nog eens worden vertaald in bits (nullen en enen, je weet wel).

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, maakt in je hoofd de juiste analogie en vergelijkingen en 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, de betekenis snapt.

Wanneer je in de zoekmachine Google de zoekterm "haar" invoert, dan moet je niet meteen verwachten dat Google het dichtstbijzijnde kapsalon op het scherm tovert. "Haar" heeft meerdere betekenissen (vb. een bezittelijk voornaamwoord). Meer nog, je voert zoektermen in omdat je beseft dat Google nog minder raad zou weten met de vraag "Wat is het dichtstbijzijnde kapsalon?" Google herkent geen natuurlijke taal, zoals zo goed als geen enkel softwareprogramma eigenlijk. *Daarom moet je ook niet lachen met Google Translate als hij geschifte vertalingen oplevert. Het is eerder verbazingwekkend te noemen wat "Google Translate" wel kan.*

Google en andere zoekmachines snappen totaal niet waarover de inhoud van die miljarden webpagina's op het wereldwijde web gaat.

5.3.1 Hoe doorzoeken software en zoekmachines teksten?

In de meeste programma's kan je met de toetsenbordcombinatie CTRL+F een bepaald woord zoeken. Wanneer je op "zoeken" klikt of op de ENTER-toets, controleert de software de tekst woord per woord (woorden vindt hij door van spatie naar spatie te springen). Wanneer hij een woord vindt dat overeenkomt met jouw zoekterm, licht de software het woord op en vertelt je over het algemeen ook hoeveel overeenkomsten hij heeft gevonden.

Een tagcloud tovert de meest voorkomende woorden in een tekst om in een grafische weergave. Hoe vaker een bepaald woord wordt gevonden, hoe groter het wordt afgebeeld.[48]

48 Afbeelding van http://www.mickcowdanger.dds.nl/easywareSite/wp-content/uploads/2012/01/tagcloud.png, geraadpleegd op 18 september 2014.

Zo dom zijn computers nu ook weer niet. Dankzij "**reguliere expressies**" kan software ook zoeken naar *bepaalde repititieve patronen* in een tekst. Je zou bijvoorbeeld kunnen vragen om alle internetlinks in een document terug te vinden. Dat vraagt al wat ingewikkelder algoritmes om dit te herkennen. De software moet zowel *www.honden.be* als *users.telenet.be/kapsalonmarijke*

en *poesjemauw.interessantesites.nl* terugvinden.

Reguliere expressies zijn zelf voor doorwinterde programmeurs moeilijk om te programmeren.

Een voorbeeld van een reguliere expressie om links terug te vinden:

/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/

Oeps, dat ga je zeker niet op 1-2-3 leren. Besturingssystemen bouwen soms een gebruiksvriendelijke schil over de zoekfunctie, maar hierdoor gaan een aantal mogelijkheden wel verloren.

Als je bijvoorbeeld alle afbeeldingen op een Windowscomputer wil zoeken, dan kan je in de zoekfunctie dit simpelweg aanduiden. Wil je echter enkel JPG-afbeeldingen, dan helpt een instructie als:

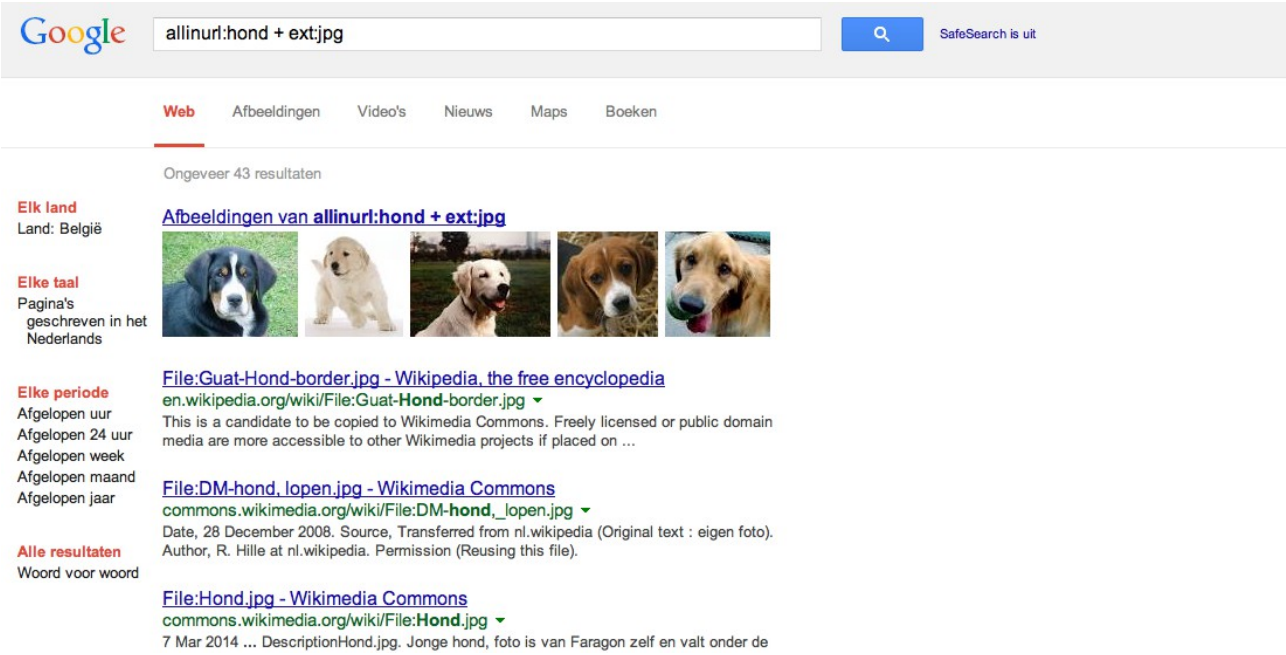
*.jpg

De zoekmachine Google maakt massaal gebruik van reguliere expressies om de teksten op het wereldwijde web te doorzoeken. Je kan ze ook voor een stuk zelf gebruiken via gebruiksvriendelijke instructies of commando's[49]:

De instructie

allinurl:hond + ext:jpg

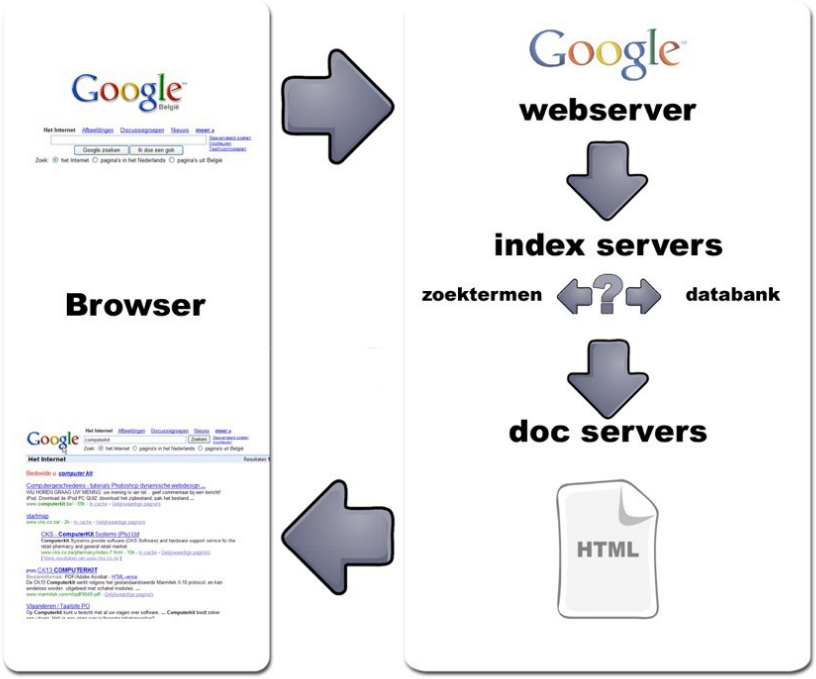
levert afbeeldingen die in hun bestandsnaam of url het woord "hond" bevatten, ook als er bijvoorbeeld "honden" of "hondenkapsalon" staat.



Reguliere expressies worden ook gebruikt in webpagina's om de invoer van een gebruiker te controleren, bijvoorbeeld om te controleren of het door jouw ingevoerde e-mailadres wel een e-mailadres is en niet een hoop onzinnige tekens.

49 Meer voorbeelden op http://www.googleguide.com/advanced_operators_reference.html.

5.3.2 Hoe werken zoekmachines?



Al snel was er op het groeiende web nood aan een zoekmechanisme om in die overvloed van pagina's en informatie een weg te vinden. Zoekmachines zoals Google spelen in op die vraag. Als we het www zouden vergelijken met een boek of encyclopedie dan is Google niet de inhoudstafel, maar eerder de index. Het indexeren van miljarden steeds wijzigende webpagina's gebeurt natuurlijk niet door mensenhanden. Dit zou veel te veel tijd in beslag nemen en zelfs voor een "gratis" dienst als Google onbetaalbaar worden. Het indexeren gebeurt met behulp van een zoekrobot, d e **Googlebot**. Uiteraard is dit geen echte "robot", maar een stuk software. De Googlebot vraagt een webpagina op bij een server. De server stuurt de gevraagde pagina terug. Met **reguliere expressies** zoekt de software naar hyperlinks. Wanneer hij zo'n link vindt, weet de software dat hij opnieuw een volgende webpagina heeft gevonden die hij op dezelfde manier kan doorzoeken. De zoekmachine zoekt dus naar hyperlinks en probeert te ontdekken waarover de inhoud van de pagina gaat, bijvoorbeeld door te controleren of bepaalde woorden vaak voorkomen. *Zie ook 5.3.1 voor meer informatie over zoekmachines.* Zoals je merkt, lijkt de werking van de Googlebot erg op die van een browser, alleen is de Googlebot veel sneller dan een menselijke surfer: hij kan duizenden pagina's tegelijk opvragen. Om een webserver niet te overbelasten, spreidt de Googlebot zijn simultane vraag over meerdere servers en websites. Vindt de Googlebot op de opgevraagde pagina's nog links naar andere pagina's, kan hij ook die opvragen.

Opgelet: begrijp dit niet verkeerd! **Het indexeren gebeurt vooraf.** Google schuimt niet het ganse world wide web af op het moment dat jij een zoekopdracht invoert. De zoekmachine heeft reeds een soort inhoudstafel of index van het web gemaakt en vult die voortdurend aan. Wanneer jij een zoekopdracht intikt, kijkt Google even in zijn databank en vist daar uit welke webpagina's het meest overeenkomen met jouw vraag.

Eenvoudig voorgesteld zou een index van een zoekmachine er als volgt kunnen uitzien:

Zoektermen	*Omschrijving van pagina*	*URL*
hond, Tervuren	Hondenkapsalon Nancy in Tervuren, gespecialiseerd in verzorging van rijkeluishonden.	Www.hondenkapsalonnancy.be
hond, Tervuren	Hondenkennel WoefWoef in Duisburg (Tervuren): voor iedereen die een rasechte straathond in huis wil halen.	Www.woefwoef.net
…	…	...

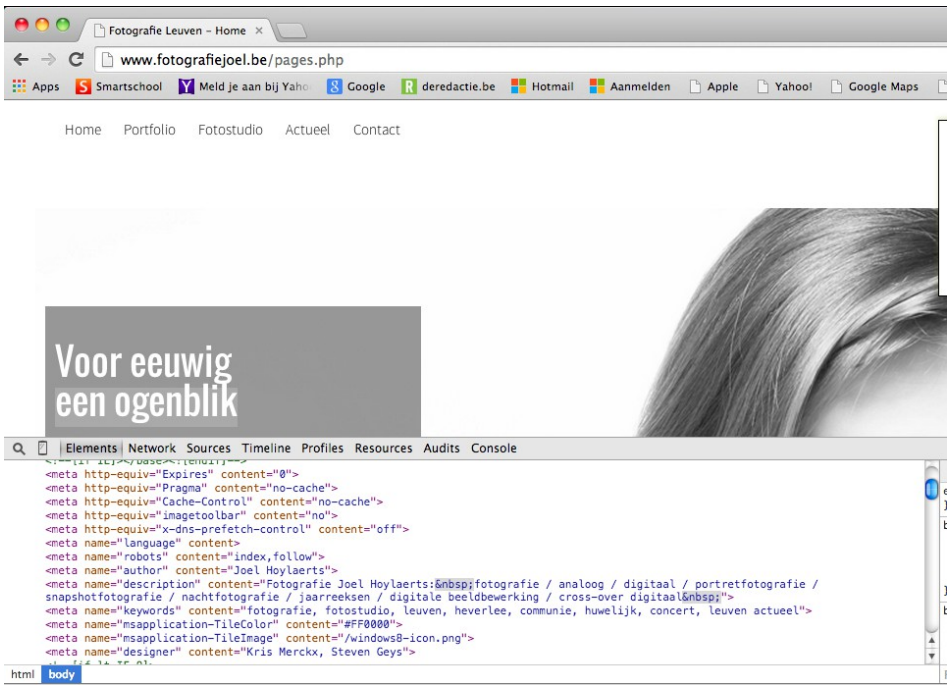
Uiteraard is dit een erg simplistische voorstelling van hoe zo'n index is opgebouwd. In werkelijkheid is de precieze opbouw van de index en de zoekalgoritmes van Google strikt geheim, maar zelfs dan moet je beseffen dat hun databank wat meer bevat dan één enkele langgerekte spreadsheettabel. Maar de tabel hierboven kan

wel een klein beetje verduidelijken (hoop ik toch) hoe een index er voor zorgt dat je ultrasnel zoekresultaten gepresenteerd krijgt.

1. De gebruiker voert de zoekterm "hond" in.

2. Hij krijgt de adressen www.hondenkapsalonnancy.be en www.woefwoef.net.

5.3.3 De noodzaak van "semantiek"



Om tekst en andere media beter toegankelijk te maken en vooral de inhoud ervan te ontsluiten voor software en zoekmachine, zijn er tal van technieken bedacht die het mogelijk maken voor de maker (auteur) of uitgever om het document beter doorzoekbaar te maken.

Bedoeling is om duidelijk aan te geven waarover elk onderdeel van een tekst gaat. Het meest duidelijke voorbeeld vinden we terug in webpagina's.

In het voorbeeld op de afbeelding heeft de webontwikkelaar in de broncode van de webpagina aangeduid waarover de inhoud van de webpagina gaat. Zoekmachines lezen die informatie uit en kunnen op basis van die gegevens jouw site een betere of minder goede plaats geven in de zoekresultaten.

In een webpagina noemt men zulke voor de gebruiker verborgen informatie "metadata".

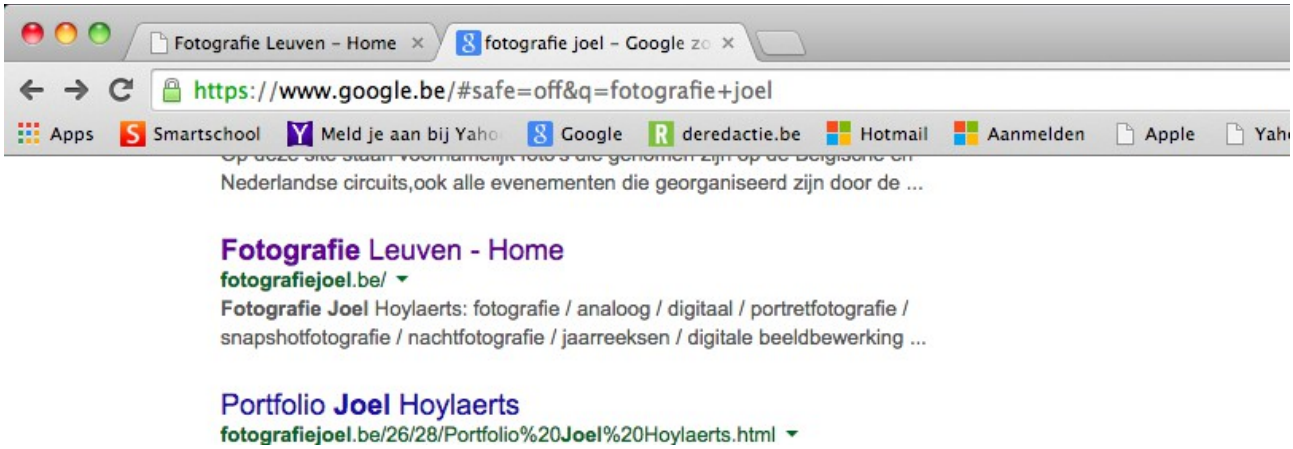
Wat zou de functie zijn van de onderstaande metadata? Bekijk ook de afbeelding met de zoekresultaten in Google.

<meta name="author" content="Joel Hoylaerts"/>

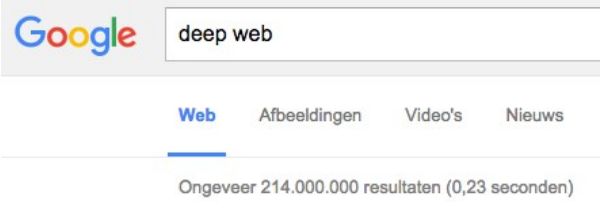
<meta name="description" content="Fotografie Joel Hoylaerts: fotografie / analoog / digitaal / portretfotografie / snapshotfotografie / nachtfotografie / jaarreeksen / digitale beeldbewerking / cross-over digitaal "/>

<meta name="keywords" content="fotografie, fotostudio, leuven, heverlee, communie, huwelijk, concert, leuven actueel"/>

<meta name='designer' content='Kris Merckx, Steven Geys'/>



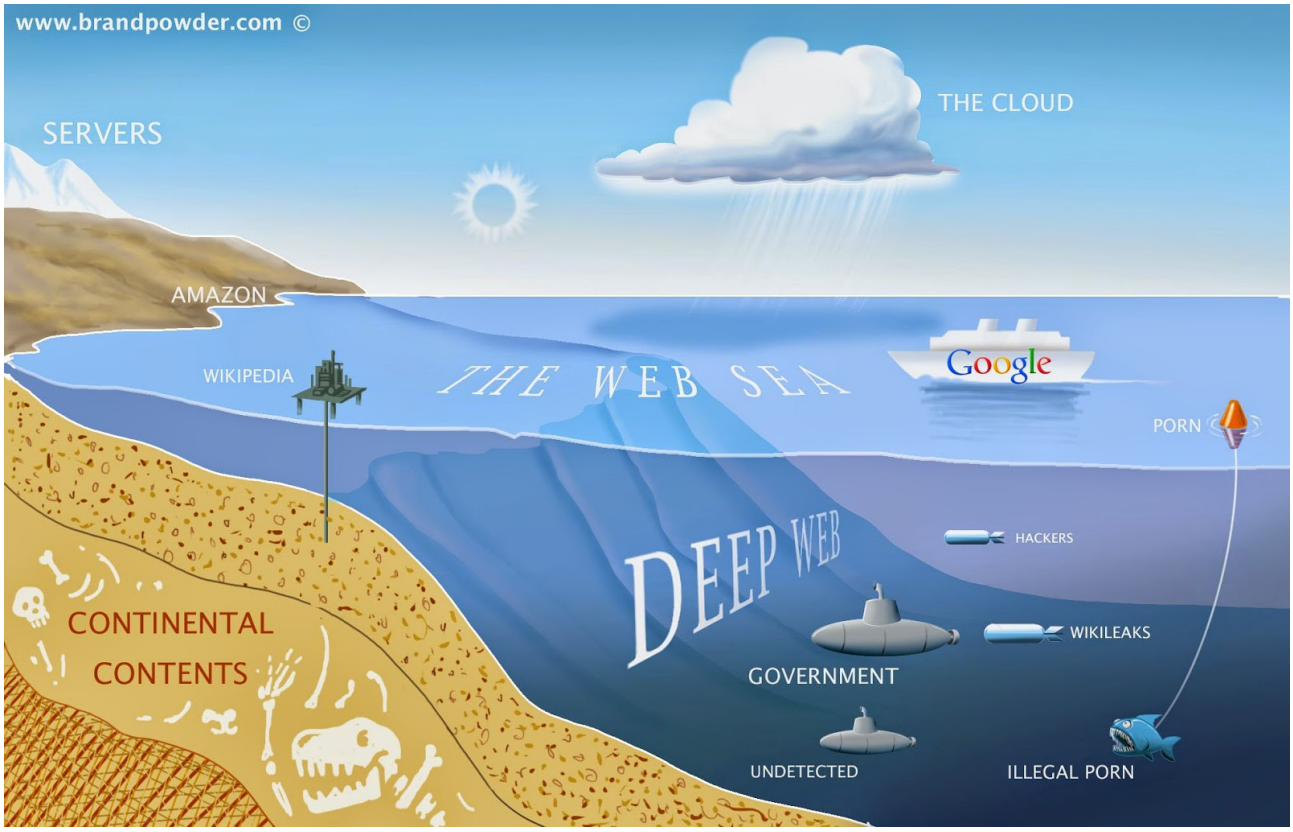
5.3.4 Het deep web



Zoekmachines vinden slechts een fractie van alle op het internet aanwezige informatie. Volgens ruwe schattingen komt in zoekmachines slechts 1/500ste deel van alle beschikbare data naar boven drijven.

De redenen:

1. Heel wat informatie zit verborgen achter gebruikersnamen en wachtwoorden. Digitale leeromgevingen, banken, overheden... publiceren hun data wel op het internet, maar die informatie zit afgesloten en vereist wachtwoordtoegang.
2. Heel veel sites (kranten, winkels, blogs..) stoppen heel veel data in een achterliggende databank. Net zoals bij Google het geval is, bestaan alle webpagina's niet echt als fysieke bestanden, maar zitten alle artikels verborgen in een databank. Wanneer de bezoeker een zoekopdracht invoert, worden de webpagina's door een stuk software op de server, aangemaakt. Vermits zoekmachines hoofdzakelijk op zoek gaan naar "hyperlinks", kunnen ze deze informatie niet (of in elk geval moeilijk) vinden.
3. Heel wat informatie wordt ook bewust verborgen omdat ze illegaal is. **Opgelet**: illegale data komen soms wel bovendrijven. Denk bijvoorbeeld aan de roemruchte Pirate Bay. Die kwam wel aan het oppervlak, ook al stond het netwerk vol met illegaal gekopieerde data.



Bron [50]

Het web telt op die manier drie niveaus: **het surface web, het deep web en het dark web.**

Uiteraard bestaan er technieken om het deep web en het dark web af te speuren. Bedoeling is dat je geen sporen achterlaat, want al je zoekopdrachten (sorry jongens als je "anoniem" surft om speelse sites te ontdekken) blijven zichtbaar. Misschien is elk spoor weg op jouw computer, maar je telecomoperator houdt lijsten van het internetverkeer bij. Anoniem surfen kan wel, maar dan moet je goed weten waar je mee bezig bent en de juiste technieken hanteren. Eén daarvan is TOR (https://www.torproject.org/).

> *"Tor (kort voor The onion router) is een open netwerk voor anonieme communicatie gebaseerd op een techniek genaamd Onion routing. Onion routing is een technologie ontwikkeld in 1995 door het US Naval Research Laboratory. Het Tor-netwerk is bedoeld om te voorkomen dat anderen door analyse van het berichtenverkeer kunnen achterhalen wat de herkomst en bestemming van berichten is."*[51]

50 "Massive deep web links", (securityzap.com), Geraadpleegd op 5 september 2015.
51 TOR netwerk, (https://nl.wikipedia.org/wiki/Tor_(netwerk)), Geraadpleegd op 5 september 2015.

Bron [52]

52 Deep web vs. Dark web, (www.deepwebtech.com), Geraadpleegd op 5 september 2015.

Deel 6:
Output – Feedback voor de mens

Een digitaal systeem kan in principe elke invoer digitaliseren: geluid, tekst, beeld, bewegend beeld, 3D-vormen. Omgekeerd kan het die vormen van data via actuatoren en randapparaten opnieuw genereren.
PC's (personal computers) werden vooral populair toen ze de mogelijkheid begonnen bieden om teksten WYSIWYG te bewerken en af te drukken. Plots kon iedereen voor drukker spelen (ook al bleef de kwaliteit vaak achterwege). Digitale technieken voor beeld- en filmbewerking maken van gewone consumenten in de 21e eeuw prosumers, gebruikers die over professionele mogelijkheden beschikken om eigen "content" te publiceren.
In dit deel bekijken we diverse vormen van digitale uitvoer: tekstverwerking en drukwerk, afbeeldingen, film, animatie, 3D, 3D-print en audio. Omdat veel van die digitale technieken nog aardig wat sporen van het analoge verleden bevatten, duiken we soms terug in de tijd om duidelijk te maken waarom een bepaalde techniek is zoals hij is en welke gevolgen dit nog heeft voor gebruikers in het digitale tijdperk.

6.1 Beeldschermen en kleurweergave

6.1.1 Naar pixels staren

Digitale beeldschermen bestaan uit pixels. Elke pixel krijgt van het digitale systeem een bepaalde kleur toegekend. Het beeldscherm wordt bovendien aan een hoog tempo ververst. Dat wil zeggen dat het scherm voortdurend een ander beeld toont. Indien dit niet zou gebeuren, zou je onmogelijk je muiscursor kunnen zien verschuiven of zou je denken dat je computer was gecrasht als je een programmavenster zou verslepen.

Hoe meer pixels beschikbaar zijn, hoe hoger de beeldkwaliteit is, dat is de redenering van de meeste mensen althans. In werkelijkheid zit het net iets ingewikkelder in elkaar. De grootte van één enkele pixel is nergens vastgelegd. Een groot beeldscherm kan evenveel pixels weergeven als een klein en omgekeerd. Als op een klein beeldscherm meer pixels worden weergegeven, is het beeld veel rijker en zal het er beter uitzien dan wanneer dezelfde hoeveelheid pixels op een groot beeldscherm wordt weergegeven. Op een groot beeldscherm zie je dan al snel een 'trapeffect'.

De maat voor de weergave van pixels wordt uitgedrukt in DPI of dots per inch. Hoe hoger het DPI, hoe meer detail het beeld bevat. Computerbeeldschermen hebben een DPI tussen de 50 en de 100.

Bijvoorbeeld:

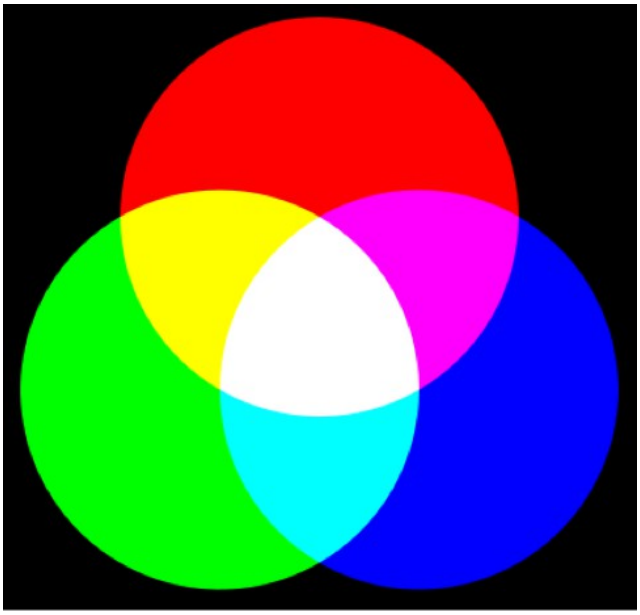
Een beeldscherm van 17 inch heeft met een schermresolutie van 1024x768 pixels een horizontaal DPI van 60,23. Vermits 1 inch = 2,54 cm geeft een computerscherm slechts 23,7 pixels/cm weer.

Wanneer je op een scherm van 23 inch eveneens 1024X768 pixels weergeeft, krijg je een horizontaal dpi van 44,5 of 17 pixels per cm.

Op een groot scherm leiden lagere resoluties dus onvermijdelijk tot slechtere beeldkwaliteit omdat de pixels worden 'opgeblazen'.

6.1.2 RGB voor weergave op beeldschermen

Beeldschermen maken gebruik van **additieve kleurmenging** die ontstaat door het vermengen van 3 lichtbronnen: de primaire kleuren rood (R), groen (G) en blauw (B). Door het mengen van de drie lichtbronnen kan in principe iedere kleur ontstaan. Wanneer ze samenvallen, ontstaat in principe wit (in de praktijk moet de lichtsterkte goed op elkaar zijn afgestemd omdat anders een kleurzweem ontstaat). Elk van de drie **lichtbronnen** gevormd wordt door een enkel lichtpunt, een pixel. Voor het oog vallen de bronnen dan samen.



De term additief komt van optellen, wat bij deze vorm van kleurmenging gebeurt. We beginnen zonder licht (zwart), en tellen daar licht met bepaalde kleuren bij op. Dit in tegenstelling tot subtractieve kleurmenging waarbij het licht geabsorbeerd wordt. De afkortingen voor de primaire kleuren (R=red, G=green en B=blue) bij deze vorm van kleurmenging geven het bijbehorende systeem de naam RGB-kleursysteem.

De kleurdiepte bepaalt hoeveel verschillende kleuren door een enkele pixel kunnen weergegeven worden. Natuurlijk kan één enkele pixel slechts één kleur tegelijk

weergeven! Hoe meer kleuren in een afbeelding kunnen worden weergegeven, hoe rijker de kleurwaarde en hoe realistischer het beeld.

Ontwerp je afbeeldingen voor weergave op beeldschermen, dus bronnen die "licht" uitzenden, dan gebruik je additieve RGB-kleuren.

6.1.3 CMYK voor drukwerk

Bij alles wat licht weerkaatst vb. professioneel drukwerk, tijdschriften... gebruikt men de CMYK-kleurmodus. **CMYK** zijn **subtractieve kleuren**. 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 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

CMYK (Cyaan, Magenta, Yellow, Key) is een systeem om met vier basiskleuren, inclusief zwart, een groot aantal kleuren te kunnen verkrijgen door subtractieve kleurmenging. Dit systeem wordt vooral gebruikt bij **drukinkten**. *Ook komt soms het CMY-systeem voor, waarbij geen zwart wordt gebruikt. Echter in druktechniek leidt dit doorgaans tot te lichte afbeeldingen, omdat de resultante van pure cyaan, magenta en geel donkergrijs is.*

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 adaptieve kleurweergave.*

Het is echter niet zo eenvoudig om drukwerk te ontwerpen dat slechts in 1 kleur van het CMYK-model is opgesteld! Als je bijvoorbeeld een foto van de zon afdrukt, zit er veel geel in, maar in dat geel kunnen tal van kleurcombinaties voorkomen. Het bevat dus meer dan enkel YELLOW.

C	*Cyaan*	bij additieve kleurmenging een combinatie van blauw en groen licht)
M	*Magenta*	een combinatie van rood en blauw licht
Y	*Yellow*	geel: een combinatie van rood en groen licht
K	*Key*	zwart

6.2 Tekstverwerking, DTP en drukwerk

Gedrukte tekst kent een lange geschiedenis. Toen ik zo oud was als jullie, tikte ik mijn eindwerk nog in op een oude typemachine. The times they are a'changin.

1992. Ik maak mijn eindwerk. Voor informatie moet ik op zoek in de bibliotheek of ga ik te rade bij iemand die het weten kan. Teksten schrijf ik met de hand uit. Een buurman heeft een al een computer en zal alles uittikken op een toen reeds fantastische Apple (sorry Windows-fans) in een of andere oeroude versie van Pagemaker, what you see is what you get, anno 1992. Het toestel heeft nog geen retina-display, maar een zwartwit-beeldscherm. Maar dat doet er niet toe. De 200 pagina's waarin mijn eindwerk resulteert rollen uit een *matrix*printer. Een paar jaar later koop ik een computer, op dat moment al een oud ding, met Windows 3.1 en WordPerfect. Mijn moeder snapt die combinatie niet. Het lijkt op een tv met typemachine of net omgekeerd. Maar ze is wel verwonderd. Teksverwerking, het gaat plots anders en sneller, veel sneller. Met tekstverwerkingssoftware kan je heel anders en effectiever schrijven: kopiëren, plakken, bewaren, *na*bewerken, verbeteren...

6.2.1 Van gedrukte tekst tot kopieermachine

Het drukken is geen Europese vinding, maar een Chinese, dat weet stilaan iedereen. De Chinezen drukten niet met losse letters, maar met blokken. De keizerlijke bibliotheek telde in de 7e eeuw dankzij de drukkunst al meer dan 40000 boekrollen. Het oudst bewaarde gedrukte boek is de Diamanten Soetra uit 868.

Marco Polo (1253 – 1324) beschreef over zijn reizen in het oosten dat Koebilai Khan niet betaalde met munten van één of ander edelmetaal, maar met door alchemie bedrukt papier. Papiergeld was echter vaak het slachtoffer van inflatie en het kreeg hierdoor, ook bij Europese reizigers een slechte naam. Niet het papiergeld, maar de speelkaart zou de gedrukte tekst in Europa introduceren. Gedrukte speelkaarten waren in China en Mongolië wijd verspreid. Terwijl de rijke Europeanen hun speelkaarten lieten schilderen, speelde het gewone volk in Duitse en Spaanse kroegen reeds in de 14e eeuw met gedrukte kaarten! Voor Gutenberg startte met het drukken van boeken, werden er in Augsburg, Neurenberg en Venetië al kaarten gedrukt. In 1441 zorgde een Venetiaanse wet zelfs voor een bescherming van de eigen kaartendrukkers tegen import. Ook het blokdrukken was in Europa reeds lang bekend. In een graf van de bisschop van Arles werden bedrukte weefsels uit de 6e eeuw gevonden! Toen Gutenberg met het drukken begon en het westen zich ongewild klaarmaakte voor een volgende informatie-revolutie, was de drukkunst in wezen al eeuwen oud. Enkel de techniek werd verfijnd waardoor hij beter inzetbaar was.

Voor een bedrijf lag het echter nog niet in de lijn van de mogelijkheden om de administratie in gedrukte vorm te archiveren. Daarvoor ontbrak een kleine handzame manier om snel gegevens te kunnen drukken. De

administratie gebeurde met de hand en de pen op papier. In 1714 vroeg de Engelsman Henry Mill (1683 – 1771) een patent aan op de **schrijfmachine**. Maar veel verder is hij niet gekomen, net zoals zovele andere "bedenkers" in de 18e eeuw. Pas in 1860 zou Christopher Latham Sholes (1819 – 1890) uit Milwaukee actief als uitgever, politicus en journalist zijn "Type-writer" voorstellen. Om financiers voor het project te vinden tikte hij brieven op zijn machine en stuurde ze op naar potentiële sponsors. James Densmore zag de mogelijkheden van de schrijfmachine meteen in en in ruil voor 25% van de aandelen, financierde hij de productie. Het prototype deed door zijn **toetsenbord** eerder denken aan een piano en was zeker nog niet rijp voor de markt. In 1870 kwamen ze met een eerste bruikbare model en wisten ze fabrikant Remington te interesseren. Op het eerste model konden enkel hoofdletters worden getikt. Pas later werd de "shift"-knop geïntroduceerd. De eerste modellen werden op een naaimachinetafel gemonteerd zodat je het voetpedaal kon gebruiken om de wagen te verschuiven. Om te verhinderen dat de armen die de letters op het blad drukken in elkaar bleven haken, werden de letters niet alfabetisch geplaatst maar in een volgorde die ervoor zorgde dat de meest voorkomende woorden konden worden getikt zonder dat er zich problemen voordeden. Dit resulteerde in de bekende **QWERTY**- en **AZERTY**-indelingen. Om tussen de spectaculaire uitvindingen op een tentoonstelling op te vallen met hun schrijfmachine gebruikte Remington **knappe jonge vrouwen die de machine moesten demonstreren. Wellicht was hij de eerste die vrouwen inzette in de reclame**. Omdat er steeds meer behoefte was een typisten werden vooral vrouwen in dienst genomen ook omdat de loonkost voor een vrouw een pak lager lag. Sholes was er niet ontevreden over:

> *"Ik voel dat ik iets gedaan heb voor de vrouwen die altijd zo hard hebben moetten werken. Dit stelt hen in staat op een eenvoudiger manier in hun levensonderhoud te voorzien. (...) Ik bouwde wijzer dan ik wist en de wereld heeft er baat bij."*

Een andere belangrijke uitvinding voor de administratie was de **kopieermachine**. Niemand minder dan James Watt (1736 - 1819), de legendarische "uitvinder" van de stoommachine, bedacht een eerste versie. Zijn succes met de verkoop van waterpompen die werden aangedreven door een stoommachine leidde tot stapels papierwerk. Een kopie moest door een kantoorklerk met de hand worden gemaakt. Watt ergerde zich steendood aan de touten die in zo'n kopie konden sluipen en ook aan de tijd die het in beslag nam. Daarom bedacht hij een procédé om teksten mechanisch te kopiëren. De originele tekst moest met een speciaal soort inkt worden geschreven. Vervolgen werd het door een drukpers gehaald waardoor de inkt doordrukte op het kopieerpapier (weliswaar omgekeerd). Via zijn netwerk en de club "The Lunar Society" waar hij deel van uitmaakte zocht hij gelijkgezinden om hem te helpen bij de verbetering van zijn kopieerapparaat. Hij verkreeg in 1780 een octrooi op zijn uitvinding en het toestel werd een groot succes. Naast een vast toestel ontwikkelde hij ook een draagbare versie. Ook andere firma's zouden met gelijkaardige toestellen op de markt komen. De opkomst van de typemachine en de uitvinding van het carbonpapier maakten geleidelijk aan de machine van Watt overbodig. In de twintigste eeuw zagen nieuwe kopieertechnieken het levenslicht zoals de Kodak Photostat en "**blueprinting**", maar de echte doorbraak van het moderne kopieerapparaat kwam in 1938 toen Chester Carlton (1906 – 1968) een

kopieerapparaat bouwde dat gebruik maakte van elektrofotografie. [53]

De kopieermachine van James Watt.[54]

6.2.2 Tekstverwerkers: iedereen is drukker

De komst van tekstverwerkers zoals StarWriter, later WordPerfect en nog later MS Word, was een verademing in vergelijking met het intikken van teksten op een klassieke en/of elektronische typemachine. Vooral de mogelijkheid om snel teksten te kunnen verbeteren, te kopiëren en te plakken was een enorme vooruitgang. Voor het eerst kon men teksten ook digitaal bewaren waardoor men ze achteraf nog kon aanpassen en/of bewerken.

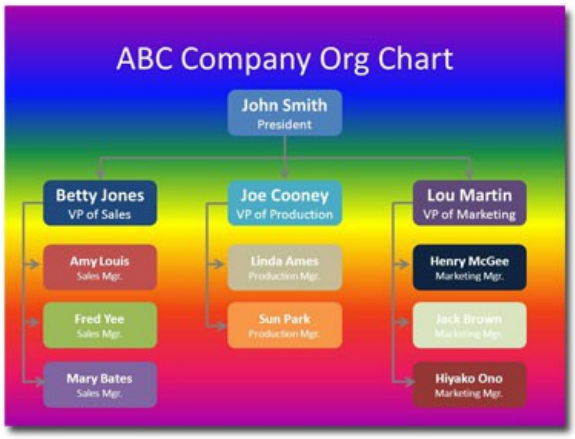
De tekstverwerkers werden en worden steeds uitgebreid met nieuwe mogelijkheden:

- opmaak van de tekst kan worden aangepast (lettertype, lettergrootte...)
- lay-outmogelijkheden (toevoegen van afbeeldingen, figuren, kleuren, schaduwen...)
- structuurelementen en informatieve invoegingen (tabellen, grafieken, titels...)

Al deze functionaliteit heeft er toe bijgedragen dat tekstverwerkers door thuisgebruikers, maar ook door professionele gebruikers voor *verkeerde* doeleinden worden gebruikt (affiches, uitnodigingen...).

Het gebruik van de PC maakt het eigenhandig publiceren, ontwerpen en afdrukken een stuk makkelijker, maar het heeft de 'kwaliteit' van het drukwerk niet altijd bevorderd.

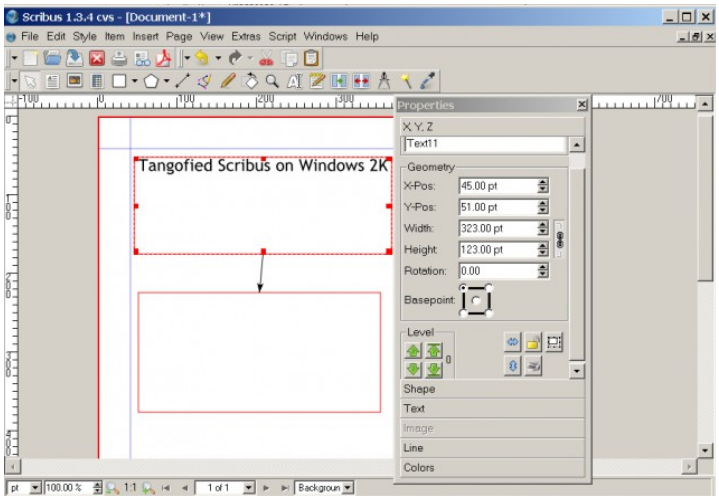
Voorbeeld: iedereen noemt zichzelf ontwerper en drukker. In welke softwarepakketten zijn onderstaande voorbeelden "ontworpen"?



53 Uit: Merckx, Kris, Niet van gisteren: Hoe verbazingwekkende techniek soms duizenden jaren oud is, Davidsfonds, 2013.
54 Afbeelding van http://mikerendell.com/james-watt-inventor-of-the-desk-top-copying-machine/, geraadpleegd op 18 september 2014.

6.2.3 Professioneel werken met tekst

De Xerox Star werd ontwikkeld in de jaren 1970. Deze computer was zijn tijd ver vooruit met het gebruik van de muis, schermiconen, het WYSIWYG (je krijgt meteen het eindresultaat te zien) bewerken van teksten, lettertypes enz. Het Star-recept kwam te vroeg, maar ook de hoge prijs maakte het toestel niet meteen tot publiekslieveling nummer één. Xerox op zijn beurt speelde al leentjebuur bij Douglas Engelbart die ergens eind zestiger jaren de muis introduceerde en demonstreerde hoe je dat onooglijke, toen nog houten, toestelletje gebruikte om teksten te kopiëren en te plakken op een scherm. Zijn naar technologische begrippen oeroude videoconference kan je op internet terugvinden. En ook Douglas was de ingrediënten van zijn mosterd elders gaan opvissen. De oorsprong van **DTP (desktop publishing en het WYSIWYG bewerken van tekst)** is dus niets nieuws. Het is niet iets dat zich beperkt tot MS Word en door Microsoft is uitgevonden.



Professionele gebruikers zoals drukkers, kranten en ontwerpers doen paginaopmaak met desktop publishing software. Over het algemeen gebruikt de wereld van ontwerpers een Applecomputer met een MacOS-besturingssysteem.

Gedurende zeer lange tijd was **Aldus (later Adobe) Pagemaker** het pakket bij uitstek voor paginaopmaak. Een andere grote speler is **Quark Xpress**.

Door het overmatige succes van het grafische programma **Adobe Photoshop** heeft ook het paginaopmaakprogramma Adobe inDesign veel aan populariteit gewonnen. **Adobe inDesign** heeft de andere grote speler, QuarkXPress, zo goed als volledig uit de markt geduwd.

In hun kielzog volgen tal van grote en minder grote software'goden'. Bijna volledig van de markt verdwenen is het overigens uitstekende **Corel Ventura**. De laatste jaren gooit het open sourcepakket en gratis **Scribus** grote ogen (http://www.scribus.net/) (zie afbeelding links).

In een DTP-programma kan de gebruiker ***tekstcontainers*** *aanmaken, tekst rond of langs 'objecten' laten vloeien. Bovendien beschikt het over allerlei mogelijkheden voor het opmaken van tekst, het toevoegen van afbeeldingen enz.*

6.2.4 Drukwerk: offset en digitale print

Een drukker maakt voor elke basiskleur (C, M, Y en K) een aluminiumplaat. Die platen worden dan vochtig gemaakt, waarbij de plaatsen zonder beeld water aantrekken. Vervolgens gaan die platen in een druktoren (1 per kleur) en worden ze op een cilinder gespannen (bij kleinere drukkers is er vaak maar één toren). *"De inkt (Cyaan, Magenta, Yellow of Key) komt in contact met de plaat en de plaatsen die het water hebben aangetrokken, stoten de inkt af. De plaatsen die geen water hebben opgenomen worden gevuld met de inkt. Die inkten worden op een rubberdoek overgedragen, die op haar beurt de inkt op het papier plaatst en het beeld vormt. Het papier gaat langs de Cyaan-, Magenta-, Yellow- en Key-toren, waardoor de kleuren weer gemengd worden, zoals je hebt*

aangegeven in de bestandsopmaak."

Digitale print kan je vergelijken met een "grote" laserprinter. De drukker gebruikt hier geen "inkt", maar toners. Ook hier gebruikt men CMYK, maar zijn anderzijds geen aluminiumplaten nodig. De kleuren worden gebakken op het papier. De startkosten zijn een stuk lager omdat de platen niet meer nodig zijn en het papier moet niet drogen.

Bij offset betaal je vooral per aluminiumplaat en zijn hierdoor de opstartkosten een stuk hoger. Bij digitale print krijg je een stukprijs. Digitale print valt duurder uit dan offset omdat de stukprijs niet evenredig daalt.

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...),

"*Ook de **papiersoort** is van groot belang. Zo absorbeert briefpapier de inkten anders dan bijvoorbeeld een glanzende flyer. Uiteraard moeten de kleuren in de opmaak ook steeds dezelfde CMYK-waarden hebben.*"[55]

6.2.5 POD

"**Short run digital printing** (bij kleine oplagen ook wel **printing on demand** of **POD** genoemd) is een relatief nieuwe technologie, waarbij boeken digitaal worden geprint in plaats van gedrukt. Boeken worden bij deze productiewijze garenloos gebrocheerd (verlijmd). De lijmkwaliteit is in de loop der tijd sterk ontwikkeld, boeken weerstaan ook intensief gebruik. Printing on demand wordt ingezet om oudere, uitverkochte boeken weer beschikbaar te maken of om nieuwe boeken te maken in een kleine oplage.

Bij kleinere oplagen (tot ca. 300 exemplaren) is short run digital printing de meest voordelige productiewijze. Daarboven wordt offset druk kostentechnisch interessanter."[56]

55 AARNINK, F. http://printdeal.be/blog/179/offset-of-digitaal-wat-is-nu-eigenlijk-het-verschil, Geraadpleegd op 24/11/2014.
56 http://www.eburon.nl/wat_is_het_verschil_tussen_offset_druk_en_printing_on_demand, Geraadpleegd op 24/11/2014.

6.3 Fotografie en beeldbewerking

We leven in een beeldcultuur. Geen enkel kind in mijn klas zou zijn lessen geschiedenis of biologie nog graag of makkelijk studeren zonder corresponderende beelden. De nieuwe generatie schoolboeken is rijkelijk geïllustreerd met foto's, tekeningen en afbeeldingen. De aanwezigheid van webcams in computers en camera's in mobiele telefoons maken fotografie en het schieten van beelden eenvoudiger dan ooit.

6.3.1 Camera obscura

De basis voor de fotografie vormt de 'camera obscura' (letterlijk: de verduisterde kamer), een term die de grote astronoom Johannes Kepler (1571–1630) in 1604 als eerste gebruikte. De camera obscura is een verduisterde ruimte, doos of bak die je eenvoudig zelf kunt bouwen met bijvoorbeeld een schoendoos. Aan één kant zit een klein gaatje in de wand. Door dit gaatje projecteert het binnenkomende licht een beeld van de omgeving op de tegenoverliggende wand. Wie zich in de ruimte bevindt of door een kijkgat naar binnen kijkt, zal het beeld van de omgeving ondersteboven geprojecteerd zien. De eerste geschreven vermelding over de camera obscura stamt van de Chinese filosoof Mo-zi (470 to 390 v.C.). Het basisprincipe werd al beschreven door Aristoteles (384–322 v.C.) in zijn 'problemen':

'Het beeld van de zon op het moment van een eclips, tenzij het om een volledige (verduistering gaat), zal, als het door een klein rond gat gaat en op de tegenoverliggende manier valt, de vorm aannemen van een maansikkel. Het beeld van zon zal deze eigenaardigheid enkel vertonen als het gaatje zeer klein is. Als we de opening groter maken, verdwijnt het beeld ...' (eigen vertaling)

Theon van Alexandrië (335–405) vermeldt eveneens het principe van de camera obscura:

'Kaarslicht dat door een opening valt, zal een lichtvlek produceren op een tegenoverliggend scherm die direct in lijn is met de opening en het centrum van de kaars.'

De Chinees Duan Chengshi beschreef in de 9e eeuw van de westerse tijdrekening dat het beeld van de camera obscura ondersteboven verschijnt. Van de Chinese wetenschapper Shen Kuo (1031–1095) is bekend dat hij experimenteerde met de camera obscura. In Europa was het Roger Bacon (1214–1294) die schreef dat de camera obscura de enige veilige methode was om een zonsverduistering te bekijken. Talloze renaissanceschrijvers, waaronder Leonardo da Vinci, Johann Zahn en Athanasius Kircher, bespreken of schetsen de camera obscura. Misschien heeft de techniek ook de integratie van het perspectief in de kunst bevorderd.

Camera obscura, afbeelding uit de 17e eeuw, vermoedelijk Italiaans. (Bron: Wikimedia)

In zekere zin kun je de camera obscura ook zien als een verborgen camera, want hij stelde je in staat om van binnenuit zonder opgemerkt te worden de omgeving in de gaten te houden. Schilders gebruiken de camera obscura al lang om het beeld van de omgeving perfect te kunnen overnemen op hun doeken. Om het beeld recht te krijgen, deden ze een beroep op spiegels. Het is niet bekend welke schilders van de techniek gebruikmaakten. Sommigen zijn van mening dat bijvoorbeeld een Vermeer mogelijk de techniek gebruikte, gezien zijn enorme oog voor detail.

Een variant van de camera obscura is de camera lucida. Aan de tekenplank was een lange staaf bevestigd waarop een prisma vastzat. Als je dit prisma in de juiste hoek afstelde, kon je focussen op het gewenste object. Het licht viel door het prisma heen op de tekenplank en zo kon de tekenaar zijn werk doen.

6.3.2 De uitvinding van de fotografie: licht vasthouden

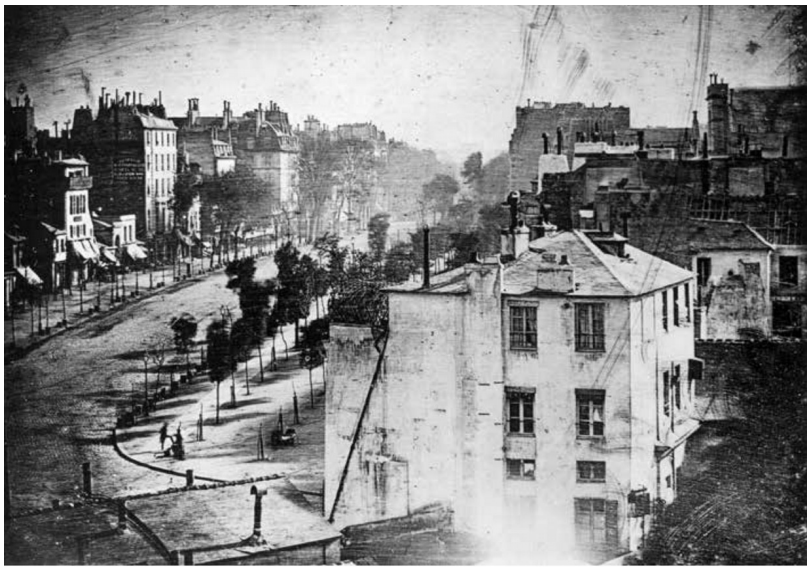
Schilders en tekenaars konden dankzij de camera obscura realistische beelden van de omgeving vastleggen, maar het zou natuurlijk veel tijd en werk besparen als het zonlicht rechtstreeks gefixeerd kon worden op het doek. Joseph-Nicéphore Niépce (1765–1833), zoon van een rijke familie van Franse aristocraten, verloor een hoop bezittingen omdat ze verbeurd werden verklaard door het nieuwe Franse bewind. Toch remde dat hem niet af in zijn wetenschappelijke en technische interesses. Samen met zijn broer vond hij in 1807 een verbrandingsmotor uit met één cilinder en een zuiger. Ook de lithografie of steendruk wekte zijn belangstellng. In 1816 experimenteerde hij met wat hij 'heliografie' (tekenen met de zon) noemde. Hij plaatste een stuk papier dat hij bedekt had met een lichtgevoelige laag zilverchloride in een camera

Niépce maakte deze foto, getiteld 'La Cour du domaine du Gras', vanuit het bovenraam van zijn huis in Saint-Loup-de-Varennes. Hij gebruikte een mengsel van bitumen als lichtgevoelige laag. (Bron: Harry Ransom Center, universiteit van Texas)

obscura, maar het beeld fixeerde niet blijvend. Hij gaf de moed niet op en begon met andere materialen te experimenteren. In 1826 slaagde hij er voor het eerst in om een kopie te maken van een ets. Hiervoor goot hij een laag met Syrisch asfalt op een koperen of tinnen plaat, die hij vervolgens in het zonlicht liet liggen. De witte gedeelten van de ets verhardden en de andere kon hij met lavendelolie wegwassen. Zo verkreeg hij zijn eerste 'heliogravure'. In datzelfde jaar zou het hem ook lukken om een eerste echte 'foto' te maken op een lichtgevoelige tinnen plaat. Doordat de belichtingstijd zeer lang was, veranderden de positie van de zon en de belichting voortdurend, waardoor het contrast en de helderheid van de afbeelding allesbehalve goed waren. Hij slaagde er echter niet in om de techniek wezenlijk te verbeteren. Bovendien hield hij uit angst voor plagiaat zijn procedés angstvallig verborgen. In 1829 wist de Parijse schilder Louis-Jacques Mandé Daguerre (1789–1851) hem te overhalen tot een samenwerking. Daguerre was tevens bedenker van het diorama, samen met het panorama het succesvolste massaspektakel van de 19e eeuw. De grootste uitdaging bestond uit het verkorten van de belichtingstijd. Als ze daarin zouden slaagden, konden duidelijkere en scherpere foto's worden gemaakt. Jammer genoeg zou Niépce die droom niet waarmaken. Daguerre ging na de dood van Niépce verder met experimenteren. Hij bedekte een koperen plaat met een zilverzoutoplossing. Hij stelde die kant bloot aan jodiumpoeder, waardoor een zeer lichtgevoelige laag ontstond, maar ook dat leverde geen bruikbaar resultaat. Toen hij een kast met oude probeersels opende, merkte hij tot zijn verbazing dat er wel beelden tevoorschijn waren gekomen op platen waar toevallig kwikzilver op gevallen was. In 1837 bereikte hij door verdere verbeteringen van zijn methode al zeer duidelijke afdrukken. Het nadeel van dit soort foto, een daguerrotype genoemd, was dat men een metalen plaatje nodig had voor elke foto. Bovendien kon men de foto's niet reproduceren.

6.3.3 Foto's reproduceren

Dankzij het 'calotype', ook 'talbotype' genoemd, van William Henry Fox Talbot (1800–1877) werd het mogelijk foto's af te drukken van een negatief. De Schotse wetenschapper James Clerk Maxwell (1831–1879), die met zijn maxwellvergelijkingen de basis legde voor de elektromagnetische theorie, demonstreerde in 1861 aan het Royal Institution in Londen de eerste kleurenfoto. George Eastman (1854–1932) bedacht in 1885 het fotogevoelige papier. Hij bestreek een lange strook papier met een lichtgevoelige laag gelatine. Daarna wond hij de strook op een spoel, die binnen zijn Kodak-fototoestel weer werd afgewikkeld en belicht. Als de rol helemaal was opgebruikt, moest je ze voor verdere ontwikkeling binnen leveren. Een jaar later kwam een verbeterde versie op de markt, waarbij het papier was vervangen door celluloid. De filmrol was een noodzakelijke stap in de ontwikkeling naar de bewegende film.



'Boulevard du Temple', Parijs – daguerreotype gefotografeerd door Louis Daguerre omstreeks 1838–1839.

6.3.4 Digitale beeldbewerking

Compositing

Grafische software zoals Adobe Photoshop kan niet alleen afbeeldingen en/of rasterafbeeldingen bewerken, maar ook tekenen. De afbeeldingsinformatie wordt beschreven door een 'raster' van verschillende beeldpunten of pixels. Elke pixel kan een bepaalde kleurwaarde bevatten. Doordat de 'kleur'informatie van elke afzonderlijke pixel dient opgeslagen te worden, nemen pixel- of rasterafbeeldingen doorgaans veel meer ruimte in op de opslagmedia. Om de benodigde opslagruimte te verkleinen zijn er diverse compressiealgoritmes ontwikkeld om afbeeldingen te 'comprimeren' (compressie).

De meeste programma's beschikken over de mogelijkheid om in **lagen** te werken. Hierdoor kan de eindgebruiker onderdelen van diverse afbeeldingen selecteren, kopiëren en in diverse lagen boven elkaar plakken. Dit proces noemt men '**compositing**'. Ook kan tekst over afbeeldingen worden geplaatst. Filters en effecten breiden de mogelijkheden nog verder uit.

Softwarepakketten:

- Adobe Photoshop (commercieel)
- Corel Paint Shop Pro (commercieel)
- GIMP (gratis en open source)
- ...

Het werken met lagen vanaf versie 3 (transparante lagen met objecten - tekst, afbeeldingen, kleuren- die boven elkaar kunnen worden geplaatst waarbij de originele afbeelding intact blijft) is een maatstaf waaraan alle andere grafische pakketten worden afgewogen.

Photoshop kan overweg met filters (en ook plugins van derden), maskers, laageffecten enz. Vanaf versie 5 is het programma sterk uitgebreid met functies voor webexport.

De grote kracht van Photoshop zit in het gebruik van lagen. Een laag kan u beschouwen als een transparant vlak dat u 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.

Painting

Paintprogramma's vervangen de klassieke schildersezel. De programma's beschikken doorgaans over alle mogelijkheden van grafische afbeeldingseditors en compositingprogramma's, maar breiden het gamma uit met digitale borstels, potloden, stiften en alle soorten verf...

Deze programma's worden meestal gebruikt in combinatie met een grafische tekentablet ter vervanging (of uitbreiding) van de klassieke muis (vb. Wacom). De grootste speler in deze markt is Corel Painter.

CGI

CGI staat voor 'computer generated images'. Uiteraard geldt dit voor alle met de computer ontworpen afbeeldingen. Toch bedoelt men hier in het bijzonder alle rasterafbeeldingen mee die met behulp van een 3D-programma zijn ontworpen.

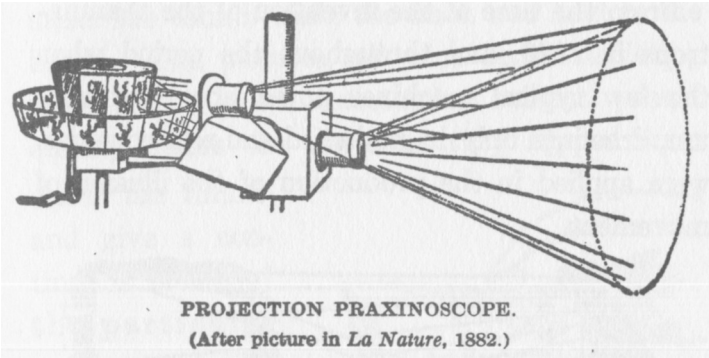
Door velen wordt Photoshop beschouwd als de industriestandaard voor zowel drukwerk en DTP (affiches, flyers, covers..., maar niet voor magazines of boeken) als voor het web wat betreft digitale beeldbewerking.

6.4 Film

Film is een illusie van bewegend beeld. Na de uitvinding van de fotografie ging men al snel op zoek naar een techniek om bewegend beeld te registreren en op een later tijdstip weer te geven. Een van de eerste geslaagde pogingen om beweging vast te leggen, kwam van de Amerikaan Muybridge in 1878. Met een reeks fototoestellen schoot hij meerdere beelden in een seconde van een voorbijrennend paard. Wanneer men die foto's snel na elkaar bekeek met een aangepast apparaat, kreeg men de illusie van beweging. Edison en de gebroeders Lumière waren de eersten om een echte filmcamera te bouwen. Lumière combineerde camera en projector in één. Nauwelijks een paar jaar later zou de goochelaar Méliès de basis leggen voor heel wat professionele filmtechnieken en special effects zoals o.a. stopmotion.

Ouder dan fotografische film is tekenanimatie. Hierbij moeten tekeningen de illusie van beweging scheppen. In de negentiende eeuw zouden apparaten zoals de praxinoscoop beeldsequenties gebruiken om bewegingen naadloos te herhalen: de basis van de moderne tekenfilm.

Naast analoge film deed in de tweede helft van de 20e eeuw de magnetische registratie van film op videobanden zijn intrede. In onze tijd is film ongetwijfeld het meest populaire medium. Dankzij de digitale revolutie ligt het binnen ieders mogelijkheid om professionele films te maken en met de wereld te delen. Enkel uw eigen fantasie en kennis legt de grens...

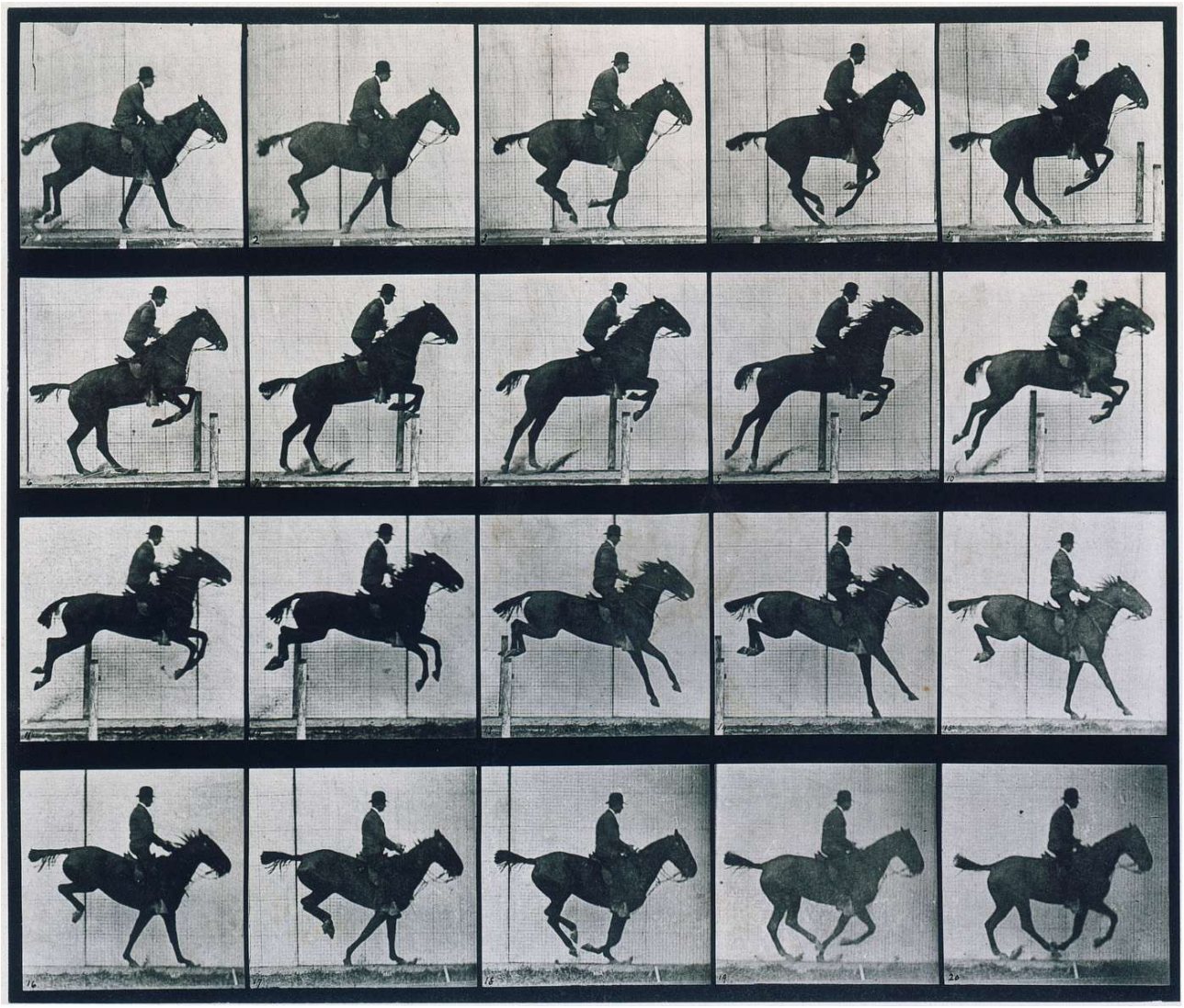


PROJECTION PRAXINOSCOPE.
(After picture in *La Nature*, 1882.)

6.4.1 (Zeer korte) biografie van de film

Voor veel jongeren is de videowebsite Youtube dagelijks voer. Ook volwassenen delen met plezier leuke films via sociale netwerksites. Het hoeft allemaal al lang niet meer zo professioneel te zijn. De gebruiker produceert zelf zijn eigen films. De grens tussen "consument" en "producent" vervaagt en versmelt zodat sommigen zelfs de term "**prosumer**" hebben ingevoerd. Tijdens de Arabische lente waren videowebsites zoals YouTube en sociale netwerksites vaak een enige manier om straatprotest in beeld te brengen en het "echte" nieuws tot in ieders huiskamer te brengen. De film maakte een razendsnelle evolutie door.

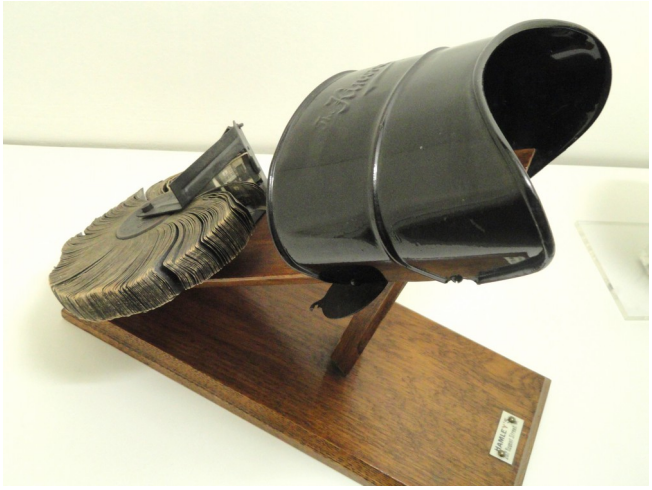
De Amerikaan Eadweard Muybridge (1830 – 1904) slaagde er als eerste in om de belichtingstijd korter dan één seconde te laten duren. Hiervoor maakte hij gebruik van een snelle mechanische sluiter. Tot die tijd bedroeg de



gemiddelde belichtingstijd een paar seconden waardoor bewegende objecten moeilijk waren vast te leggen. In 1878 als eerste in om beweging vast te leggen. Met een reeks fototoestellen schoot hij meerdere beelden van een voorbijrennend paard in één seconde. Wanneer men die foto's snel na elkaar bekeek met een aangepast apparaat, kreeg men de illusie van beweging. Muybridge's "zoopraxiscope" was gebaseerd op de "phenakistiscoop". In 1888 demonstreerde E. J. Marey (1830 – 1904) zijn "chronofotografische camera" die eruit zag als een geweer met een trommel. In de trommel zat een schijf met lichtgevoelige stroken. Dit soort foto's kon men vertonen met de in 1887 uitgevonden "tachyscoop". Op dit toestel dat duidelijk geïnspireerd was op de zoötroop waren de beelden aangebracht op een roterende schijf die je door een kijkgat kon bekijken. Anschütz

bouwde een elektrische versie van dit apparaat waarbij een reeks van 24 beelden werd getoond die werden verlicht door een spiraalvormige Geisler-buis. De Duitse firma Siemens bracht onder de naam "Elektrischer Schnellseher" een muntautomaat met ingebouwde tachyscoop op de markt.

Bewegende film was dankzij de kortere belichtingstijd en de flexibele Eastmanfilm waardoor foto's veel sneller achter elkaar konden gemaakt nog maar kwestie van tijd. Zowel Thomas Alva Edison als de Franse gebroeders Auguste en Louis Lumière stonden vooraan in de strijd om de eerste bewegende film aan het publiek te kunnen vertonen. Edisons kinetoscoop stamt uit 1891 en is duidelijk geïnspireerd op de "peepshow". Door het kijkgat kan de toeschouwer een film bekijken. De filmband was opgerold op een spoel en opgespannen tussen een massa spoelen. W.K.L. Dickson bouwde de eerste filmstudio, de Black Maria in 1894. Al snel kwamen er ook viewers voor thuisgebruik zoals de mutoscoop van Casler en de filoscoop van Short uit 1998. De afzonderlijke beelden werden samengevoegd zoals in het welbekende "**flipbook**" door ze aan één kant aan mekaar vast te maken en er dan met een mechanisme versneld door te bladeren. Ofwel werden ze rond een rotatieas bevestigd. Een gelijkaardig procédé werd toegepast bij de Kinora, het meest succesvolle "home video"-systeem in Engeland voor 1912. De opkomst van de bioscoop en het afbranden van de Kinorafabriek in 1914 betekenden de definitieve doodsteek.



Het verhaal van Louis Aimé Augustin Le Prince (1842 – 1890) is even bijzonder als vreemd en meer dan een verfilming waard. Vijf jaar voor de "uitvinding" van de gebroeders Lumière patenteerde hij een apparaat voor het maken van film en eentje voor de projectie ervan. Hij maakte gebruik van een film met geperforeerde randen die door (maltezerkruis)tandwielen werd voortbewogen. In 1890 filmde hij in Parijs en hij ze daar ook vertonen aan een select publiek. In september van datzelfde jaar bezocht hij zijn broer in Dijon. Hij stapte op de trein naar Parijs en verdween sindsdien zonder ook maar één spoor na te laten.

William Friese-Green (1855 – 1921) werkte verder op het principe van de "biofantascoop" van John Arthur Roebuck Rudge (1837 -1903). De camera maakte vier à vijf beelden per seconden, wat relatief traag is. De beelden werden vervolgens afgdrukt op glazen plaatjes voor de "toverlantaarn" van Rudge. Toen Green celluloid leerde kennen, besloot hij daarmee verder te gaan. Hij patenteerde een aantal modellen maar zijn model uit 1893 lijkt wel heel sterk op het model van F.H. Varley van 1889. Volgens een verhaal dat niet kan bewezen worden, bezorgde hij Edison een uitgebreide beschrijving van de werking van zijn apparaat. Zijn verdere leven was er een van twaalf stielen en dertien ongelukken.

De Amerikaan Jean Aimé Le Roy (1854 - 1932) kondigde op 22 februari 1895, een week voor de eerste voorstelling van Lumière, een "Cinématographe" aan. Lumière had die naam op dat moment nog niet gebruikt. Le Roy kletste misschien wat uit zijn nek, maar was anderzijds niet aan zijn proefstuk toe. Het blijkt echter vooral een goed zakenman. Hij was wellicht de eerste die betaalde "cinemavoorstellingen" organiseerde.

Film hing echt in de lucht in die tijd, klaar om opgevist te worden. Tientallen mensen van verschillende nationaliteiten en ontwikkelden de meest uiteenlopende apparaten die als doel hadden het medium voor bewegende film te worden: getthemoneygraph, chronophotographoscope, counterfivoscope, klondikoscope, vileocigraphiscope... De broers Max en Emil Skladanowsky demonstreerden hun "bioscoop" op allerheiligen 1895 in de Berlijnse Wintergarten. Hun toestel behaalde een snelheid van 8 beelden per seconde en maakte gebruik van dubbelprojectie (bi-oscope).

De gebroeders **Lumière** waren dan misschien niet de eerste filmmakers, toch zou het hun naam zijn dit tot in onze tijd aan de uitvinding van de film en de cinema wordt gekoppeld. Louis Jean Lumière (1864 – 1948) en Auguste Marie Louis Nicolas (1862 – 1954) bestudeerden nauwgezet het werk van Anschütz, Reynaud en **Edison**. In 1895 verkregen ze een patent voor een apparaat dat tegelijkertijd kon filmen en projecteren. De naam "Kinétoscope de (en) projection" werd snel gewijzigd in "Cinématographe". Op 22 maart 1895 gaven ze hun eerste publieke voorstelling in Parijs: "La Sortie des ouvriers de l'usine Lumière". Het hek was van de dam. In geen tijd groeide film uit tot een massamedium. Nauwelijks een paar jaar later zou de goochelaar **Méliès** de basis leggen voor heel wat professionele filmtechnieken en special effects, zoals onder meer stopmotion.

Film groeide uit tot een massamedium en veroverde via de televisie ook het thuisamusement... Film vormt een essentieel onderdeel van elke multimedia-applicatie.

6.4.2 Hoe werkt 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 om de illusie van beweging te creëren. Daarom tonen videoprogramma's naar het aantal uren, minuten en seconden, ook het aantal verstreken frames. De tijdscode (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

...

00:00:00.28

00:00:00.29

00:00:01.00

00:00:01.01

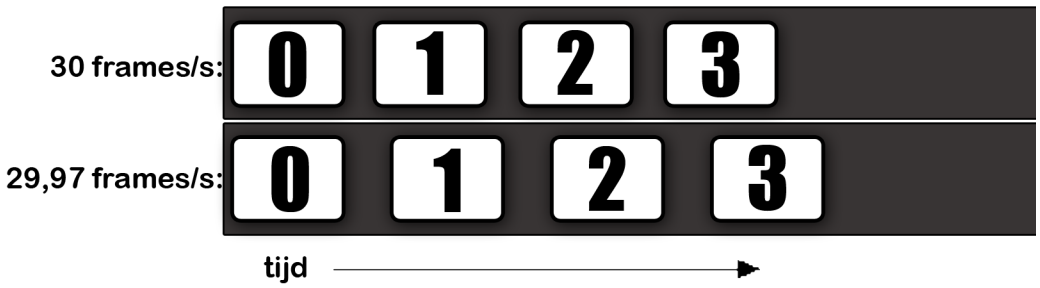
6.4.3 Frames per seconde: 25, 30 of 29,97?

Omdat het elektriciteitsnet in de Verenigde Staten wisselstroom met een frequentie van 60 Hz levert, koos men bij de lancering van zwartwittelevisie voor het uitzenden aan 30 beelden per seconde (60 velden). De 60 Hz-stroom vormde een goede synchronisatiemogelijkheid voor die snelheid. De introductie van kleurentelevisie schepte een probleem. De toegevoegde kleurcomponent interfereerde met het meegestuurde audiosignaal. Omdat het aanpassen van de audio oudere televisietoestellen incompatibel zouden maken, moest men gaan knippen in de bandbreedte van het videosignaal. De beeldsnelheid werd verlaagd naar 29,97 fps. Deze snelheid zorgt nog steeds voor verwarring bij veel amateurs en professionals. Het zorgt echter voor een bijkomend probleem ivm de timecode van een **NTSC** (Amerikaanse)-film.

In Europa levert het elektriciteitsnet stroom met een frequentie van 50 Hz. Het veel later dan NTSC geïntroduceerde en betere **PAL** komt dan ook met een beeldsnelheid van 25 fps (50 velden).

Wisselstroom of AC (Alternating Current) is een elektrische stroom met een doorheen de tijd wisselende stroomrichting. De spanning wisselt tussen 2 punten (uitgedrukt in volt) met een frequentie van 50 of 60 keer per seconde. Die frequentie drukt men uit in Hertz (Hz).

De 29,97 fps van NTSC-film zorgen voor heel wat verwarring. Uiteraard moet je die snelheid niet letterlijk nemen, maar je moet ze ook niet naar boven of beneden afronden! Zoals hogerop reeds gezegd werden elke minuut een aantal frames ge'dropt' om ruimte te maken voor de kleurinformatie. In de praktijk komt het er op neer dat in 100 seconden tijd 2997 volledige beeldframes zijn getoond. De tijd tussen elk afzonderlijk frame bedraagt 1/29,97 seconden of 0,03336666... seconde. De beelden lopen dus een fractie trager dan aan 30 fps, want 0,03336666... is iets langer dan 0,3333 seconde.



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

Het onderstaande stappenplan maakt het wat duidelijker:

1 Opname field 1 (bovenste lijn...)
2 Opname field 2 (lagere lijn...)
3 Mix (=interlacing) van field 1 en field 2 in frame 1
4 Opname field 3
5 Opname field 4
6 Mix (=interlacing) van field 3 en field 4 in frame 2

Analoge camera's en videorecorders registreren eveneens 50 beelden per seconde, maar mixen ze niet in 1 frame (50 not interweaved analog fields per second).

1 Opname field 1 (=frame 1) (oneven scan lines)
2 Opname field 2 (=frame 2) (even scan lines)
4 Opname field 3 (=frame 3) (oneven scan lines)
5 Opname field 4 (=frame 4) (even scan lines)

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 televisietoestel zorgen de interlaced/interweaved velden voor een vloeiender beweging. Wanneer je een interlaced film digitaliseert, levert de software 25 (interlaced) beelden per seconde op. je krijgt dus niet de 50 velden te zien. Met software zoals AviSynth of VirtualDub kan

je de interlaced beelden weer omzetten in velden. **Op een computerscherm (bijvoorbeeld in videomontagesoftware) zorgen interlaced frames voor vervelende lijnen in het beeld**. Wanneer je filmbeelden op een computer maakt (bijvoorbeeld slideshow, animatiefilm...) worden de beelden niet als velden ontwikkeld maar werkelijk als 25 beelden per seconde opgeslagen in het filmbestand. In dat geval spreekt men van **progressive scan**.

DVD-schijven kunnen zowel interlaced beelden bevatten als films in progressive scan. Een bioscoopfilm die rechstreeks vanaf de originele film op DVD is gezet, zal progressive scan-beelden bevatten. Wanneer je zelf met een DVD-recorder beelden van VHS of analoge televisie opneemt, staan de beelden in een interlaced formaat.

Vermits (analoge) televisietoestellen 50 velden verwachten, moet de DVD-speler of het afspeelapparaat progressive scan-beelden weer gaan omzetten in 50 beelden. Bij deze omzetting wordt geen echte interlacing toegepast, maar worden de 25 beelden simpelweg verdubbeld.

Waarom registreert een camera dan niet meteen 50 progressive beelden? Toen televisie aan zijn opmars begon bleek analoge technologie niet in staat om aan die snelheid te filmen en televisie was niet in staat om beelden met zo'n snelheid ook weer te geven. Interlacing is een oude, maar niettemin zeer slimme compressiemethode waarbij de benodigde bandbreedte wordt gehalveerd. In zones van het filmbeeld waarin geen beweging voorkomt, blijft de volledige resolutie behouden. Bij bewegende beelden wordt de resolutie gehalveerd, maar de beweging blijft vloeiend (**smooth**).
Bij heel wat nieuwe camera's kan je in de ***Progressive Mode*** filmen, maar dan filmen ze slechts 12,5 beelden per seconde!

6.4.5 Meer frames zijn niet altijd beter

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 vaak aanvoelt. Wanneer je een dicht bewolkte hemel filmt met vrijwel stilhangende wolken, zal dit zelfs bij een lage beeldsnelheid als vloeiend ervaren worden. 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 uw ogen, ziet je het niet meer scherp maar vaag. Onze hersenen ervaren het niettemin als een vloeiende beweging. Een film met 50 scherpe progressive beelden zou eerder als schokkerig ervaren worden dan een 24 fps-bioscoopfilm met wat **blur** (vervaging) op de beelden!

6.4.6 Flikkerend beeld of flikkervrij?

Onze ogen nemen de werkelijkheid vloeiend weer en delen die niet echt op in frames per seconde. Dit verklaart meteen ook waarom een filmbeeld en zeker een televisiebeeld kan flikkeren. Wanneer het ene frame door het andere vervangen wordt, wordt 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 elke frame 3 keer. Hierdoor worden de zwarte overgangen iets korter en frequenter. Op die manier ziet je in de bioscoopzaal films aan 72 fps... ook al zijn er in werkelijkheid maar 24 progressieve afbeeldingen. Dezelfde techniek wordt toegepast bij een televisiescherm. Hoe hoger de Hz over verversingsgraad, 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 ook van de zwarte beelden tussen de frames.

Onderzoek heeft uitgewezen dat een lichtflits van 1/220e van een seconde door onze ogen geregistreerd wordt. 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.

6.4.7 Containers en codecs

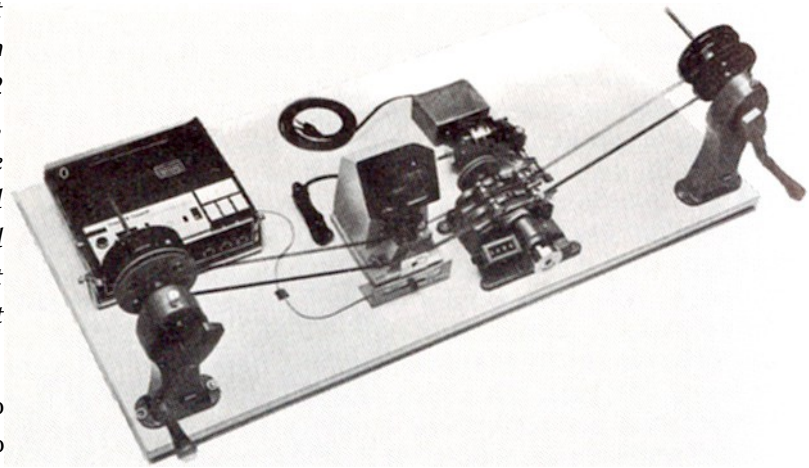
Zelfs doorwinterde videofreaks raken niet altijd wijs uit de terminologie van codecs en containers. Doorheen de jaren zijn er diverse manieren ontstaan om video digitaal te coderen. Een codec is een softwarematige manier om video en audio in bits en bytes (nullen en enen) om te zetten. Een co- dec kan die bits omgekeerd ook weer omzetten in beelden en audio. Codec staat immers voor COderen/DECoderen. Een container bewaart het videospoor en één of meerdere audiosporen in één 'verpakking'. Sommige containers kunnen ook 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. Een AVI-bestand speelt u doorgaans af met Windows Media Player of VLC. Dat is echter geen garantie op succes. Als Win- dows Media Player de gebruikte codec niet ondersteunt, zal de video en/of de audio niet kunnen weergegeven worden!

U kunt niet zomaar elke codec in elke container stoppen, want ook dat kan problemen opleveren. In de onderstaande tabel vindt u een overzicht van de meest voorkomende combinaties.

Container	Volledige naam	Videocodecs	Audiocodecs
3GP/3GP2	3rd Generation Mobile	H.263, MPEG-4, H.264	AMR-NB/WB, (HE-)AAC
AVI	Audio Video Interleave	MPEG-4, DV, MJPEG, Indeo, Cinepak	MP3, MP2, (AD)PCM, AC3
ASF	Advanced Streaming/ Systems Format	Windows Media Video, VC-1, MS MPEG4v3	Windows Media Audio (WMA)
DIVX	DivX Media Format	DivX	MP3, AC3, PCM
DV	Digital Video	DV	PCM
DVR-MS	Microsoft Digital Video Recording	MPEG-2	MP2, AC3
EVO	Enhanced VOB	H.264, VC-1, MPEG-2	(E)AC3, DTS (HD), PCM
FLV	Flash Video	H.263, VP6, H.264	MP3, AAC, ADPCM
M2TS/MTS	MPEG-2 Transport Stream	H.264, VC-1, MPEG-2	(E)AC3, DTS(HD), PCM
MKV	Matroska Video	H.264, MPEG-4	MP3, AC3
MP4	MPEG-4	MPEG-4, H.264	AAC
MPG	MPEG Program Stream	MPEG-1, MPEG-2	MP2
MOV	Quicktime Movie	H.264, MPEG-4, MPEG-1, MJPEG, Sorenson Video	MP3, AC3, PCM
OGM	Ogg Media	Ogg Theora, Xvid	Ogg Vorbis, MP3, AC3
PS	MPEG-2 Program Stream	MPEG-2	MP2, AC3, DTS, PCM
RM(VB)	Real Media (variable bitrate)	Real Video	Real Audio, AAC
TS/TP/TRP/ PVR/VDR	MPEG-2 Transport Stream	MPEG-2, H.264	MP2, AC3
VOB	Video Object	MPEG-2	AC3, DTS, PCM, MP2
WMV	Windows Media Object	Windows Media Video, VC-1	Windows Media Audio

Bron: CT, magazine voor computer en techniek

6.4.8 Soorten montagesoftware



Voor de komst van videosoftware moest een thuisgebruiker zijn videobeelden in volgorde filmen. Monteren met 2 videorecorders was weliswaar mogelijk, maar leverde zeker geen bevredigende resultaten op. Er waren natuurlijk wel toestellen waarmee u uw filmmateriaal kon knippen en mon- teren, maar het gebruik daarvan vereiste toch wel wat kennis.

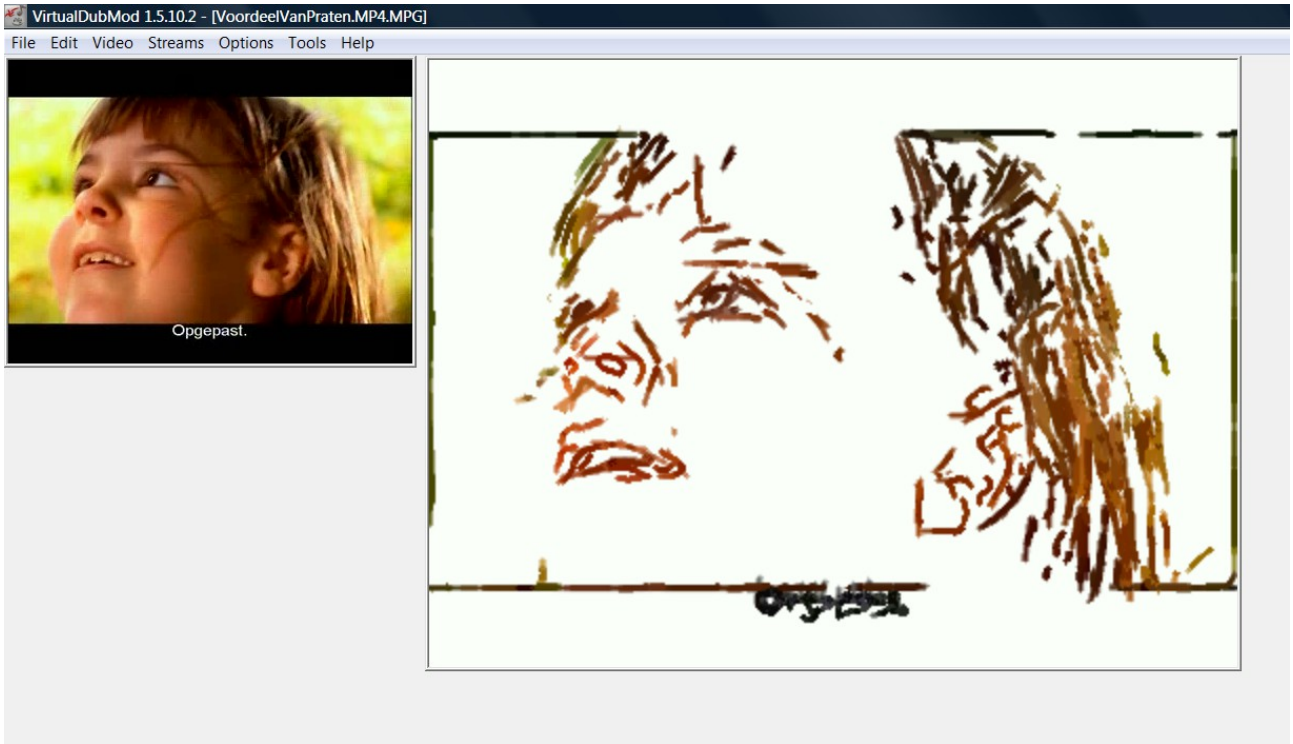
Videosoftware laat gebruikers toe video **non-linear (niet- lineair)** te maken. Zo kunt u meerdere filmbestanden samenvoegen, knippen, plakken en monteren. Met professionele software kan videomateriaal ook geschaald worden en kunnen snelheid, positie en kleuren worden aangepast. Ook het toevoegen van audiolagen en ondertitels wordt er een stuk gemakkelijker op. Belangrijk is dat het bronmateriaal steeds onaangetast is. Deze mogelijkheid noemt men dan ook non-destructive editing (niet-destructief), omdat het bronmateriaal niet echt verknipt of aangepast wordt.

Videomontagesoftware kan ruwweg in vier grote categorieën worden onderverdeeld:

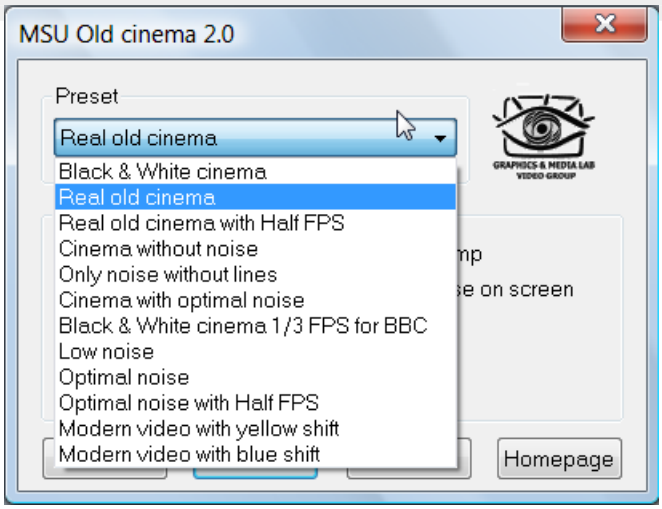
1. Eenvoudige montagesoftware met één videolaag, vaak gratis of toch zeer goedkoop (bv. Windows Movie Maker, iMovie...)
2. Montagesoftware voor thuisgebruikers met één videolaag of meerdere videolagen. Prijzen in de orde van 100 tot 200 euro. Afhankelijk van het pakket worden er min of meer professionele mogelijkheden ingebouwd. Soms gaat het om een eenvoudige versie van een gelijknamig en duurder professioneel pakket (bv. Adobe Premiere Elements, Magix Movie Edit Pro...)
3. Dure professionele software met uitgebreide mogelijkheden (bv. Adobe Premiere, Final Cut...). Prijzen schommelen tussen 250 en meerdere duizenden euro.
4. Open source of vrij te gebruiken professionele software. Rijzende ster is zonder enige twijfel LightWorks.

Wanneer u een softwarepakket wilt gebruiken en/of aankopen, stelt u zich best de vraag welke opties voor u belangrijk zijn. Heel wat mogelijkheden zitten ook in goedkope en zelfs gratis software. Andere functies vindt u enkel in de duurste pakketten. En ook al vermoedt u dat u die geavanceerde functies toch nooit zult gebruiken, ze bieden natuurlijk wél fantastische mogelijkheden. Het belangrijkste blijft de meerwaarde die een videosoftwarepakket u kan bieden.

Effecten gebruiken voor het effect is uit den boze. U moet een effect enkel toepassen omdat u een bepaald doel voor ogen heeft. Wat betreft het monteren van video geldt het motto 'overdaad schaadt'. Het verhaal moet immers altijd belangrijker zijn dan eender welk effect u gebruikt.



Een video-filter in het gratis VirtualDubMod.



Video-effecten.

Het onderstaande overzicht kan u op weg helpen bij uw keuze.

1. Hoeveel video- en audiolagen kunt u over elkaar plaatsen?

De meeste pakketten werken op een gelijkaardige manier. Geïmporteerde filmbestanden, audio en afbeeldingen komen in een soort bibliotheek terecht. Vervolgens kan de gebruiker de geïmporteerde bestanden naar het video- spoor en het audiospoor op de tijdlijn slepen en ze daar ordenen. De fragmenten kunnen daar nog verder worden verknipt of aangepast, er kan gesleuteld worden aan het geluidsniveau...

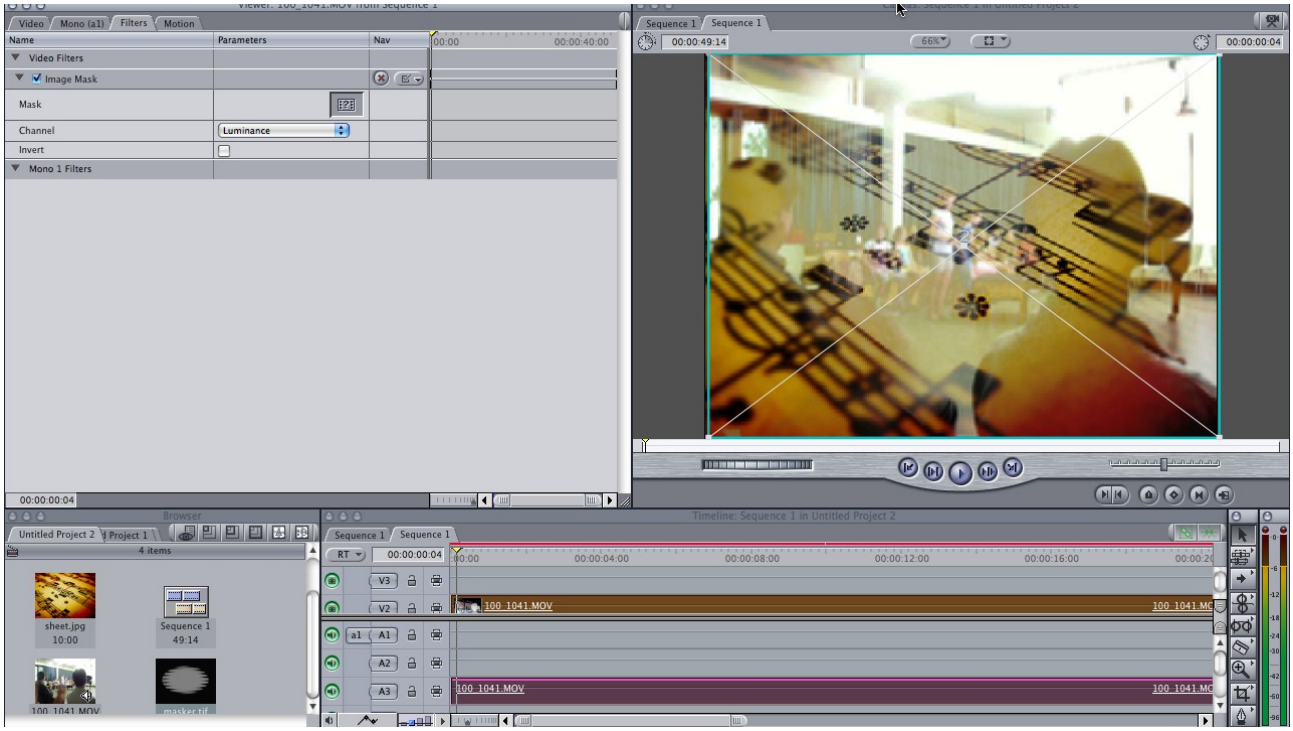
De tijdlijn in het gratis Cinelerra (Ubuntu Linux)

Vaak kan men de **weergave van de tijdlijn** aanpassen.

- Filmfragmenten als een afzonderlijk blokje met thumbnail.
- Filmfragmenten als een reeks thumbnails. Dit vraagt veel meer werkgeheugen.
- Fragmenten als rechthoekige blokken met een lengte in verhouding tot hun tijdsduur.
- Een paar programma's geven de tijdlijn weer als een blok van afbeeldingen (iMovie, Pinnacle Studio,...). De meeste applicaties tonen de tijdlijn als een lange, verschuifbare ho- rizontale balk.

In de meeste programma's kunnen meerdere audiosporen worden toegevoegd, maar slechts één videospoor. Sommige gebruikers vragen zich af waarom er meerdere videolagen over elkaar geplaatst worden, als men toch enkel de bovenste kan zien. Een onterechte vraag... Door de lagen te schalen, kunt u immers picture-in-picture-effecten maken, chromakey-technieken toepassen, lagen animeren en heel wat meer.

De voordelen van "meerdere videosporen" in Final Cut Express:



2. Transities of overgangseffecten

Met behulp van transities kunt u diverse filmfragmenten in elkaar laten overgaan of overvloeien. Deze overgangseffecten zijn een lust voor het oog en zorgen bij veel gebruikers in het begin voor een wow-effect. Toch maken transities het eindresultaat niet altijd professioneler. Let er maar eens op in hoeveel – of beter: hoe weinig – DVD-films of tv-reeksen transities worden gebruikt. Als u kiest voor een overgang, probeer die dan over heel uw montage te gebruiken. Het is tenslotte niet de bedoeling om alle effecten in één filmpje te tonen. Het verhaal moet primeren. Een simpele droge cut werkt vaak veel beter dan een film die je huiskamer binnendendert...

3. Templates, sjablonen, thema's...

Sommige softwarepakketten (bijvoorbeeld iMovie HD en iMovie 9) bevatten een aantal geanimeerde sjablonen waar- in u filmfragmenten of foto's kunt slepen. Dit oogt vaak mooi- er en professioneler dan klassieke transities. Ook hier geldt dat eenheid van stijl belangrijker is dan een opeenstapeling van effecten. In de duurdere pakkettten is deze mogelijkheid vaak niet meer aanwezig, maar daar beschikt u als gebruiker

natuurlijk over alle mogelijkheden om de sjablonen zélf te creëren. Hiervoor moet u natuurlijk wel goed onderlegd zijn in grafische software en animatie.

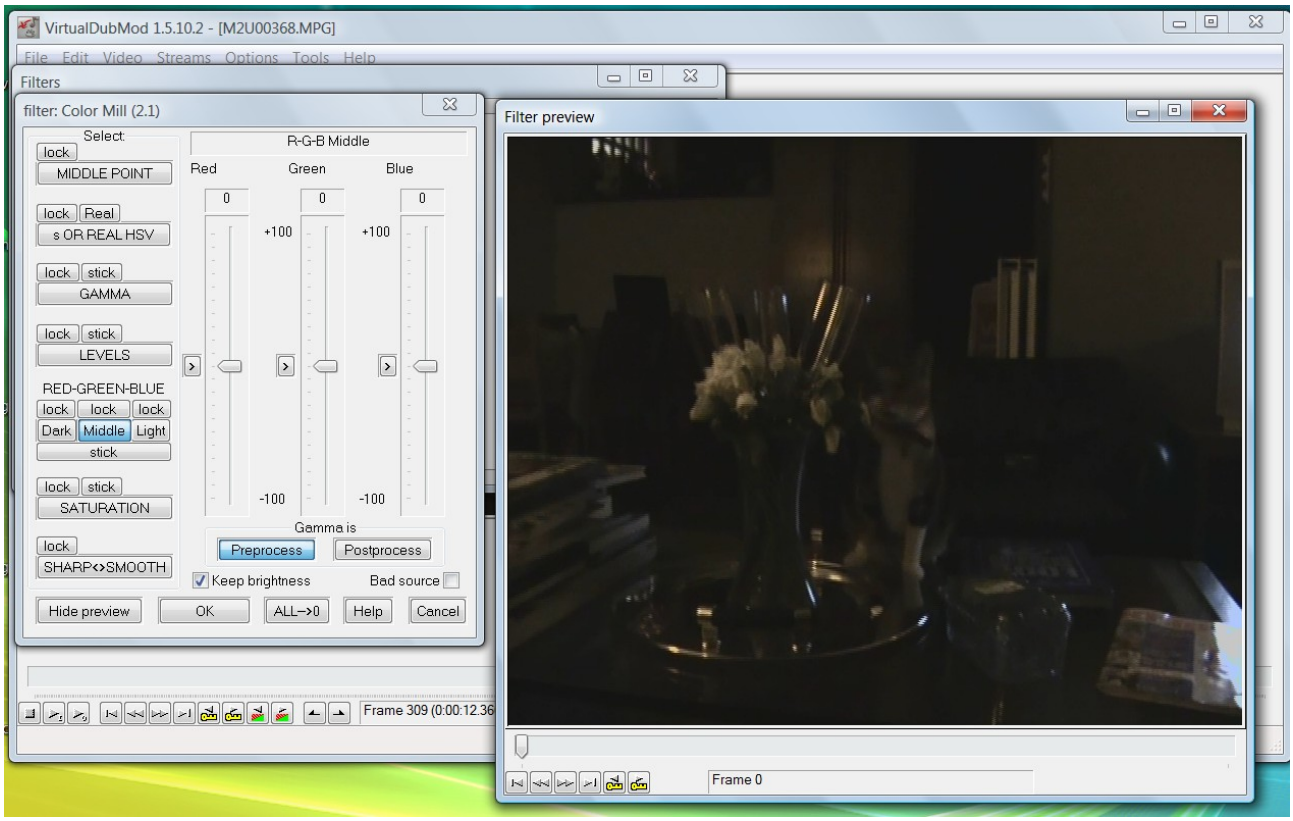
4. Audio

In zowat alle videomontagepakketten kunnen meerdere audiosporen worden ingevoegd. Naast het geluid van de film, kunt u dus gemakkelijk een een voice-over of muziek toevoegen. Het is meestal ook mogelijk om het audiospoor los te koppelen van het videospoor (**demuxen**, extract audio) en het audiospoor te verwijderen of te vervangen. In heel wat software vindt u een reeks geluidseffecten en/of audiofilters voor het bewerken van het geluid.

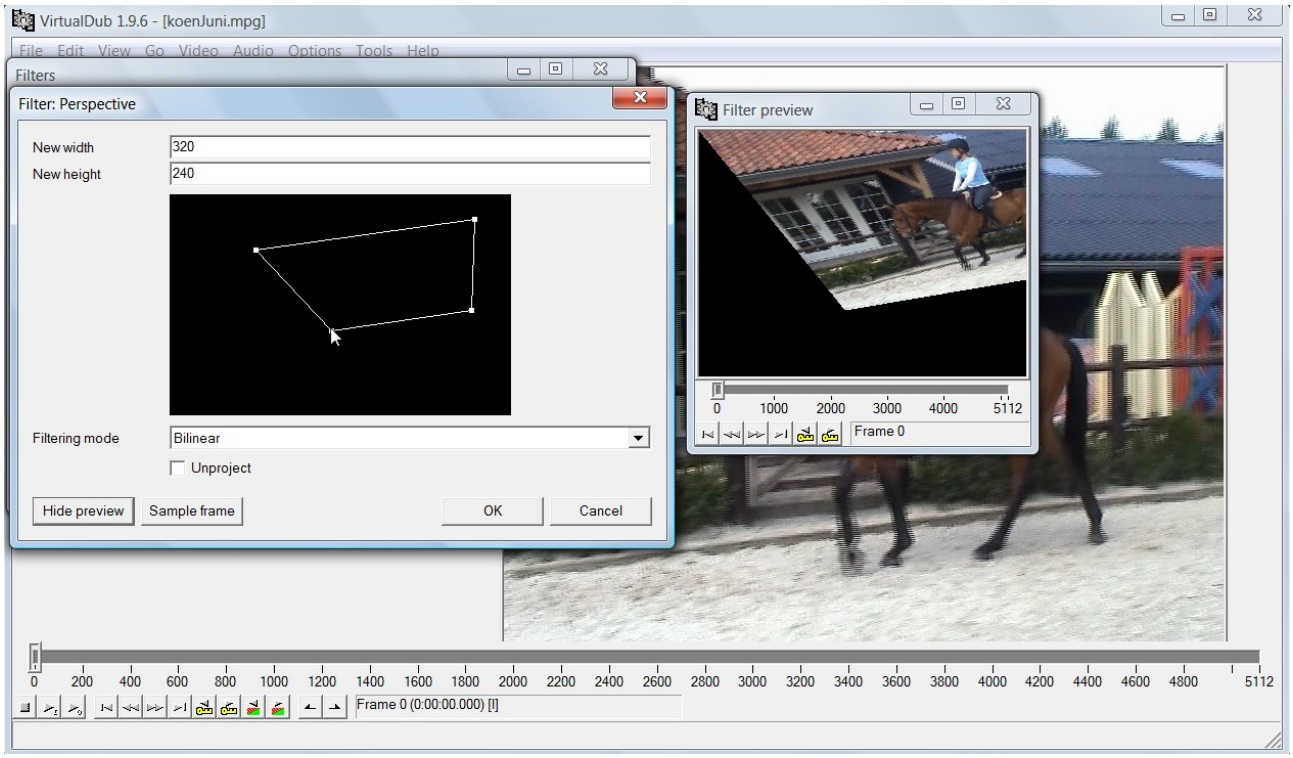
5. Video-effecten en video bewerken

Zelfs de goedkoopste of gratis software heeft over het algemeen een pak video-effecten aan boord. Meestal laat een video-effect zich simpel naar de geselecteerde clip (het filmfragment) slepen. Met duurdere software kunt u de effecten doorheen de tijd animeren met keyframes.

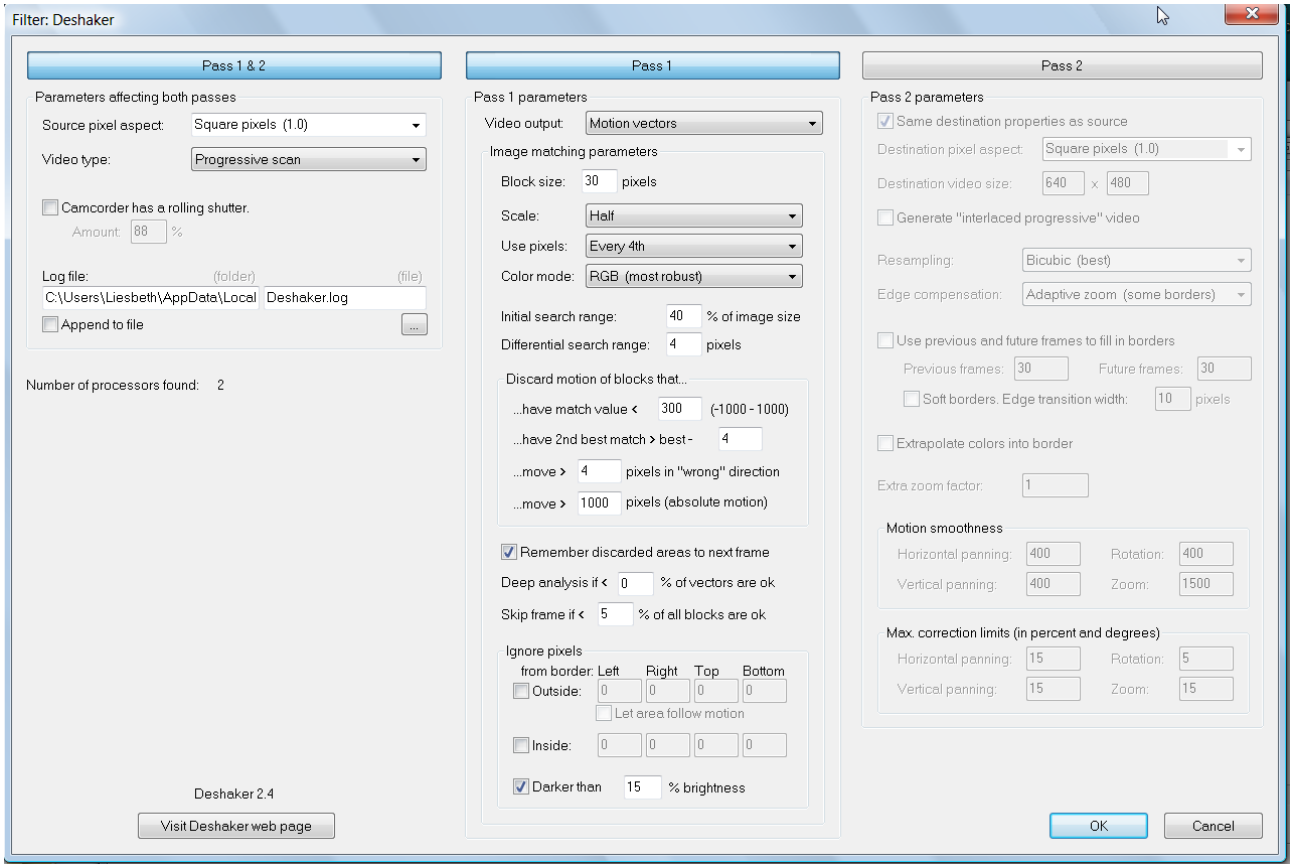
Kleuren bewerken in het gratis VirtualDub:



Film bijsnijden in VirtualDub.



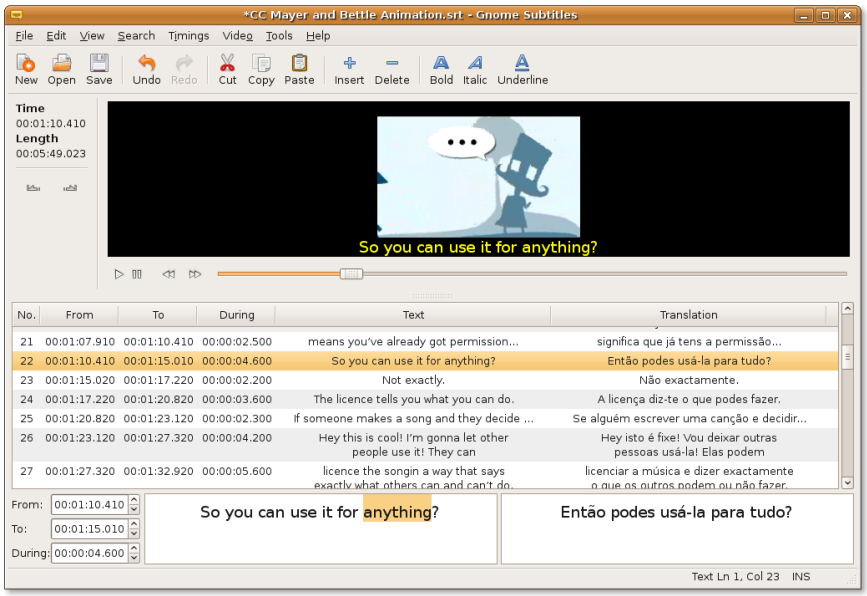
Schokkerige filmbeelden stabiliseren met Deshaker:



5. Titels en ondertitels

Naast een tijdlijn, transities en video-effecten zijn titels een vast onderdeel van videomontagesoftware. Professionelere pakketten bevatten vaak een apart hulpprogramma voor het samenstellen van titels. Let echter goed op: wanneer u film monteert voor een klassieke televisie, moet u rekening houden met het title safe-gebied (zie deel 1). In professionele programma's kunt u dit gebied weergeven, maar met low budget-programma's is dit niet altijd mogelijk. Wanneer u een titel te dicht bij de onderkant of de zijkanten van de film laat beginnen, bestaat het risico dat hij niet op een tv kan weergegeven worden. Dit probleem wordt echter steeds minder groot. Softwarematige mediaspelers, digitale tele- visies en lcd-schermen hebben dit probleem namelijk niet.

Met de meeste videomontage-programma's kan geen ondertiteling worden ingevoerd. Oorzaak hiervan is dat onderti- telingssporen meestal apart op een DVD worden opgeslagen, en niet in het videospoor.



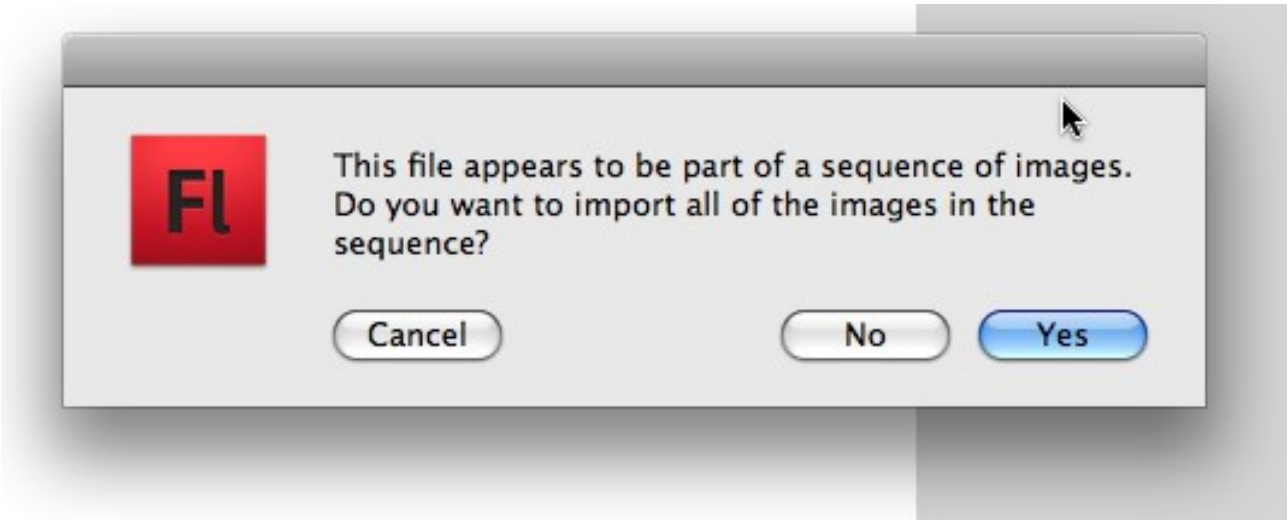
Ondertitels met Gnome Subtitles (Linux).

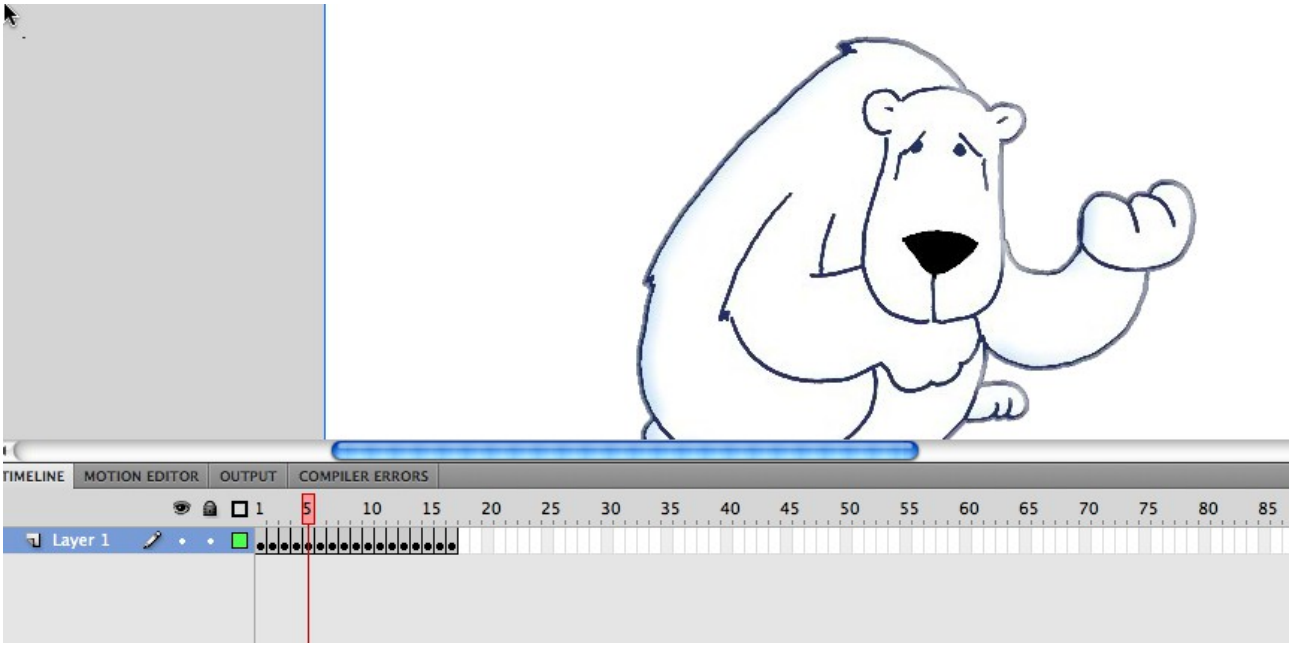
6. Film van foto's of omgekeerd

Heel wat programma's staan het importeren van genummerde reeksen afbeeldingen toe. Op die manier kunt u een film maken die gedeeltelijk of uitsluitend uit foto's bestaat. Zo maakt men bij documentaires, waarvoor men niet altijd over bewegende beelden beschikt, dankbaar gebruik van het Ken Burns-effect: zachtjes inzoomen op een foto, waardoor de aandacht van de kijker op een bepaald detail wordt gefixeerd.

Omgekeerd kan je met heel wat software een film ook exporteren als een genummerde reeks afbeeldingen (MPEG Streamclip, Adobe Flash, Adobe Photoshop, Processing...). In een programma als Adobe AfterEffects kan je de reeks afbeeldingen importeren. Dat is ook erg handig als je afbeeldingen maakt via *stop motion*.

Een afbeeldingenreeks importeren in Adobe Flash:

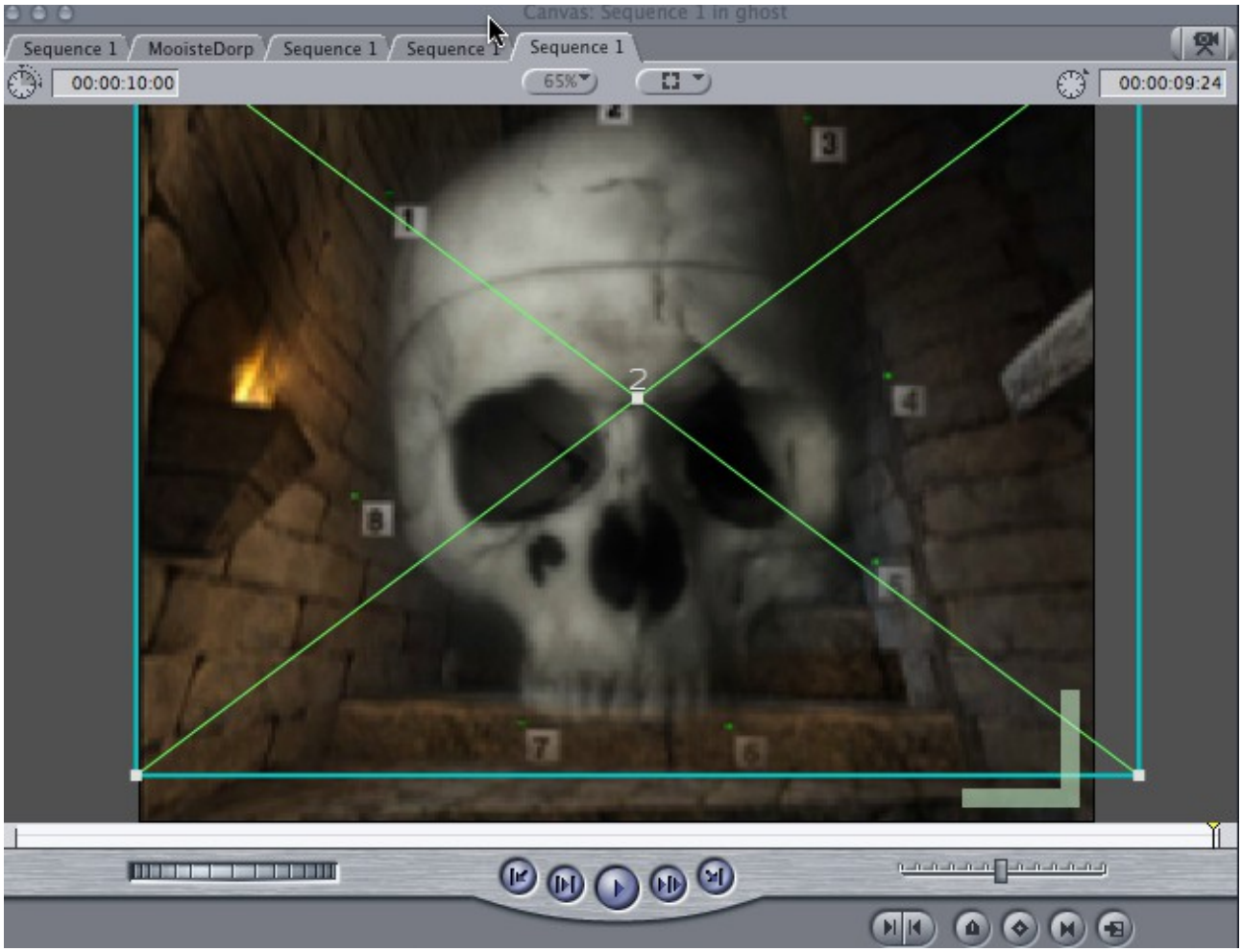




7. Compositing

Compositing betekent het maken van composities met afbeeldingen en film. Adobe Photoshop en The GIMP zijn bijvoorbeeld beeldbewerkingsprogramma's, maar hun belangrijkste doel is compositing. In meerdere lagen kunt u afbeeldingen of delen ervan boven elkaar leggen, effecten toevoegen enz. De ontwerper kan lagen met een verschillende graad van vulling of transparantie op elkaar stapelen, laten doordrukken of overvloeien. Videomontagesoftware waarin u meerdere videosporen over elkaar kan leggen, biedt soms ook compositingmogelijkheden. Adobe AfterEffects en Apple Motion zijn bekende en "straffe" voorbeelden van video-compositing-programma's.

Compositing, chromakey en maskers animeren in Final Cut.



8. Animatie

Heel wat videomontagesoftware beschikt over mogelijkheden om videolagen of afbeeldingen via keyframes te animeren (zie volgende onderdeel).

Animatie langs een "pad" in Adobe Flash. Zie 6.5 Animatie

9. Rendering

Veel video-amateurs krijgen koude rillingen als ze aan **renderen of rendering** denken. Wanneer u filtereffecten of transities aan filmbeelden toekent, moeten de beelden meestal eerst *gerenderd* of ontwikkeld worden. Dit betekent dat de software eerst de filter op elk beeld moet toepassen vooraleer de gebruiker het effect van de filter of transitie kan zien. Aan de hand van een voorbeeldanimatie toont de meeste software vooraf welk resultaat het effect zal hebben. Rendering kan bijzonder veel tijd in beslag nemen. De volgende factoren kunnen bepalend zijn voor de snelheid van het renderproces.

- **De gebruikte hardware:** snelheid van de processor, hoeveelheid RAM-geheugen, snelheid van de GPU (grafische processor). Vreemd genoeg maakt niet alle videosoftware gebruik van de kracht van de grafische processor om de beelden te renderen.
- **De software:** sommige pakketten (meestal de wat oudere) renderen enorm traag.
- **De gebruikte importformaten:** zorg er voor dat uw **footage** (het beeldmateriaal dat u importeert) in een containerformaat en codec staat waarmee uw videomontagesoftware vlot kan werken. Het oudere Final Cut Express werkt bijvoorbeeld vlot met QuickTime-mov en DV, maar kan soms uren doen over het renderen van MP4-materiaal. Bij een MP3-audiospoor geeft Final Cut Express zelfs met een bieptoon aan dat het niet met dit formaat overweg kan.
- **De tijdsduur van het effect of de transitie:** Hoe langer het fragment waarop de filter is toegepast, hoe langer het renderen zal duren.

10. Import- en export

De import- en exportformaten zijn niet het grootste probleem. Met conversiesoftware (o.a. VLC, Super, MPEG Streamclip, Ffmpeg, Handbrake...) kunt u er altijd voor zorgen dat het beeldmateriaal wordt geconverteerd naar een formaat dat door uw videomontagesoftware kan worden ingelezen. Wilt u hoogstaand videomateriaal maken (voor DVD, Blu- ray, HD...), dan moet u er natuurlijk wel voor zorgen dat de bewuste software hiermee overweg kan.

Exporteer de eindmontage naar een zo hoog mogelijke kwaliteit. Indien u de film voor meerdere doeleinden wilt gebruiken, kunt u met conversiesoftware uw film nog naar diverse andere profielen of formaten exporteren.

6.5 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 zelf ze inzetten in uw filmprojecten? Hoe gebruikt je professionele filmtechnieken zoals chroma-key en masking zonder massa's geld te moeten neertellen voor dure software?

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, 3D... 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.

Animatie voor een toverlantaarn, 19e eeuw:

6.5.1 Animatieprocédés

Professionele animatoren onderscheiden diverse vormen van animatie:

Animatie	*Kenmerken*	*Voorbeeld*
full animation	Een tekening per 1 of 2 frames Traditionele animatiefilms van hoge kwaliteit.	Walt Disney, Warner Bros...
limited animation	Diverse onderdelen van een tekening kunnen meerdere frames duren. Minder gedetailleerd. Kan schokkerig, maar ook zeer gestileerd overkomen. Vaak gebruikt voor (goedkopere) TV-producties, anime en webanimatie...	Flinstones
Rotoscoping	Live-actie beeld per beeld traceren (natekenen, doortekenen). Wordt ook wel gebruikt als basis of inspiratiebron voor karakteranimatie.	Walt Disney
Live-action/animation	Een animatie wordt bovenop een echte film gelegd.	Who Framed Roger Rabbit?

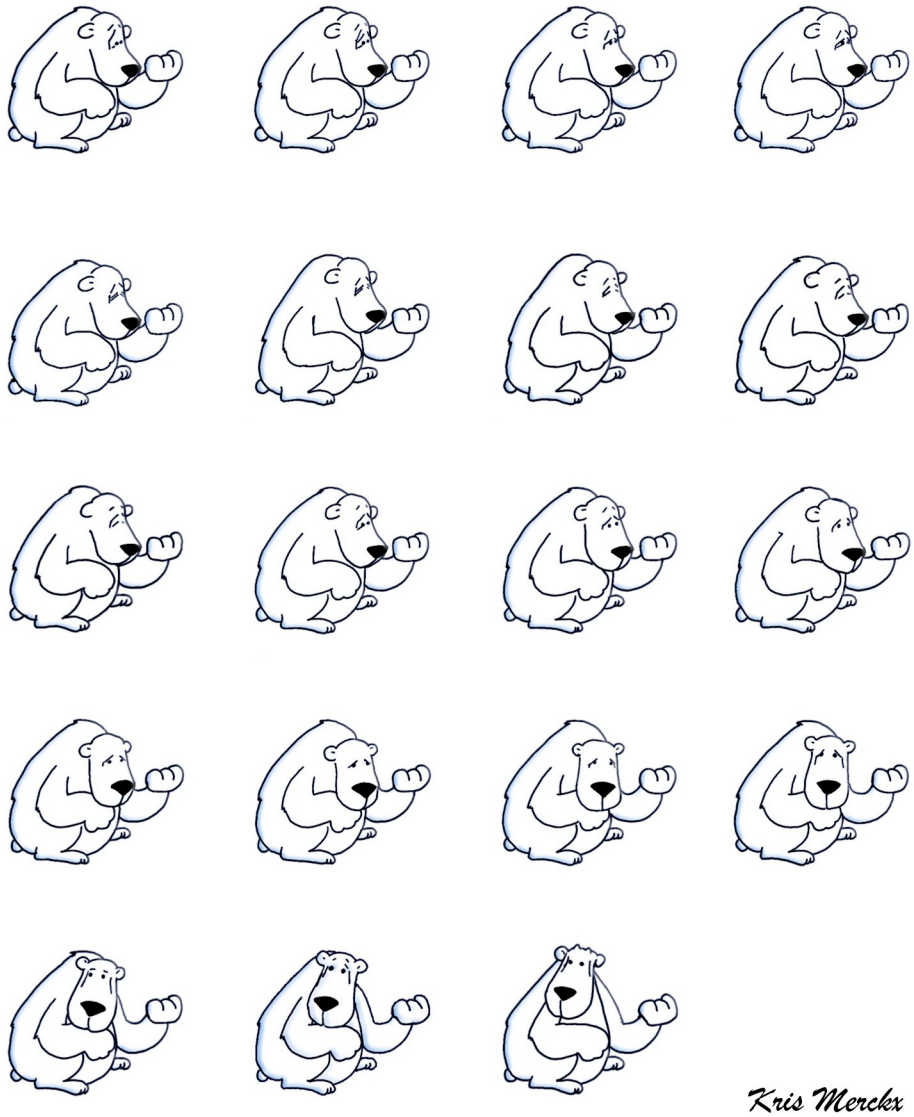
6.5.2 Vormen van stop motion

Bij stop motion wordt elk beeldframe afzonderlijk gefotografeerd. Zorg dat je in dit geval het fototoestel of de camera op een statief plaatst.

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 talen 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 gebruikmaken van ijzerdraad en plasticine.
Cutout animatie	Stop motion-animatie waarbij je gebruikmaakt van uitgeknipte figuren (van papier, stof...). Een tekening of foto van een persoon verknipt je in al zijn onderdelen: romp, hoofd, bovenarmen, onderarmen, handen, bovenbenen, onderbenen, voeten... Met draadjes of klemmen maakt je de onderdelen weer aan elkaar vast. Leg ze op een plat oppervlak met een egale kleur.
Graphic animation	Een vorm van stop motion waarbij onderdelen van foto's, teksten enz. frame per frame worden geanimeerd. Dit kan ook snel met software als Adobe Flash of Adobe AfterEffects gebeuren.
Objectanimatie	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.

6.5.3 Tekenfilm

Net zoals een echte film bestaat een tekenfilm uit een reeks beeldjes per seconde. Bij een tekenfilm worden de beelden tekening per tekening met de hand gemaakt. Anders dan bij een film worden de figuren niet samen in de achtergrond getekend. Het uiteindelijke beeld wordt in al zijn onderdelen opgesplitst en pas achteraf laag voor laag weer samengesteld. Dit betekent dat één seconde tekenfilm vaak uit veel meer dan 25 afzonderlijke



tekeningen bestaat. Voor een echte tekenfilm moet je natuurlijk een handje kunnen tekenen. Anders dan wat velen denken, gebeurt het meeste werk bij een (klassieke) tekenfilm nog altijd met de hand. Het computerwerk beperkt zich tot het inkleuren en monteren van de beelden.

Aan de hand van een storyboard bouwt een tekenfilmmaker 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.

De diverse tekeningen worden gescand en in een animatieprogramma in lagen geplaatst en gemonteerd. Van groot beland 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.

6.5.4 Animatie- en compositingsoftware

Professionele animatoren gebruiken dure animatiesoftware en compositingsoftware. De grenzen tussen de diverse programma's vervagen doordat ze vaak dezelfde of gelijkaardige functionaliteit bieden. De laatste jaren groeit ook het aantal open source pakketten voor animatie.

Software	*Functie*	*Specificaties*	*OS*
Adobe After Effects	compositing, filmanimatie	After Effects kan je zien als Photoshop voor film. Het programma kan Photoshopbestanden inlezen,	Windows, Mac OS X

		legen en laageffecten op een tijdlijn animeren via keyframes. Bijzonder is dat je verschillende onderdelen in een ouder-kind-relatie kan animeren (parenting). Wanneer je het 'ouder'element beweegt, verschuift ook het kindelement.	
Apple Motion	animatie, compositing	Wordt sinds 2006 niet meer als los pakket verkocht, maar als onderdeel van Final Cut Studio. Het programma gaat de concurrentie aan met Adobe After Effects en Discreet Combustion.	Mac OS X
Apple Shake	compositing, postproductie	Een programma voor het nabewerken van filmmateriaal, maar ook met ruimte animatiemogelijkheden. Werd o.a. ingezet voor The Lord of the Rings, King Kong, Harry Potter, War of the Worlds, Star Wars Episode III: Revenge of the Sith, Fantastic 4, Mission Impossible 3, Poseidon, The Incredibles, Hulk, Pirates of the Caribbean 2.	Mac OS X, Linux. De ondersteuning voor IRIX en Windows werden later opgegeven. Momenteel wordt Shake niet meer ontwikkeld. je kan nog wel een licentie op de broncode kopen... voor amper 50000 dollar (terwijl de laatste Mac-versie amper 499 dollar kostte i.p.v. de vroegere 2999).
Cinepaint	beeldbewerking, film	Het open source Cinepaint is ontwikkeld op basis van GIMP. Het programma werd gebruikt voor o.a. Harry Potter, The Last Samurai, The Lord of the Rings, 2 Fast 2 Furious	alle platformen
Combustion	animatie, compositing	Vroeger van Discreet, thans van Autodesk, de makers van o.a. 3D Studio Max en Autocad. Qua functionaliteit vergelijkbaar met After Effects	Windows, Mac OS X
PlasticAnimationPaper	tekenanimatie	Versie 4 kan je gratis downloaden. Het is echter even zoeken vooraleer je overweg kan met de gebruikersinterface: http://www.plasticanimationpaper.dk/	Windows, Mac OS X, Linux
pro motion	tekenanimatie	Een teken- en animatieprogramma voor Windows ontworpen naar het voorbeeld van Amiga Deluxe Paint	Windows

		(DPaint). Ideaal voor het genereren van minuscule pixelanimaties voor o.a. mobiele telefoons en kleine videoconsoles http://www.cosmigo.com/promotion	
Synfig Animation Studio	vectoranimatie	Open source animatieprogramma met een massa interessante mogelijkheden: www.synfig.org. Windowsgebruikers moeten zowel gtk als gtkmm, synfig en synfigstudio installeren. De downloadlinks staan op de site.	Windows en Linux. De versie voor Mac OS X is momenteel niet meer beschikbaar. je kan ze wel zelf compileren op basis van de gratis ter beschikking staande broncode of ze installeren via FINK.
Take5	stopmotion, lijntest	Take5 is een Belgisch product. Heel wat animatoren zijn nog bekend met zijn voorganger Take2 voor Amiga. Take5 kan gebruikt worden voor stop motion met tekeningen, klei, plasticine, cutout, zandanimatie... Het programma kan ook gebruikt worden voor een lijntest (snelle test van de animatie).	Windows
Toon Boom	storyboard, animatie	Het Canadese Toon Boom ontwikelt software voor webanimatie, film, games... DisneyToon Studios gebruikte het voor het inkleuren van een massa animatiefilms. Op http://en.wikipedia.org/wiki/Toon_Boom vind je een overzicht van alle tekenfilmproducties waarvoor Toon Boom-producten werden gebruikt. De thuisgebruikersversie gaat voor een kleine 300 dollar over de toonbank.	Windows, Mac OS X
TVP animation (TVPaint)	animatie, tekenfilm	Heeft de laatste jaren de interesse gewekt van heel wat professionele animatoren. De firma verkoopt ook het animatiepakket Mirage Studio dat vroeger werd verdeeld door Bauhaus Software	Windows, Mac OS X

Deze lijst van professionele software is verre van volledig. Heel wat grote studio's ontwikkelen hun software eigenhandig en die wordt niet altijd vrijgegeven.

Via www.schoolbox.nl kan je als student of leraar academische versies aankopen van heel wat dure softwarepakketten.

Op internet vind je nog een aantal goedkope en zelfs gratis animatiepakketten, maar de mogelijkheden daarvan zijn niet altijd even uitgebreid: FrameByFrame, SingleFramer, FrameThief, Pixen, Pencil, KoolMoves, Swish, KToon, Anime Studio, MonkeyJam, iStopmotion, Creatoon, Sam Animation...

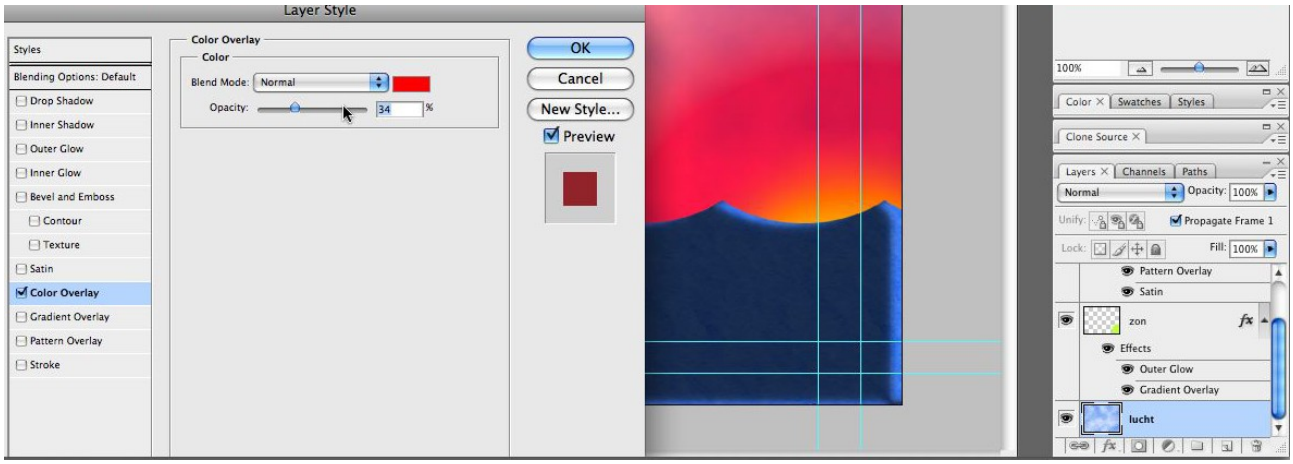
6.5.5 Werken met tijdlijnen, lagen en keyframes

In de meeste animatie- en 3D-software kan je uw informatie opbouwen in verschillende lagen en die lagen ook aanpassen doorheen de tijd op een tijdlijn.

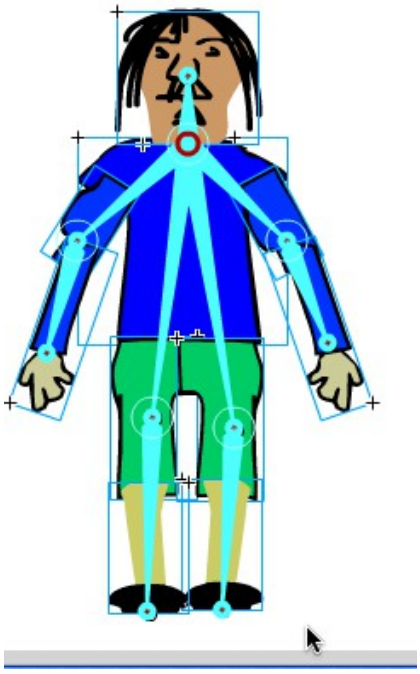
Wanneer je een beeldenreeks importeert, kan je instellen hoeveel frames elke tekening afzonderlijk (voor limited animation) of alle tekeningen (full animation) in beeld blijven. Wanneer je uw tekeningen of lagen met behulp van software tekent, kan je met behulp van keyframes eigenschappen zoals grootte, rotatie enz. van de objecten in de verschillende lagen aanpassen.

Tijdens de lessen bouwen we een gelaagde animatie in Adobe Photoshop.

Lagen en laagstijlen in Adobe Photoshop:



Een virtueel skelet animeren in Adobe Flash met de "bone"-tool.



6.6 Drie of vier dimensies

De term 3D zorgt voor heel wat verwarring. Zeker ook omdat de term in de loop der tijd niet altijd dezelfde lading dekte. Hierdoor bestaan er heel wat misverstanden over dit begrip. Onze ogen ervaren de werkelijkheid in drie dimensies: hoogte, breedte, diepte. Een film toont die drie dimensies wel, maar geeft die op een plat (2D) oppervlak weer. De diepte wordt net zoals in een schilderij gesuggereerd door het perspectief dat natuurlijk automatisch in een foto of film aanwezig is. Daarom kun je zeggen dat films altijd 2D zijn, ook al geven ze de echte werkelijkheid weer. Immers, bij een film zijn de drie dimensies niet 'tastbaar' aanwezig.

6.6.1 Perspectief, peepshows en 3D

Thuis spelen kinderen 3D-games op hun spelconsoles. In de bioscoop kan je kiezen tussen een 3D- of 2D-versie van een film. Voor velen lijkt het nieuw. Maar wat is 3D nu eigenlijk? Perspectief en 3D verschaffen allebei de illusie van diepte. Het perspectief en de verschillende vormen van "3D" vinden hun wortels in de scenografie, de kunst van het opbouwen van scenes en decors in het theater en de schilderkunst. Voor het opwekken van de illusie van diepte, zijn doorheen de tijden diverse technieken ontwikkeld. Sommige waren bedoeld voor een groep mensen, andere konden slechts bekeken worden door één persoon tegelijkertijd. Vormen van perspectief werden reeds in de Klassieke Oudheid toegepast, maar zou vooral vanaf de Renaissance zijn weg vinden naar de schilder- en tekenkunst. Middeleeuwse schilders waren er zich eveneens van bewust dat je afstand kon suggereren door onderdelen kleiner af te beelden.

Schilderijen en theaterscenes waren bedoeld om simultaan door meerdere mensen te laten bekijken. Daarnaast ontwikkelde zich een tweede tak waarbij de illusie voor slechts één persoon bedoeld was. De Europese "peepshows" en de Japanse "rotstuintjes" zijn daar vroege voorbeelden van.

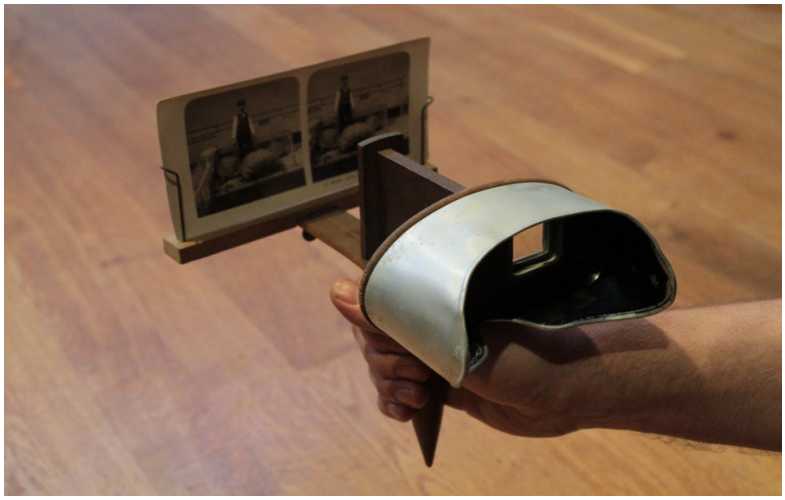
> *"Kijk door de kamer. Verlaat de kamer, sluit de deur en kijk nu door het sleutelgat. Zie hoe dramatisch het uitzicht van de kamer verandert. Manipulatie van de ruimte is zeer oud. Japanse rotstuintjes en Europese peepshow, beide manipuleren ze de ruimte, tonen verrassende, kunstmatige, overeenkomsten, maar ook grondige verschillen." (Balzer)*

Ik vermoed dat de meeste lezers bij het woord "peepshow" denken aan donkere kamertjes van waaruit wellustige mannen naar dansende vrouwen kijken die zich ontkleden. Jammer voor u als dit de reden waarom u verder leest. De peepshow bestaat uit een kleine doos met kijkgat waarbinnen een kleine scene is opgebouwd. Op het einde van de zestiende eeuw dook de peepshow op in de Europese steden als een vorm van straatentertainment. Hij moest concurreren met andere amusementsvormen zoals dansende beren, acrobaten, straatzangers en poppenspelers. Tegen betaling van een kleine bijdrage mocht je door het kijkgaatje piepen. Binnenin de kijkdoos of peepshow werden tegen de achterwand panorama's of tekeningen geplaatst. Die konden worden vervangen of van een bewegend mechanisme voorzien waarbij een lange tekening tussen 2 spoelen werd bewogen. Om de illusie van diepte te suggereren plaatste men soms meerdere uitgeknipte laagjes

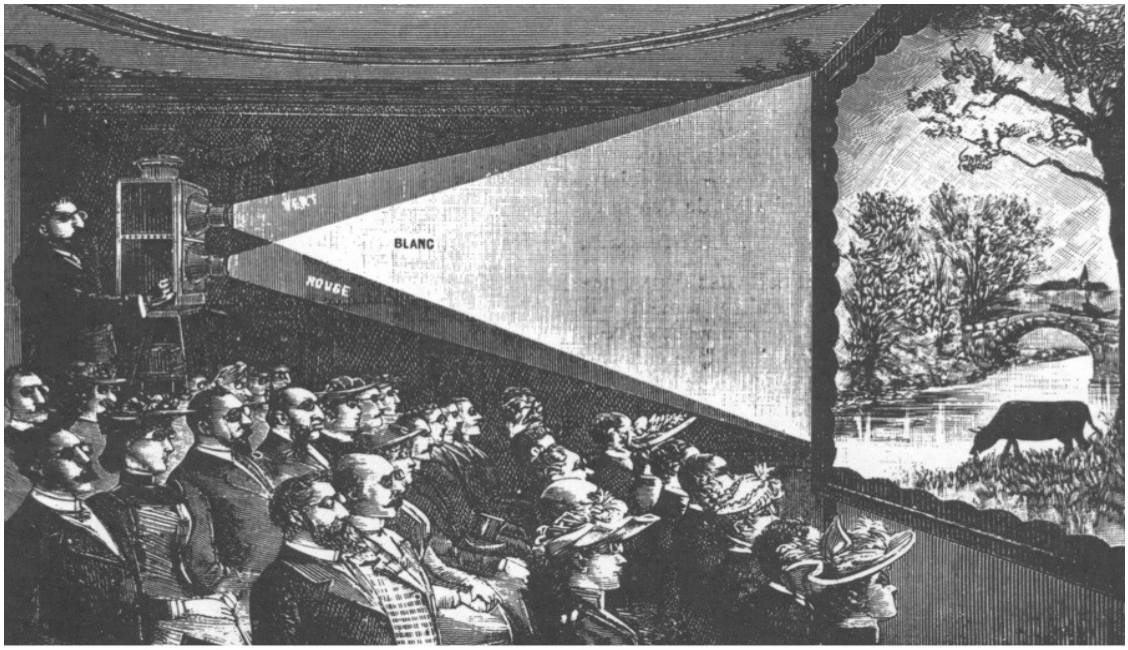
tekeningen achter elkaar. Zo kwamen landschapselementen, personen, dieren en voorwerpen letterlijk los van de achtergrond.

Vergelijkbare technieken waren het "panoptisch polyorama", de "megalethascoop" en de "piepeieren" van albast. Bij het polyorama werden verschillende transparante en/of beschilderde glasplaten achter elkaar geplaatst. Tijdens de voorstelling kon je de ene plaat door de andere vervangen waardoor je bijvoorbeeld een dag- plots in een nachtscene kon veranderen. De **laagjestechniek** duikt later weer op in de tekenfilm en is zoals gezegd ook aanwezig in moderne softwareprogramma's voor beeldbewerking en filmmontage.

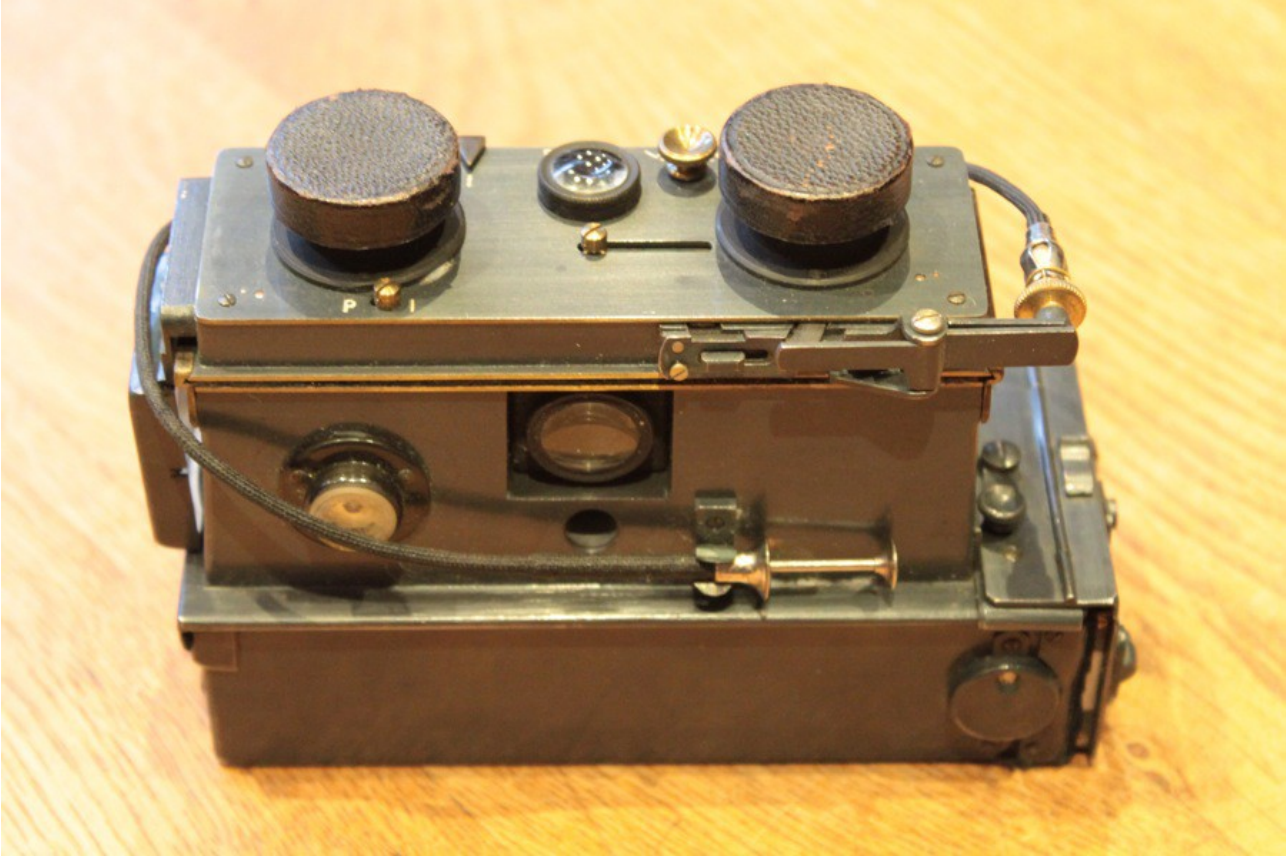
Al snel na het ontstaan van de fotografie ging men op zoek naar een manier om een echt dieptezicht te creëren met behulp van foto's. Euclides beschreef reeds dat elk oog een apart beeld van de werkelijkheid opvangt die worden samengevoegd tot een volledig beeld waardoor we dieptezicht verkrijgen. Claudius Galienus (131 – 201), Leonardo da Vinci en Della Porta hebben zich over dit fenomeen gebogen. De Antwerpse wiskundige, theoloog en architect François d'Aiguillon (1566 – 1617) gebruikte als eerste het woord "**steroscopisch**". Tussen 1832 en 1838 zou de Engelse natuurkundige Charles Wheatstone (1802 – 1875) een apparaat op punt stellen waardoor je twee nagenoeg dezelfde tekeningen vanuit een ander gezichtspunt kon bekijken, de eerste "stereoscoop" en dieptezicht verkreeg. 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. In de negentiende eeuw was het toestel bijzonder populair. Wij kennen het toestel vooral onder de latere merknaam "**Viewmaster**".



Een voorstelling met anaglypische 3D in de tweede helft van de 19e eeuw:



De Verascope, een van de eerste stereoscopische fototoestellen (eind 19e eeuw):



6.6.2 Soorten 3D

De laatste jaren is de opkomst van 3D in de filmwereld niet meer te stuiten. Televisies moeten zorgen voor een echte 3D-ervaring. In de bioscoop kunnen bezoekers kiezen tussen de 2D- of de 3D-versie van een film. Kiezen ze voor de laatste dan moeten ze een speciale bril dragen. De gebruiker/kijker krijgt hierdoor het gevoel in de film te zitten, hij ervaart de beelden niet als vlak, maar ze lijken tastbaar.

Vaak gebruikt men in deze moderne 3D-technieken nog steeds de **stereoscopie** of **anaglypische beelden**. 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.

Sommige brillen bevatten elektronica voor de weergave van 3D. De elektronica in liquid crystal shutter glasses kan synchroon met het scherm licht in één van beide brilglazen blokkeren of doorlaten. Afwisselend wordt er een beeld aan het linkeroog en het rechteroog getoond. Maar dat is slechts één mogelijke techniek.

Met **autostereoscopie** bedoelt men elke techniek waarbij men stereoscopische beelden kan bekijken zonder gebruik te maken van een speciale bril. Sommige methodes controleren voortdurende de stand van het hoofd van de kijker om te bepalen welk beeld er aan elk oog moet worden gepresenteerd.

Bij zo wat alle hoger genoemde technieken krijgt de kijker wel een gevoel van 3D, maar zelden ervaart hij het op dezelfde manier als de echte 'werkelijkheid'. In de echte wereld verandert ons beeld van de omgeving (scene) wanneer we ons hoofd bewegen, op en neer of naar links of rechts kijken: de zogenaamde 'bewegingsparallax'. Een aantal autostereoscopische displays zoals de Nintendo 3DS slagen er zelfs in om het effect van **bewegingsparallax** te simuleren.

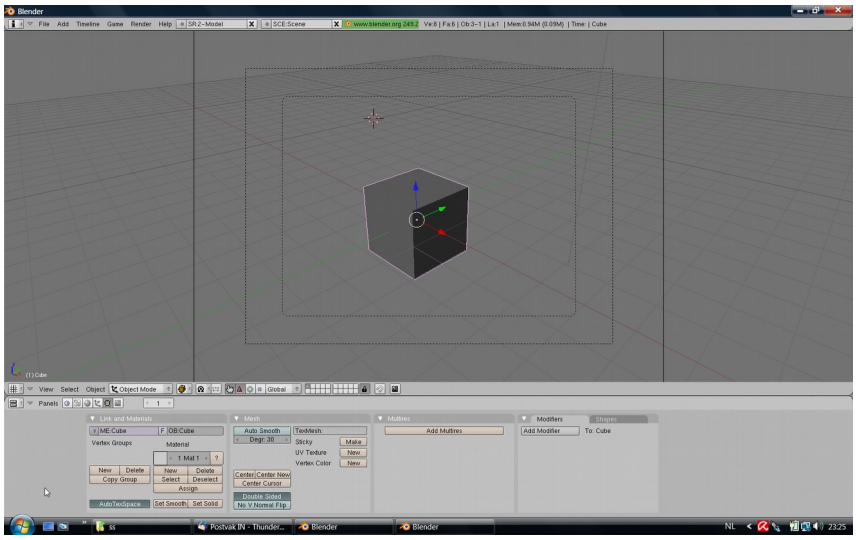
Een **anaglyph** splitst een beeld op in 2 kleuren: een rood beeld voor het linkeroog en een cyaan (combinatie van groen en blauw) beeld voor het rechteroog. Het alom bekende rood-cyaan-brilletje laat enkel rood licht door bij het linkeroog en cyaan bij het rechteroog. Naast het werken met subtractieve kleurenmenging of secundaire additieve kleuren, gebruikt men tegenwoordig ook wel eens additieve kleurmenging met gele (combinatie van rood en groen) en blauwe filters of 'primaire additieve kleuren'. Heel wat mensen schrikken als je hen vertelt dat rood en groen licht samen geel opleveren. Dit komt omdat ze op school enkel subtractieve kleurmenging hebben geleerd. Kortweg komt het er op neer dat je op papier of stof dat enkel licht "weerkaatst" andere kleuren moet mengen om een bepaalde kleur te krijgen, dan bij dingen die licht uitzenden zoals displays, televisieschermen of monitoren (additieve kleurmenging). In 1853 illustreerde W. Rollman als eerste het principe van de anaglyph. Joseph D'Almeida projecteerde in 1858 anaglyphische 3D-beelden met een toverlantaarn door gebruik te maken van rode en groene filters. Net zoals in onze tijd moesten de toeschouwers speciaal geprepareerde brillen dragen. Louis Ducas du Hauron drukte in 1891 voor het eerst anaglyphische afbeeldingen op papier. William Friese-Green maakte in 1889 de eerste 3D-film op basis van deze techniek en hij stelde hem vier jaar laten aan het grote publiek voor. Vanaf de vijftiger jaren van de twintigste eeuw kwam stilaan de term 3D in gebruik. Met succesrijke films als "*The Creature from the Black Lagoon*" (1954) probeerden de filmmaatschappij de belangstelling voor 3D-film van de grond te krijgen.

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

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. De Franse schilder G.A. Bois-Clair demonstreerde deze techniek al in 1692. Wanneer een kijker voorbij zijn schilderijen wandelde, leken ze een voor een in elkaar over te vloeien. In 1896 opperden Jacobson en Berthier de techniek onafhankelijk van elkaar de mogelijkheid om de techniek ook voor foto's te gebruiken. De Amerikaan Frederick E. Ives bracht het in praktijk met zijn "Parallax Stereogram" in 1903 en de Fransman Estenave in 1906.

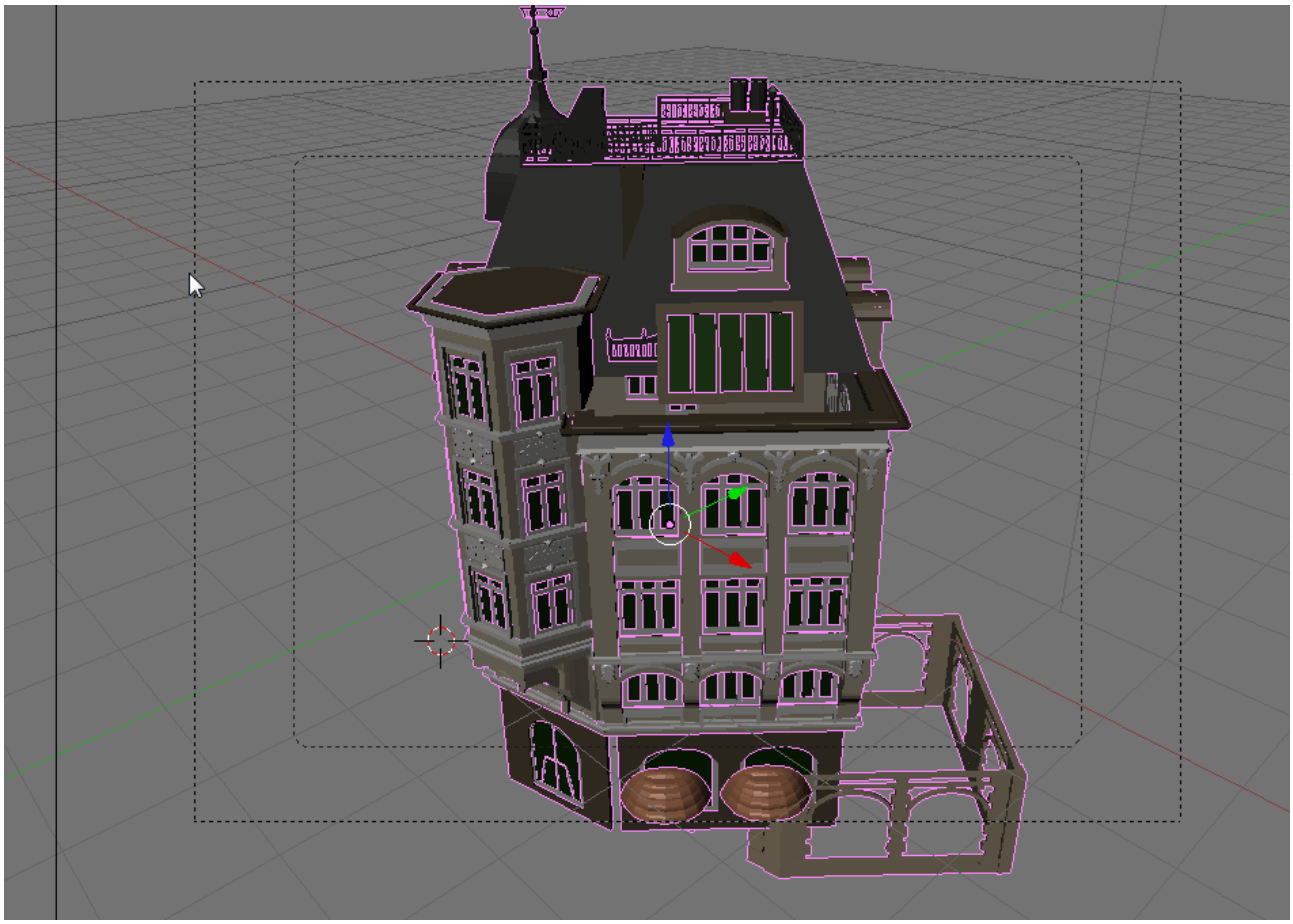
6.6.3 Verwarring over 3D

De term 3D zorgt voor heel wat verwarring. Zeker ook omdat de term in de loop der tijd niet altijd dezelfde lading dekte. Hierdoor bestaan er heel wat ***misverstanden*** *over dit begrip. Onze ogen ervaren de werkelijkheid in drie dimensies: hoogte, breedte, diepte. Een film toont die drie dimensies wel, maar geeft die op een plat (2D) oppervlak weer. De diepte wordt net zoals in een schilderij gesuggereerd door het perspectief dat natuurlijk automatisch in een foto of film aanwezig is. Daarom kun je zeggen dat films altijd 2D zijn, ook al geven ze de echte werkelijkheid weer. Immers, bij een film zijn de drie dimensies niet 'tastbaar' aanwezig.*



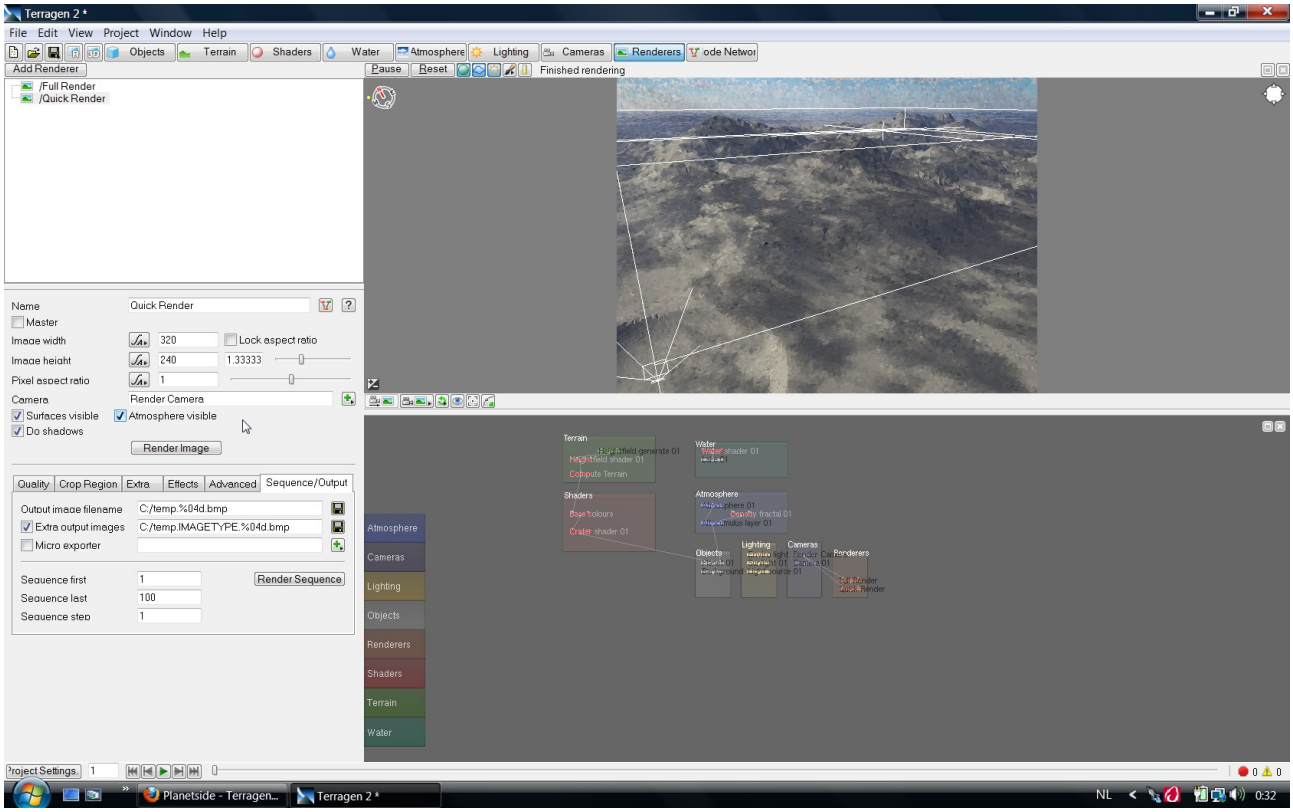
Wanneer we over 3D spreken in software bedoelt men eigenlijk de mogelijkheid voor het bouwen van

virtuele modellen naar analogie met de werkelijkheid. In dit geval tekenen we een 'model' om het nadien te ontwerpen of in productie te brengen (bijvoorbeeld een auto) of gebruikt men 3D-software voor het combineren van filmbeelden met 3D-personages, -gebouwen, -voertuigen. Zo'n 3D-model kun je met de software vanuit verschillende (virtuele) camerastandpunten bekijken, ook al bestaan de modellen in werkelijkheid niet. Hierdoor is het ook relatief makkelijk om ze aan 'echte' film toe te voegen. Zulke **special effects** zijn vaak een stuk goedkoper dan het bouwen van maquettes of modellen voor een filmset. Het eindresultaat van een film die gebruik maakt van 3D-software voor het opzet- ten van een filmscene, is echter nog steeds een 2D-film. **In games wordt 3D interactief.** De speler kan de 3D-modellen aansturen. In dit geval is de 3D-scene interactief omdat de gebruiker zelf bepaalt vanuit welke hoek de scene bekeken wordt. In feite zijn de ogen van de speler zelf de virtuele camera waardoor naar het 3D-beeld wordt gekeken. 3D-animaties maken In tegenstelling tot een klassieke 2D-foto kan een 3D-model geroteerd en vanuit elke hoek bekeken worden. Laat u echter niets wijsmaken: het maken en animeren van 3D-modellen is zeker geen lachertje. De leercurve is bijzonder hoog, zowel op het vlak van modelleren en animeren als wat de gebruikte software betreft. De interface van 3D-software komt vaak zeer vreemd over voor newbies en 3D-groentjes.



In 3D-software kunt u werken met een aantal primitives (kubussen, cilinders), die onderdelen samenvoegen en bewerken tot u het gewenste model verkrijgt. In een tweede fase moet u de modellen ook van een vulling voorzien (kleur, afbeelding). Zo kunt u bijvoorbeeld een foto van een metaalplaat gebruiken als vulling van een 3D-vliegtuig. Net zoals in sommige 2D-animatieprogramma's kunt u 3D-modellen voorzien van een animeerbaar armatuur of skelet. Via **motion capture** kunnen ook de bewegingen van een echte acteur op een 3D-figuur worden overgebracht.

Een virtuele camera in Terragen:



Aan de hand van **particle-systemen** kunnen ook verschillende weersomstandigheden gesimuleerd en geanimeerd worden. Hetzelfde geldt voor natuurlijke effecten, zoals het bewegen van haar en gras. 3D-programma's kunnen een virtuele camera bij een model plaatsen en het beeld van de camera virtueel 'renderen' (=ontwikkelen) naar een gewone afbeelding. Net zoals in 2D-animatiesoftware beschikt 3D-software ook over een tijdlijn waarin u de positie van de objecten of het camerastandpunt in de tijd kunt animeren. Ook schaduw en belichting kun- nen perfect gesimuleerd worden. Voor elk object kunt u de fysische omstandigheden van de werkelijkheid nabootsen door bijvoorbeeld de hardheid van een object of materiaal in te stellen. Zo mag een 3D-figuur uiteraard niet door een muur kunnen wandelen. Dat zou al te belachelijk zijn. Het renderen van de filmbeelden (het omzetten van de 3D-scène en animatie naar een filmbestand of beeldenreeks) vraagt enorm veel rekenkracht van een computer. Daarom wordt voor veel 3D-films een **renderfarm** (een netwerk van computers) gebruikt.

Een virtueel 3D-landschap gemaakt met Terragen.

6.6.4 3D in software

Leren werken met 3D valt buiten het bestek van deze cursus. Het vraagt heel wat techniek, vaardigheid en kennis en een grondig inzicht in de functionaliteit van de gebruikte software. Toch willen we je een klein beetje op weg zetten.

Het onderstaande **overzicht van 3D-software** is verre van volledig. De markt voor professionele 3D-software die gericht is op animatie en film, is de laatste jaren wat kleiner geworden door een aantal overnames. Heel wat kleinere 3D-pakketten zijn opgehouden te bestaan.

3D Studio Max	Discreet/Autodesk
Blender	Blender Foundation
Cinema4D	Maxon
Houdini	Sidefx
Maya	Autodesk, vroeger Alias Wavefront
Modo	Luxology
LightWave	Newtek
Sketchup	Google
SoftImage	Autodesk, vroeger Avid
...	...

Rijzende ster aan het 3D-firmament is, sinds enige jaren een open source product. Het programma is gratis en beschikbaar voor Mac **Blender** OS X, Linux, Windows, Unix e.a. Om de mogelijkheden van het programma te onderstrepen heeft de Blender Foundation samen met een aantal grafici en animatoren een paar 3D-animatiefilms gemaakt die je gratis kan downloaden via www.blender.org. je vindt er ook alle bronbestanden en werkdocumenten die voor de films werden gebruikt.

Op http://wiki.blender.org/index.php/Doc:Manual vind je uitgebreide handleidingen om je op weg te helpen met het programma.

Op artist-3D.com vind je heel wat 3D-modellen die je gratis kan downloaden en in Blender kan importeren.

Blender is (hoofdzakelijk) het werk van de Nederlander Ton Roosendaal. Hij ontwikkelde het programma voor de animatiestudio NeoGeo en Not a Number Technologies (NaN). Na het failliet van NaN werd de broncode van Blender 'vrijgekocht' voor 100000 euro en vrijgegeven onder de GPL-licentie.

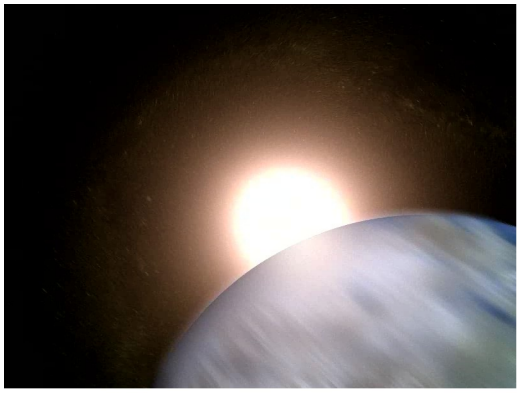
Een relatief volledig overzicht van 3D-software vind je op

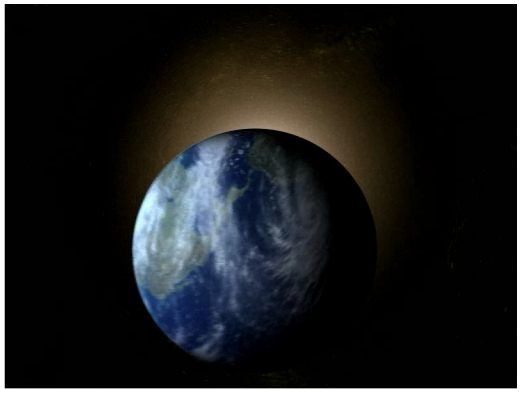
http://en.wikipedia.org/wiki/3D_computer_graphics_software.

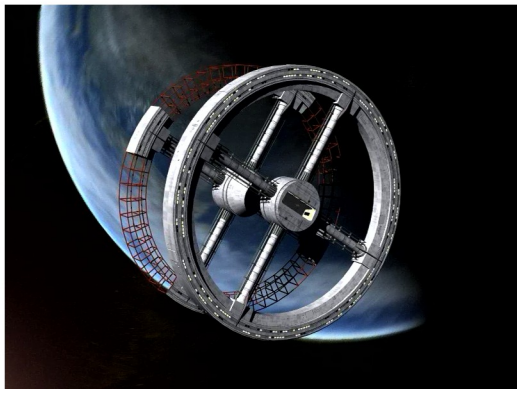
Extensie	*Type*	*Uitleg*
.3ds	3D Studio	Standaardformaat van 3D Studio Max. Algemeen in gebruik als 3D-uitwisselingsformaat.
.3mf	3D MetaFile	Bestandsformaat van Apple's QuickDraw 3D.
.b3d	Bryce 3D	Bestandsformaat van Bryce 3D.

.car	Carrara	Bestandsformaat van Carrara.
.dae	Collada (COLLAborative Design Activity)	Open XML-standaard voor het uitwisselen van 'bestanden' tussen interactieve 3D-applicaties. Het dae-formaat moet een einde maken aan incompatibele formaten.
.dxf	DXF	Eenvoudig uitwisselingsformaat voor 3D-bestanden. Ondersteunt geen texture mapping (vullingen).
.kmz	Google Earth	Het KMZ-formaat is het standaard 3D-formaat van Google Earth. In feite is het niet meer dan een ZIP-bestand met daarin o.a. XML-bestanden.
.obj	Wavefront OBJ	Standaardformaat van Alias Wavefront. Veelvuldig gebruikt voor de uitwisseling van 3D-bestanden tussen diverse programma's. Texture maps kunnen in afzonderlijke .mtl-bestanden opgeslagen worden.
.pz3	Poser	Standaardformaat van Poser 3 en Poser 4.
.rds	Ray Dream Studio	Bestandsformaat van MetaCreations Ray Dream Studio.
.wrl	VRML	Virtual Reality Modelling Language was oorspronkelijk bedoeld als een 3D-formaat voor internet, maar is nooit echt doorgebroken. Voordeel is dat je het kan open in een teksteditor.

3D-animatie gebaseerd op de film "2001: A Space Odyssey":







6.6.5 Afdrukken in 3D

Een interessante en recente ontwikkeling zijn 3D-printers waarmee een computergebruiker op de computer gemaakte 3D-modellen kan "afdrukken". Het toestel print het object in verschillende fijne laagjes kunststof af zodat we een reconstructie in reliëf kunnen maken van om het even welk echt of denkbeeldig object. Het roept vergelijkingen op met de pantograaf en het aanwijsapparaat (pointing device) waarmee je op een makkelijke manier schilderijen (2D) of beeldhouwwerken (3D) verkleind of vergroot kan kopiëren. Volgens sommige onderzoekers (Bulletin Archéologique 30 - 2003) was de pantograaf reeds in de Klassieke Oudheid bekend. De Jezuïet, natuurkundige en astronoom Christoph Scheiner (1573 of 1575 – 1650) uit Ingolstat bouwde de eerste bekende pantograaf in 1603. De pantograaf voor het kopiëren van 3D-objecten werd uitgevonden door James Watt (1736–1819) en op punt gesteld door Benjamin Cheverton (1796–1876) in 1836.

6.7 Audio

6.7.1 Wat is geluid?

Zelf ben ik opgegroeid met geluidsdragers als de langspeelplaat of grammofoonplaat, de muziekcassette en sinds halverwege tachtiger jaren met de CD-speler. Deze cirkelvormige geluidsdragers hebben stilaan plaats geruimd voor digitale opslagformaten. Op weg naar school fietsen of lopen talloze jongeren, door een koptelefoon van de buitenwereld afgesneden, luisterend naar de muziek van hun iPod-mediaspeler of smartphone. De mogelijkheid om duizenden liedjes in nagenoeg "perfecte" geluidskwaliteit op zo'n klein toestel te persen, tart elke verbeelding en werd een paar generaties geleden afgedaan als verre toekomstmuziek. Het wondermiddel dat dit mogelijk maakt heet geluidscompressie waarbij de opgeslagen informatie aanzienlijk kan verkleind worden zonder 'merkbaar' kwaliteitsverlies. Een smartphone kan niet alleen muziek of geluid bewaren, maar ook geluid ontvangen en weergeven. Diverse technieken zitten in één toestel ingesloten: opnemen, coderen, bewaren, doorsturen en ontvangen van geluidsgolven.

Geluid, en dus ook muziek, is in wezen niet meer dan het trillen van lucht (of vloeistoffen of andere materialen) door drukveranderingen. We kunnen geluid voorstellen als een golf met pieken en dalen. De hoogte van een golf of de amplitude bepaalt het volume. Wanneer een pianist een toets heel hard aanslaat, produceert de piano vervolgens een geluidsgolf met een hoge amplitude. Muziek bestaat uit het achter elkaar plaatsen van verschillende tonen in een patroon. Hoge tonen herkennen we in een geluidsgolf door een hogere frequentie d.w.z. meer golfjes binnen een bepaalde tijd. Hoe lager de frequentie of het aantal golfjes, hoe lager de toon. In de natuurkunde wordt de frequentie weergegeven in het aantal trillingen of oscillaties per seconde, uitgedrukt in Hertz of Hz. De laagste toon op een piano laat de lucht ongeveer 27,5 keer per seconde trillen.

Wanneer een geluidsgolf ons oor bereikt, gaat het trommelvlies trillen. De frequentie bepaalt hoe snel het membraan of trommelvlies vibreert. De amplitude laat het vlies sterker of minder sterk vibreren. De vibratie van het trommelvlies brengt drie botjes in het middenoor in beweging (hamer, aambeeld en stijgbeugel). De stijgbeugel brengt de trilling van het trommelvlies op zijn beurt over op de cochlea of het slakkenhuis dat gevuld is met een soort vloeistof. De trilling van die vloeistof wordt overgebracht op minuscule haarcellen die door het spanningsverschil elektrochemische signalen via de gehoorzenuw doorsturen naar de hersenen. Uiteindelijk zijn het de hersenen die zorgen voor het decoderen van het geluid en de geluidsperceptie.

Het menselijk oor herkent een ongelooflijk spectrum van frequenties van ongeveer 20 Hz tot 20000 Hz. Het is bekend dat dieren vaak een ander frequentiebereik hebben. Sommige dieren produceren bijvoorbeeld geluiden die het menselijk oor niet kan waarnemen wegens te laag of te hoog. Tot groot ongenoegen van mijn hond

merkte ik als kind dat het dier begon te huilen als ik op een mondharmonica of bugel speelde. Zelf had ik het gevoel dat ik samen met mijn hond aan het musiceren was, maar de hond moet dit geheel anders ervaren hebben. Het dier moet dezelfde pijn hebben gevoeld als wij mensen bij een plotse schelle toon waarbij we onwillekeurig onze oren afdekken. Een van mijn oudste herinneringen heeft met oorpijn te maken. Het geluid van de trommels van de voorbijtrekkende fanfare raakte mijn prille trommelvliezen als mokerende hamerslagen terwijl ik zo'n geluid nu als eerder doffe klanken ervaar. Dit is niet ongewoon: bij het ouder worden verliezen veel mensen het vermogen om hoge frequenties waar te nemen.

6.7.2 Geluidsopname, -conservatie en -weergave

Geluidsopname en weergave is al evenmin evident. Het veronderstelt drie essentiële onderdelen:

1. de mogelijkheid om geluidstrillingen om te zetten in een andere energievorm (elektriciteit)

2. het bewaren van die informatie (vasthouden voor lange tijd)

3. het opnieuw kunnen laten "horen" van die informatie.

Een telefoon is het eerste toestel dat stap één en stap drie combineerde. Op 26 oktober 1861 demonstreerde de Duitse leraar Philipp Reis in Frankfurt een "telefoon". Reis sprak in een houten hoorn die binnen in het toestel uitmondde op een gespannen stuk darmvlies van een varken waarop een klein stukje platina was geplakt. Wanneer je voor het vlies geluid produceerde, tikte het platina tegen een puntig stukje platina aan. De luidspreker bestond uit een sigarenkistje waarop 2 houten hulpstukken waren gelijmd. Tussen die stukken hout had hij een breinaald bevestigd waar omheen hij een spoel van geïsoleerde koperdraad had gewikkeld. De spoel en de breinaald mochten elkaar niet raken. Hij verbond de microfoon en de luidspreker met 100 meter lange koperdraad zodat een stroomkring werd gevormd wanneer je er een batterij tussen plaatste. Wanneer hij voor de microfoon viool speelde, trilde het vlies waardoor de twee stukjes platina tegen elkaar tikten en korte stroomstootjes via de kring werden doorgegeven. Aan het andere uiteinde ontstond een magnetisch veld in de spoel waardoor de breinaald aan het trillen ging. Het houten kistje versterkte de trillingen en gaf het geluid van de viool weer. Een menselijke stem was eveneens hoorbaar, maar niet verstaanbaar omdat het toestel wel de frequentie doorgaf, maar niet het volume. De eerste bruikbare telefoon kwam van Alexander Graham Bell (1847 – 1922) die de juiste techniek eerder per toeval ontdekte.

Thomas Alva Edison slaagde er als eerste in om een techniek de ontwerpen (hij was de eerste om een patent aan te vragen, want anderen waren met gelijkaardige technieken bezig) waarbij geluid kon worden "bewaard" voor latere weergave. Het principe was weerom even simpel als geniaal. Een naald die aan een trillend membraan was bevestigd, kraste groeven in een met tinfolie bedekte wassen cilinder die op een metalen cilinder draaide. De opnamekwaliteit was aanzienlijk verbeterd door zijn koolweerstandmicrofoon. Het geluid kon nadien opnieuw worden weergegeven door een naald tegen de roterende cilinder te plaatsen. De trillingen van de naald werden versterkt door een hoorn. De toestellen beschikten over een mechanische motor die met een zwengel werd aangedreven.

6.7.3 Geluid digitaliseren en bewerken

Als je geluid kan omzetten in elektrische signalen, wordt het vrij eenvoudig om die elektrische signalen via PWM (pulse width modulation) te digitaliseren. Eens gedigitaliseerd kan je geluid dan ook vrij eenvoudig bewerken en manipuleren. Met het open source en gratis programma Audacity (audacity.sourceforge.net) maakt u opnames in één of meerdere sporen, en kunt u audiobestanden en opnames ook bewerken, samenvoegen en converteren.

Vaak zit het geluid niet helemaal zoals u het gewenst had. Via het Effecten-palet kunt u op verschillende wijzen het geluid, de toon, het volume,... aanpassen. U moet echter vaak meerdere effecten toepassen om het gewenste resultaat te bereiken. Vooraleer u een effect kunt gebruiken, selecteert u een fragment, een audiospoor of de volledige opname.

Versterken	Het geluid versterken. Audacity weet zelf hoeveel het geluid kan versterkt worden. Soms zijn er bepaalde hoge pieken, terwijl de rest van de opname nog te stil blijft. In dit geval kunt u de optie Oversturing toestaan aanvinken om enkel die fragmenten te versterken die te stil zijn. Let op: deze laatste optie is niet altijd aan te raden bij muziekopnames.
Compressor	De compressor-plugin doet net het omgekeerde. Wanneer bepaalde fragmenten te luid of overstuurd zijn, kunnen ze met de Compressor verlaagd worden.
Normaliseren	Het geluidsniveau normaliseren tot op een bepaald niveau.
Toonhoogte aanpassen	De toon van de ene 'noot' naar de andere 'verhogen' of 'verlagen' (=transponeren). Bijvoorbeeld een lied in A (la) verhogen naar Bb (si b). Als u dit effect gebruikt om bijvoorbeeld een mannenstem in een vrouwenstem te veranderen, kunt u best ook nog de toonregeling en de snelheid (tempo) wijzigen.
Toonregeling	Via toonregeling kunt u de tonen (bastonen, midtonen, hoge tonen) en hun diverse niveaus aanpassen. Wanneer u de optie Schuifjes aanvinkt, krijgt u, zoals bij een equalizer, een aantal schuifregelaars om de diverse tonen aan te passen. U kunt hier kiezen uit een reeks presets of profielen om uw opname bijvoorbeeld als een oude Decca 78-toerenplaat te laten klinken. Het is ook mogelijk om eigen profielen op te slaan. Met Mac OS X kunt u in Audacity ook gebruikmaken van de huiseigen (grafische) Apple-equalizer. Tip: Open de equalizer van een mediaspeler en kies een profiel naar keuze. Bekijk de instellingen van de diverse tonen en pas ze op gelijkaardige wijze aan in Audacity.
Omkeren	Hiermee speelt u een bepaald fragment achterstevoren af. Wie weet hoort u wel verborgen boodschappen...

Andere mogelijkheden: bassen versterken, echo, phaser, wahwah,...

Bibliografie

Alle teksten tenzij anders aangegeven zijn geschreven door Kris Merckx

Heel wat hoofdstukken bestaan uit materiaal dat reeds eerder verscheen. Voor een volledige lijst van mijn boeken en bibliografische referenties, verwijs ik u door naar

www.computerkit.be/icaros/apps/werken/boeken.asp

Inhoudstafel

Hello<DIGITAL/>World.....1

Deel 0:.....2
0.1 Het IPOS-model.....3
0.2 Hoe werkt *onze* “computer”?.....4

Deel 1. Digitale en elektronische verwerkingssystemen.....5
1.1 Je weg vinden in het elektronische en digitale landschap.....5
1.1.1 Hardware en software.....6
1.1.2 Embedded systemen en firmware.....7
1.1.3 Voordelen en nadelen.....7
1.1.4 Embedded systeem.....8
1.1.5 SoC's en microcontrollers.....8
1.2 Digitale “dingen”.....9
1.3 Analoog naar digitaal.....9
1.3.1 Sensoren.....9
1.3.2 Actuatoren.....9
1.3.3 Discretisatie.....10
1.3.4 Afspraken.....10
1.3.5 Binair coderen, ruis en foutonderdrukking.....11
1.3.6 Van digitaal naar analoog.....12
1.3.7 Programmeerbare automaten zijn zeer oud (randinformatie).....13
1.4 Microcontrollers en SoC's voor doe-het-zelvers.....16
1.4.1 Arduino.....16
1.4.2 Wiring.....16
1.4.3 Raspberry Pi.....17
1.4.4 Intel Galileo.....17
1.4.5 Little bits.....18
1.5 Oefeningen.....18

Deel 2: Interfaces.....19
2.1 Digitale systemen in de omgang: interfaces.....20
2.1.1 Interactiviteit.....20
2.1.2 Soorten interactieve interfaces.....20
Command Line Interface (CLI).....20
Text User Interface (TUI).....21
Graphical User Interface (GUI) en WIMP.....22
Natural User Interface (NUI): Speech, Touch and Gestures (STAG).....23
Tangible user interface (TUI).....24

Zooming/zoomable user interface (ZUI)....24
Augmented Reality....25
Vormen van AR en de fysieke wereld als interface....26
Virtual Reality....28
Brain Computer Interfaces....28
Leven we in een computer? (The Matrix)....29
2.1.3 User Interface design en UX-design....30
2.1.4 Interactie-ontwerp....30
1. Modulariteit....30
2. Schaalbaarheid....31
3. Toegankelijkheid en gebruiksvriendelijkheid....31
4. Stijl en opmaak....33
5. Leesbaarheid en bestendiging....36
6. Fouten beperken....39
7. Functionaliteit....40
8. Herkenbaarheid....41
9. Storytelling....43
2.2 Goochelen met code....44
2.2.1 Binaire code en programmeertalen....44
2.2.2 Omzetten van programmeercode naar binaire code....46
2.2.3 Functies en methodes....46
2.2.4 Praktijkvoorbeeld 1: Processing....48
2.2.5 Praktijkvoorbeeld 2: javascript....49
2.2.6 Praktijkvoorbeeld 3: in semi-natuurlijke taal met SIRK....50
2.2.7 Aan condities voldoen....50

Deel 3: Verwerking: de *logica* van een digitaal systeem....52
3.1 Computers en taal....53
3.1.1 Computers begrijpen jou niet....53
3.1.2 NLP: natuurlijke talen begrijpen....53
3.1.3 Turingtest: AI of slechts een stapje richting AI?....55
3.1.3 *Taal en de echte wereld* vertalen voor computers....56
3.1.4 Tekst begrijpen, the NLP-way....57
3.1.5 Robotjournalisme....60
3.1.6 Nog moeilijker: van taal naar taal, van spraak naar tekst, van beeld naar tekst, van tekst naar taal... (enz.)....62
3.2 Digital image processing....63
3.2.1 Digitaliseren van afbeeldingen....63
3.2.2 Pixels in een tabel of matrix....63
3.2.3 Pixels en additieve kleuren weergeven....65
3.2.4 Pixels en resolutie....67
3.2.5 Compressie....67
3.2.6 Pixels bewerken....68
3.2.7 Problematische patroonherkenning....69
3.2.8 Vectorafbeeldingen....70
3.3 Operating systems....71
3.3.1 OS en API....71
3.3.2 Open en gesloten besturingssystemen bij de vleet....73
3.3.3 Basisbesturing: BIOS, UEFI, firmware en bootloader....75
3.4 Software en SaaS....75

3.5 Algoritmes die ons (digitale) leven dirigeren ... 77
3.5.1 Sorteeralgoritmes ... 77
3.5.3 Foutcorrectie ... 77
3.5.4 Dijkstra ... 78
3.5.5 Linkanalyse ... 78
3.5.2 Conversie en compressie ... 79
3.5.3 Foutcorrectie ... 80
3.5.4 Dijkstra ... 80
3.5.5 Linkanalyse ... 80
3.5.6 Suggesties? ... 81
3.5.7 Veiligheid ... 81
3.5.8 Controle en voorspelling ... 81
3.5.9 Dobbelen ... 82
3.6 Mechanische, elektronische en digitale computers ... 83
3.7 DNA- en quantumcomputers ... 84
3.8 Artificiële intelligentie ... 85

Deel 4: Storage: Langetermijngeheugen – data ... 86
4.1 Tekstbestanden of binaire bestanden? ... 87
4.1.1 ASCII ... 87
4.1.2 UNICODE en UTF-8 ... 88
4.1.3 Platte tekst en/of binaire bestanden ... 88
4.2 Bestandssystemen ... 90
4.2.1 Opslagmedium in blokjes ... 90
4.2.2 Bestanden terugvinden ... 91
4.2.3 Soorten bestandssystemen ... 92
4.3 Databanken ... 93
4.3.1 Tabellen met relaties ... 93
4.3.2 CRUD ... 95
4.3.3 Kleine en grote databanken ... 95
4.4 Gestructureerde of semantische data voor informatie-uitwisseling ... 96
4.4.1 HTML en XML ... 96
4.4.2 Met goedkeuring en zonder goedkeuring stelen ... 97
4.5 Big data ... 98
4.6 Gegevens kleiner maken ... 100
4.7 Rotte gegevens en dataverlies ... 101

Deel 5: Netwerken ... 102
5.1 De werking van het internet ... 103
5.1.1 Clients en servers: de klanten en hun obers ... 103
5.1.2 Goede afspraken maken goede vrienden: protocollen ... 103
5.1.3 Packet switching: uitwisselen van informatie tussen computers ... 105
5.2 Van hypertekst tot het web ... 105
5.2.1 Wat is hypertekst en hypermedia? ... 105
5.2.2 Teksten mondiaal verspreiden Paul Otlet, de Belgische uitvinder van het analoge internet <RANDINFO> ... 107
5.2.3 De komst van het digitale internet ... 108
5.2.4 Weven op het web ... 108
5.3 Computers begrijpen jou niet ... 109
5.3.1 Hoe doorzoeken software en zoekmachines teksten? ... 109

5.3.2 Hoe werken zoekmachines?....111
5.3.3 De noodzaak van “semantiek”....112
5.3.4 Het deep web.... 113

Deel 6: Output – Feedback voor de mens....116
6.1 Beeldschermen en kleurweergave.... 117
6.1.1 Naar pixels staren.... 117
6.1.2 RGB voor weergave op beeldschermen.... 117
6.1.3 CMYK voor drukwerk....119
6.2 Tekstverwerking, DTP en drukwerk....120
6.2.1 Van gedrukte tekst tot kopieermachine....120
6.2.2 Tekstverwerkers: iedereen is drukker.... 122
6.2.3 Professioneel werken met tekst.... 123
6.2.4 Drukwerk: offset en digitale print.... 123
6.2.5 POD.... 124
6.3 Fotografie en beeldbewerking.... 125
6.3.1 Camera obscura.... 125
6.3.2 De uitvinding van de fotografie: licht vasthouden.... 126
6.3.3 Foto's reproduceren.... 127
6.3.4 Digitale beeldbewerking....128
Compositing.... 128
Painting....128
CGI.... 128
6.4 Film....129
6.4.1 (Zeer korte) biografie van de film.... 130
6.4.2 Hoe werkt timecode en fps?.... 132
6.4.3 Frames per seconde: 25, 30 of 29,97?.... 133
6.4.4 Fields, frames en progressive scan.... 133
6.4.5 Meer frames zijn niet altijd beter....135
6.4.6 Flikkerend beeld of flikkervrij?....135
6.4.7 Containers en codecs.... 136
6.4.8 Soorten montagesoftware.... 138
6.5 Animatie.... 147
6.5.1 Animatieprocédés.... 147
6.5.2 Vormen van stop motion....149
6.5.3 Tekenfilm....150
6.5.4 Animatie- en compositingsoftware....151
6.5.5 Werken met tijdlijnen, lagen en keyframes.... 154
6.6 Drie of vier dimensies.... 155
6.6.1 Perspectief, peepshows en 3D.... 155
6.6.2 Soorten 3D....157
6.6.3 Verwarring over 3D.... 158
6.6.4 3D in software.... 161
6.6.5 Afdrukken in 3D.... 163
6.7 Audio.... 163
6.7.1 Wat is geluid?.... 163
6.7.2 Geluidsopname, -conservatie en -weergave.... 164
6.7.3 Geluid digitaliseren en bewerken.... 165