

HEMELMECHANICA

DE HEMEL IN JE BROEKZAK

Een profielwerkstuk van Jan-Willem Buurlage en Thijs Scheepers.

Planeten: De dwalende sterren	3	Tekenen	17
Over dit werkstuk	3	De combinatie telt	18
Geschiedenis van de hemelmechanica	3	Toegepaste Wiskunde	19
Vragen	5	De wiskunde achter het aanklikken van een object in de 3D ruimte.	19
StarDust Gepresenteerd	6	Een versimpelde wiskundige uitwerking, 2D voorstelling.	19
De Afweging	10	Een uitgebreide wiskundige uitwerking, 3D voorstelling.	19
Keuzeverantwoording voor het besturingssysteem en het platform	10	De wiskunde achter het dubbel klikken en inzoomen op een object in de 3 dimensionale ruimte.	20
Het ontwikkelen voor iPhone is gratis	10	Een versimpelde wiskundige uitwerking, 2D voorstelling.	20
Het platform is stabiel	11	De hemelbol	21
Gebruikers zijn gewend aan applicaties	11	De verschillende vlakken	21
Nadelen	11	De wetten van Kepler	22
Het Ontwikkelproces	12	De positie van de planeten	22
Photoshop	12	Sterrenposities	24
Xcode	12	Sterrentijd	24
Instruments	12	Locatie op de aarde	25
Git	13	Getallen voorbeeld	25
De fasen van het ontwikkelen	14	Aanvullende Wiskunde	27
Bedenkfase	14	Matrixvermenigvuldigen	27
Ontwerp fase	14	Bolcoördinaten	27
Opzetfase	14	Poolcoördinaten	28
Implementatiefase	14	Conclusie	30
Verbeterfase	14	Bronnen	31
Afrondings- en gebruiksfase	14	Wikipedia	31
Introductie in Cocoa	15	Internet	31
Gericht op objecten	15	Tijdschriften	31
Methods	15	Boeken	31
Hiërarchische verdeling van objecten	16	Uren Registratie	32
OpenGL	16		
Functies	17		

PLANETEN: DE DWALENDE STERREN

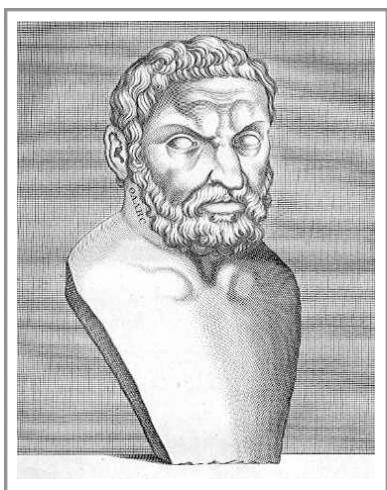
Over dit werkstuk

In dit profielwerkstuk onderzoeken we de wiskunde waarmee de hemel en zijn objecten beschreven worden. In één woord onderzoeken we de hemelmechanica. Daarnaast hebben we veel tijd gestoken in het ontwikkelen van een programma voor de iPhone (een mobiele telefoon) genaamd StarDust waarmee je eenvoudig kunt zien welke planeten, welke sterren, en welke zogenoemde "deep-sky" objecten – waar aan de hemel staan. Bovendien kun je zien wat de fase is van de maan, waar de planeten staan rond de zon en nog veel meer. Met dit programma konden wij onze wiskunde in de echte wereld controleren, en wij hopen dat het andere mensen helpt de hemel te ontdekken.

Zo is dit profielwerkstuk dan ook niet enkel en alleen gericht op de wiskunde van de hemelmechanica. Veel pagina's zullen gewijd zijn aan het programma. We zullen een beschrijving geven van het ontwikkelproces, en vertellen waarom we gekozen hebben voor het iPhone platform. Ook geven we een korte inleiding in de programmeerwereld waarin we vele weken gezeten hebben. Ook zal hier overigens de nodige wiskunde in voor komen. Het werkstuk is opgedeeld in twee kolommen, voornamelijk omdat dit type werkstuk, met veel toelichtende figuren en plaatjes van het programma zich daar erg goed voor leent.

Geschiedenis van de hemelmechanica

Al sinds de oudheid heeft de mensheid zich verwonderd over de hemel. Millennia lang heeft de mens zichzelf vragen gesteld, en op deze vragen antwoord proberen te geven. De Babyloniers en de Egyptenaren hadden, voor die tijd, zeer nauwkeurige apparatuur om waarnemingen te doen, vooral met het oog op kalenders – maar ook om voorspellingen te doen over hemelverschijnselen. De eerste die echter verklaringen gaven waren de Grieken Thales en Anaximander.



Thales

Thales beschreef hoe een zonsverduistering werkte, en schijnt er zelfs een te hebben voorspeld. Van het ene moment op het andere was er, behalve de traditionele verklaring dat de "goden boos waren", een verklaring voor een fenomeen dat de mensheid al sinds het begin van zijn bestaan heeft proberen te begrijpen. Anaximander heeft de grootte van een aantal hemellichamen bepaald, bijvoorbeeld van de maan. Van de sterren dacht hij dat het gaten waren in het firmament waarachter vuren brandde.

In het oude Griekenland had je al filosofen en wetenschappers die een heliocentrisch wereldbeeld hadden, d.w.z. ze waren aanhangers van een theorie waarbij de zon (Grieks: *hélios*) in het midden stond, en de planeten rond de zon draaien. Toch heeft het tot in de 17e eeuw geduurd voordat het heliocentrische model in de westerse wereld het

won van het geocentrische model (geo = aarde). Een van de voornaamste redenen hiervoor is dat de (Rooms-katholieke) kerk dit geocentrische beeld aan bleef hangen, in de bijbel staan immers verschillende passages waarin explicet gezegd wordt dat de aarde het centrum is van het universum. Daarnaast hielden de bekendste Griekse wetenschappers, zoals Ptolemeus en Aristoteles, er een geocentrisch (wereldbeeld er op na).

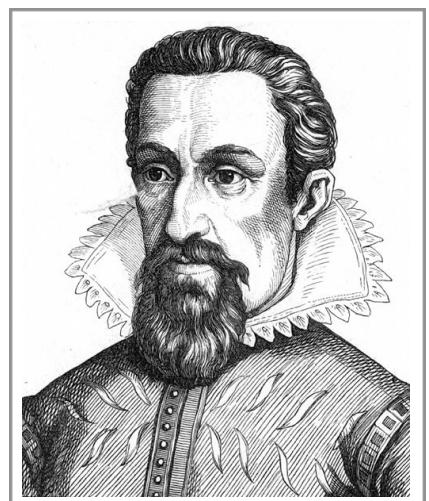
In dit wereldbeeld, van de bekendste Griekse wetenschappers, was er alleen plaats voor "perfecte" vormen, waarvan de cirkel (en de bol) de meest perfecte was. Zowel de planeten als de zon gingen rond de aarde in perfecte vormen. En alle sterren zaten op de rand van één bol waarvan de aarde het middelpunt was.

Pas met de komst van Copernicus begon de heliocentrische theorie weer aan populariteit te winnen. Het was Copernicus die, weliswaar na zijn dood, een ware revolutie op gang bracht. Deze revolutie staat tegenwoordig te boek als de Copernicaanse revolutie. In zijn boek (*Over de omwentelingen der hemellichamen* (*De Revolutionibus Orbium Coelestium*)) beschreef Copernicus een heliocentrisch wereldbeeld, echter zonder empirische bewijzen. Pas toen Galileo bepaalde waarnemingen deed, zoals die van de fasen van Venus en de manen van Jupiter, die alleen verklaard konden worden met een wereldbeeld waarbij de planeten rond de zon draaiden werd de heliocentrische theorie geaccepteerd. Zonder kleerscheuren ging dit echter niet. Lang heeft Galileo door zijn publicaties zich moeten verdedigen tegen de kerk, en heeft zelfs een aantal publicaties weer moeten intrekken.

Kepler was degene die als eerste een wiskundige beschrijving gaf van de theorie van Copernicus. Johannes Kepler was iemand die overal patronen in zocht. Terwijl hij over tabellen keek met data over de bewegingen van Mars, probeerde hij een passende theorie te bedenken. Hij adopteerde de theorie van Copernicus, maar bleef vast houden aan cirkelvormige banen.

Pas nadat waarnemingen van Tycho Brahe, Kepler was de assistent van Brahe, deden inzien dat cirkelvormige banen niet konden kloppen kreeg zijn theorie de vorm die het uiteindelijk zou krijgen: de zon staat in het midden, en planeten draaien in ellipsvormige banen rond de zon, met de zon in een van de brandpunten van deze ellips.

Kepler had geen bewijs voor zijn formules, het kon niet afgeleid worden uit de natuurwetten die in die tijd bestonden. Door middel van de theorie kon eenvoudig voorspellingen worden gedaan over planeetbewegingen, en het had ook een erg eenvoudige verklaring voor een eeuwenoud mysterie: de teruggaande beweging die planeten soms maken. De aarde passeert op dat moment de (hoek)positie van b.v. Mars, waardoor het vanuit het oogpunt van de aarde lijkt alsof Mars terug beweegt. Met de geocentrische theorie had je enorm complexe banen nodig om een verklaring te geven voor dit fenomeen.



De theorie van Kepler bestaat uit 3 wetten. De eerste wet van Kepler zegt dat alle planeten in ellipsvormige banen rond de zon bewegen, waarbij de zon in een van de brandpunten staat. De tweede wet van Kepler zegt dat een denkbeeldige lijn, tussen de planeet en de zon, gelijke oppervlakken aflegt. De derde en laatste wet van Kepler, zegt dat het kwadraat van de periode van een planeet, evenredig is met de derde macht van zijn gemiddelde afstand tot de zon.

Kepler's wiskundige beschrijving kwam haast een eeuw na de theorie van Copernicus. Het zou nog ongeveer 80 jaar duren tot Newton de wetten van Kepler af zou leiden uit zijn net opgestelde bewegingswetten. Newton stelde zijn wetten op aan het einde van de 17e eeuw. Uit zijn vergelijkingen bleek dat planeten inderdaad in ellipsvormige banen rond de zon bewegen. De derde wet van Kepler, bij een denkbeeldige planeet die een cirkelvormige baan heeft, en een massa die te verwaarlozen is bij de massa van de ster, kan je dankzij Newton's gravitatiewet eenvoudig afleiden.

Je stelt hierbij dat de centripetale kracht: $F_c = mv^2$, een formule al eerder opgesteld door Christiaan Huygens, geleverd wordt door de zwaartekracht. (zie rechts)

Sinds de 20ste eeuw hebben we een beter beeld van de zwaartekracht, en waarom hij werkt zoals hij werkt. Vooral dankzij de relativiteitstheorie van Einstein, die o.a. bewegingen van Mercurius verklaart die met Newton's theorie niet te verklaren zijn, hebben we een duidelijker beeld van de zwaartekracht. En dankzij dit beeld kunnen we tegenwoordig met enorme precisie de bewegingen van de planeet over vele millennia berekenen, en ook over millennia terug.

Zo kunnen we door de moderne hemelmechanica hypothesen opstellen over mysteries als bijvoorbeeld de ster van Bethlehem. Waren dit twee planeten die zo dicht bij elkaar stonden dat ze als één ster leken te schijnen? Uit berekeningen blijkt dat in 7 B.C. drie planeten erg dicht bij elkaar stonden. Jupiter, Venus en Mercurius. 7 B.C. is volgens deskundigen ook ongeveer het jaar waarin Jezus geboren werd. De drie planeten zouden echter niet lijken op een ster, waardoor het onwaarschijnlijk is dat de ster van Bethlehem eigenlijk een verzameling planeten was. Het belangrijkste is dan ook dat we d.m.v. de hemelmechanica hypothesen als deze kunnen opstellen, en evt. uitsluiten.

Naast het berekenen van de planeetbanen heeft de wiskunde o.a. ook geleid tot de ontdekking van Neptunus. Toen Uranus ontdekt werd bleken er in zijn baan bepaalde onregelmatigheden te zijn, die niet met Kepler's vergelijkingen te verklaren zijn. Dit kon alleen wanneer er buiten de baan Uranus een extra planeet was. De wiskundigen John Couch Adams en Urbain Le Verrier berekenden onafhankelijk van elkaar de baan van deze planeet. In 1846 werd de planeet voor het eerst waargenomen, op de plek voorspeld door deze twee wiskundigen.

Sinds de komst van moderne telescopen hebben we steeds een beter beeld gekregen van het zonnestelsel. Zo weten we dat tussen de baan van Mars en Jupiter een gordel licht vol planeetachtige objecten (planetoïden). Deze gordel heet dan ook de planetoïdengordel. Ook is in de afgelopen 100 jaar Pluto ontdekt en erkent als planeet, en vervolgens gedegradeerd tot dwergplaneet, omdat de baan van Pluto blijkt te liggen in een gordel met veel objecten die qua grootte en structuur erg op Pluto lijken. Dit is pas sinds 1992 bevestigd met waarnemingen. Deze gordel, die buiten de baan van Neptunus ligt, wordt de Kuipergordel genoemd, naar de Nederlander die hem voor het eerst voorspeld had. Ook is er voorspeld dat er nog een grotere wolk met ijsachtige objecten, dit is nog een Hypothese, opgesteld door eveneens een Nederlander: Jan Hendrik Oort. Deze theorie verklaart waarom we kometen blijven waarnemen, zelfs miljarden jaren na de vorming van het zonnestelsel. Deze wolk heet, naar zijn bewerker, de Oortwolk.

Als mensheid zijn we in een stadium beland waar we onze kosmische buurt aan het ontdekken zijn, zelfs in de twee decennia dat wij op de wereld geleefd hebben is er enorm veel veranderd in het beeld dat de mens had van het zonnestelsel. We sturen ruimtesondes naar planeten, en eveneens naar de zojuist genoemde gordels. We hebben enorm gedetailleerde foto's van de oppervlakten van andere planeten. Buiten ons zonnestelsel zijn er al 500 zogenaamde exoplaneten ontdekt. De geschiedenis van de hemelmechanica even fascinerend als zijn toekomst, en het is erg waarschijnlijk dat we in de komende jaren nog veel meer te weten zullen komen.

Vragen

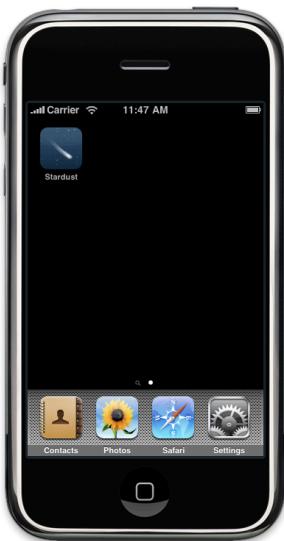
- I. Hoe bouw je een applicatie voor de mobiele telefoon? (Blz. 9 t/m blz 13)
- II. Hoe kan je de sterrenhemel weergeven op het beeldscherm van de telefoon? (Blz.14 t/m 17)
- III. Wat is de wiskunde achter de hemelmechanica? (Blz. 18 t/m blz. 28)**

STARDUST GEPRESENTEERD

In dit hoofdstuk zullen wij beschrijven hoe StarDust werkt. En hoe een gebruiker StarDust goed kan gebruiken voor het sterren kijken in de nacht.

Wij moeten iedereen die dit profielwerkstuk leest er eerst even op wijzen dat op <http://www.moteoflife.net> een mooi instructie filmpje komt te staan. Als je geen iPhone of iPod Touch in je bezit hebt, kan je via dit filmpje de StarDust ervaren.

Om StarDust op te starten zul je op het applicatie icoontje van StarDust moeten klikken. Dit icoontje staat op het beginscherm. Het icoontje is te herkennen aan de vallende ster.



StarDust zal even een paar seconden moeten laden. Gedurende deze laadperiode wordt de 3D omgeving gebouwd en worden de 10.000 sterren, die het programma bevat, ingeladen.

Als het programma uiteindelijk is opgestart zal het scherm een situatie als deze tonen. De horizon word aangegeven door een grijze lijn. Op deze witte lijn staan bolletjes die het kompas voorstellen. Met dit kompas kunt u er voor zorgen dat de telefoon in de juiste richting wijst.

Nieuwere iPhones hebben een digitaal kompas ingebouwd. Als StarDust op een zo'n iPhone draait kan je de omgeving ook automatisch mee laten draaien met de richting waarin je het apparaat houdt. Je kan deze functie natuurlijk ook uitschakelen. En daarnaast kan je zonder kompas natuurlijk gemakkelijk het de hemel kaart bewegen met je vinger.

Je kan滑den, dit is het snel bewegen van de vinger over het scherm en plotseling loslaten. De bol zal door gaan met draaien en vervolgens langzaam vertragen. Je kan inzoomen door dubbel te klikken op het scherm of met tweevingers op het scherm uit elkaar te bewegen.



Je kan met pijltje rechts onderin het menu omhoog halen. Vanuit het menu kan je gaan naar de volgende submenu's:

- Locatie
- Datum / Tijd
- Zonnestelsel weergave
- Instellingen

Daarnaast kan je ook op zoeken drukken. Hiermee kan je sterrenbeelden, sterren, planeten, Messier-objecten (de eerder genoemde deep-sky objecten), de zon en de maan vinden. In dit voorbeeld zoeken wij op:

"Ursa Major". Dit is de engelse benaming voor de Grote Beer.



Als je op "Zoek" drukt zal de grote beer ook gevonden worden. Je kan hier duidelijk het steel pannetje herkennen. En er is te zien dat de grote beer nu nog vlak boven de horizon staat.

Vervolgens zou je bijvoorbeeld een Messier-object, vlak bij de grote beer, aan kunnen klikken.



Een Messier-object (deep-sky object) ligt vaak erg ver van ons vandaan. De meeste Messier-objecten zijn goed te bewonderen met een telescoop en zijn bijzonder mooi. Als je op de i-knoop drukt kan je meer informatie over het Messier-object opvragen. Het informatiescherm ziet er als volgt uit:



Deze informatie bevat onder andere het dichtstbijzijnde sterrenbeeld, het type Messier-object, afstand, zichtbaarheid; originele Right-ascension en declination; en huidige azimuth en altitude. (Azimut en hoogte, coördinaten die verderop in dit hoofdstuk beschreven zullen worden)

Right-ascension en declination (Rechte klimming en declinatie) zijn coördinaten die vast staan. Deze coördinaten zijn bol-coördinaten die op het moment van de standaardepoche¹ vastgelegd waren.

Azimuth en altitude zijn de huidige coördinaten van het object, op jou positie, en de tijd. Deze coördinaten zijn tevens uitgedrukt in bol-coördinaten.

Natuurlijk kan er niet alleen op Messier-objecten geklikt worden. Dit kan ook met sterren. Hier kan informatie over worden opgevraagd. Dat ziet er als volgt uit:



Dit informatie paneel bevat tevens de catalogusnummers van de sterren.

Je kan ook op een planeet klikken en hier informatie over opvragen. Ook hier komt een mooi plaatje tevoorschijn.



1. De standaardepoche is de referentie waarmee de coördinaten van de vaste sterren vastgelegd worden. Voor de nieuwe sterren catalogusen is dit 01-01-2000 00:00:00 UTC.

Je kan de planeten niet alleen tussen de sterren bewonderen. Je kan ons zonnestelsel ook van bovenaf bekijken in onze zonnestelsel functie. Deze ziet er als volgt uit:



In dit zonnestelsel-weergave kan je gemakkelijk door- en terugspoelen, zodat je goed kan zien hoe de planeten over hun elliptische banen wandelen.

Je kan ook in deze weergave weer gewoon met je vingers dubbel klikken en in- en uitzoomen.

Het uitgezoomde veld ziet er als volgt uit:



Naast deze standaard functies zijn er ook nog functies ingebouwd die de enthousiasteling erg goed kan gebruiken tijdens het bewonderen van de hemel.

Als je buiten, met je telescoop, en naar de hemel aan het turen bent zijn je ogen na een tijdje aan de donkere gewend. Het licht van het beeldscherm van de telefoon kan dit natuurlijk helemaal verpesten. Vandaar dat wij de rood-modus hebben ingebouwd. In het instellingen menu kan je deze functie activeren.



Rood licht heeft een lage golflengte en heeft daarom een klein effect op de ogen in verhouding tot de andere kleuren licht. Dit is erg prettig voor de telescoop gebruiker. Ook natuurlijk voor iemand die met het blote oog naar de sterren kijkt.

Daarnaast kan, indien er ver ingezoomd wordt, ook een vizier geactiveerd worden in het instellingen menu. Op deze manier weet u altijd waar u met uw telefoon naar toe wijst.

Dit is handig voor het precies lokaliseren van een specifieke ster. Dit vizier kan namelijk goed gebruikt worden door de telescoop gebruiker die gebruik maakt van ster-hoppen.



Naast al deze functies heeft StarDust ook nog een paar verborgen functies. Namelijk:

- StarDust is meertalig en kan gemakkelijk worden uitgebreid met een nieuwe taal. De huidige talen zijn Nederlands en Engels.
- StarDust heeft een Bayer en Flamsteed interpreteerder. Deze interpreteerder maakt van een reeks karakter een mooi opgemaakte zin met hierin Griekse letters die toegewezen zijn aan bepaalde sterren. Deze interpreteerder extraheert daarnaast ook nog de naam van een sterrenbeeld uit deze reeks karakters.

- StarDust laat de fase van de maan zien. Als je in het programma op zoekt waar de maan staat kun je dus ook zien of het volle of nieuwe maan is.

Dit was een klein overzicht van wat StarDust allemaal kan. Ik moet er ook op wijzen dat StarDust heel veel animaties heeft ingebouwd die niet te zien zijn in een profielwerkstuk op een stukje papier. Dus ik raad toch aan om even de filmpjes te bekijken op de website <http://www.moteoflife.net>

DE AFWEGING

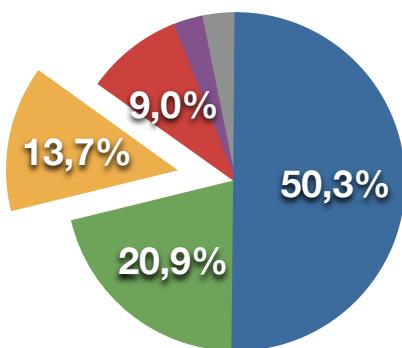
Wij hebben onszelf het doel gesteld een software toepassing te ontwikkelen waarmee iedereen gemakkelijk de nachtelijke hemel kan bewonderen. Als je de naar de sterren kijkt doe je dat niet vanuit je woonkamer maar vanuit de achtertuin of vanuit het park. En op deze plek heeft natuurlijk niemand een computer paraat. Daarom wilde wij graag de toepassing ontwikkelen voor gebruik op de mobiele telefoon. De mobiele telefoon heeft de laatste paar jaar, in rap tempo, technologische vooruitgang geboekt. Deze nieuwe krachtige telefoons worden smartphones genoemd. De gemiddelde smartphone heeft meer rekenkracht onder de motorkap dan een kantoor computer 8 jaar geleden. Het is dus heel goed mogelijk om onze applicatie op een smartphone te laten draaien.

Keuzeverantwoording voor het besturingssysteem en het platform

In de wereld van mobile telefoons zijn er verschillende besturingssystemen¹ die de mobiele telefoons zijn functies geven.

Hieronder staan de meest gebruikte en hun marktaandeel. Natuurlijk heeft elk platform zijn eigen voor- en nadelen. Deze hebben wij in overweging genomen.

- | | |
|-----------|------------------|
| ● Symbian | ● Blackberry |
| ● iPhone | ● Windows Mobile |
| ● Android | ● Overig |



Voor het gemakkelijk ontwikkelen van software kwamen wij er al snel achter dat het iPhone platform van Apple de beste keus was. Om dit even toe te lichten hebben wij een paar voordelen hieronder neer gezet.

- I. Het ontwikkelen vereiste geen licentiekosten. Het ontwikkelen voor iPhone is dus gratis.
- II. Het platform is stabiel vanwege het beperkte aantal ondersteunde apparaten en vanwege de verzekering dat alle systeem voorzieningen altijd beschikbaar zijn.
- III. Daarnaast zijn iPhone gebruikers het snelst geneigd om applicaties van een derde partij op hun telefoon te zetten. En deze vervolgens dan ook vaak te gebruiken.

Op deze voordelen zullen wij nu iets dieper ingaan.

Het ontwikkelen voor iPhone is gratis

Apple stelt aan alle software ontwikkelaars een gratis software pakket beschikbaar waarmee erg makkelijk voor de iPhone kan worden ontwikkeld. Dit in tegenstelling tot bijvoorbeeld Microsoft, wiens softwarepakket honderden euro's kost.

Apple's software pakket bestaat uit een aantal onderdelen. Al deze onderdelen hebben wij uitbundig gebruikt tijdens het ontwikkelen van StarDust.

- Xcode
- Instruments
- Interface Builder
- iPhone Simulator



In het hoofdstuk 'het ontwikkelproces' komt het gebruik van deze programma's verder aan de orde.

1. Een besturingssysteem, of in het engels een operating system (OS), is een stuk software dat de hardware van een bepaald apparaat aanstuurt. Dit stuk software slaat een verbinding tussen de applicaties en de hardware.
2. Een platform is een industrie term voor de combinatie van hardware en het besturingssysteem.

Het platform is stabiel

Het besturingssysteem van de iPhone is speciaal gemaakt voor dat specifieke apparaat. Dit apparaat wordt naast de software ook door Apple zelf ontworpen en gebouwd. Dus Apple heeft zelf volledige controle over onder andere design keuzes in het product. Dit komt stabiliteit ten goede.

Om deze bewering even met perspectief te bekijken hebben bijvoorbeeld telefoons met het Windows Mobile besturingssysteem een slechtere stabiliteit. De apparaten worden namelijk gebouwd door bedrijven als HTC, Sony Ericsson en HP. Maar de software wordt ontwikkeld door Microsoft. Nu zijn er dus al 4 of meer partijen die samen moeten beslissen over de functies van de nieuwe versies van Windows Mobile. Het zal lastiger zijn om, in verband met gebruiksvriendelijkheid, functies weg te laten. Lang niet alle partijen zullen hier altijd mee instemmen.

Het besturingssysteem van de iPhone laat geen achtergrond processen toe. Dus op de achtergrond MSN laten draaien is niet mogelijk. Als de gebruiker een applicatie heeft opgestart kan deze applicatie op alle systeem bronnen rekenen. Bijvoorbeeld kan de applicatie gebruik maken van vrijwel alle geheugen ruimte en alle processor cycli, met uitzondering van het gebruik van het besturingssysteem zelf natuurlijk. Dit is erg prettig voor een ontwikkelaar. Deze weet namelijk dat de gebruiker altijd de ervaring met het programma zal krijgen die hij zelf ook bedoelt heeft.

Gebruikers zijn gewend aan applicaties

Momenteel zijn er voor de telefoons van Apple wel 15.000 applicaties te krijgen. Gebruikers kunnen gemakkelijk applicaties uitzoeken, downloaden en hebben deze vervolgens binnen een paar seconden op hun telefoon geïnstalleerd.



De applicatie winkel, AppStore, is erg simpel te navigeren en levert de gebruiker ook nog een centrale plek voor updates aan applicaties. Stel er zou een nare fout in StarDust zitten waar wij pas na de publicatie achter komen. Dan kunnen wij binnen een week onze StarDust gebruikers informeren over de nieuwe update, en deze kunnen de gebruikers vervolgens ook weer gemakkelijk downloaden.



Het hoofdvenster van de iPhone nodigt een gebruiker uit. Alle functies van het apparaat zijn in verschillende applicaties onderverdeeld. En op het hoofdvenster staan deze weergegeven. Er staan in totaal 20 grote knoppen op het scherm waarop men kan klikken voor het gebruiken van een van deze functies. Het is dus voor een gebruiker heel logisch om voor een nieuwe functie gewoon aan dit hoofdvenster een grote nieuwe knop toe te voegen.

Nadelen

Helaas heeft het ontwikkelen voor de iPhone ook een paar nadelen. Volgens ons waren deze echter niet overheersend.

- I. Alle applicaties die beschikbaar in de applicatie winkel komen te staan worden eerst beoordeeld door Apple.
- II. Er is geen mogelijkheid, anders dan de AppStore, om applicaties te verspreiden onder het grote publiek.

Apple wil graag controle houden over de applicaties in de AppStore en dus op elke iPhone. Apple heeft bijvoorbeeld ook de mogelijkheid een bepaalde applicatie in een keer van alle iPhones, waar deze op staat, te verwijderen. Dit doen zij in verband met het bestrijden van mogelijk kwaadaardige software. Maar dit kan natuurlijk ook als een inbreuk op de privacy van gebruikers worden ervaren.

HET ONTWIKKELPROCES

Het ontwikkelen van een applicatie is nog best een opgave. Er moet veel uitgedacht worden. Naast de code schrijven moet er bijvoorbeeld ook worden gedacht aan het design en hoe de gebruiker gemakkelijk en intuïtief het programma kan leren gebruiken.

Ook is het bijvoorbeeld van belang dat er met z'n tweeën tegelijk aan het programma gewerkt moet kunnen worden. Als de één stukjes code hier verandert en de ander daar, moet dit natuurlijk vervolgens wel allemaal mooi samenvallen.

Om het ontwikkel proces beter te begrijpen lichten wij eerst een paar programma's toe die wij gebruikten voor het ontwikkelen van StarDust.

Photoshop

De applicatie moet natuurlijk niet alleen goed werken. Het oog wil ook graag iets.



Voor het ontwerpen van de interface elementen gebruiken wij Photoshop. Interface elementen zijn onder andere de achtergrond van het menu, de knopjes, het selectie vizier etc.

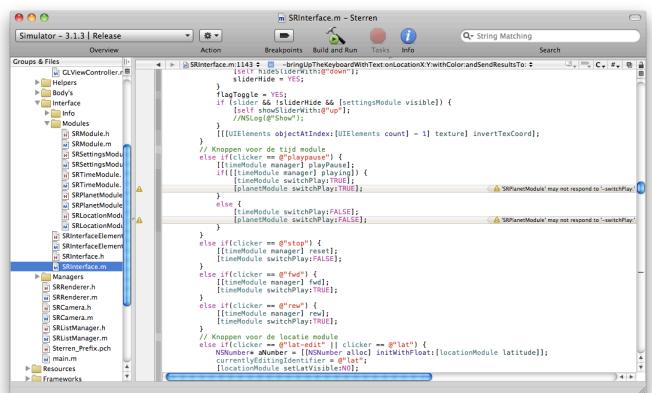
Photoshop is een heel divers programma. Velen kennen Photoshop voor het bewerken van foto's. Naast het bewerken van foto's is photoshop ook volledig uitgerust voor het ontwerpen van websites, advertenties, tijdschriften en dus ook telefoon applicaties.



Xcode

Code moet ook in een programma geschreven worden. Sommige ontwikkelaars doen dit in een super simpel programma zoals Kladblok. Apple levert echter de Xcode IDE mee. IDE staat voor interactive developer environment.

Xcode is een programma waar alle stukken code goed bij elkaar gehouden worden. Er kan simpel op een build-knop gedrukt worden om de huidige code uit te proberen.



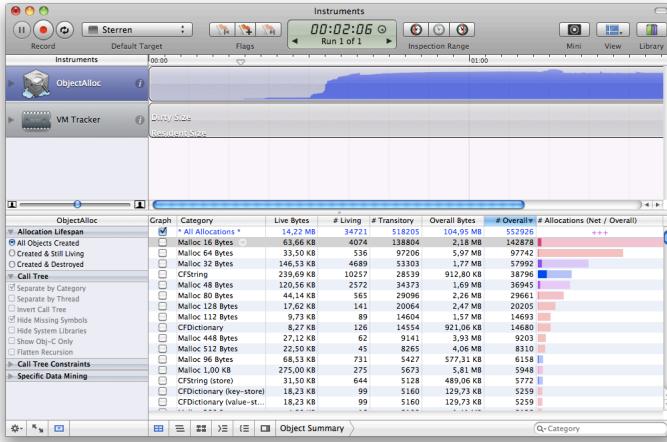
Naast het handig organiseren van code geeft Xcode correcties per regel. En maakt gebruik van code-highlighting, het weergeven van code in verschillende kleuren. Dit houdt code erg overzichtelijk.

Instruments

De software suite van Apple bestaat echter niet alleen uit Xcode. Ook Instruments wordt mee geleverd.



Instruments is een programma waarmee de prestatie van een applicatie gemeten kan worden.



Instruments heeft ons geholpen met het naar boven krijgen van onze framerate. Framerate is het aantal verversingen dat het beeldscherm per seconde kan doen. Als er elke verversing meer gebeurt duurt deze langer en is er dus een lagere framerate.

Wij hadden namelijk een stukje code geschreven dat alle sterren op het scherm projecteerde. Door dit met Instruments nader te analyseren kwamen wij er achter dat dit specifieke stukje code erg inefficiënt was. Dit stukje hebben wij toen herschreven, en als gevolg daarvan werd onze framerate bijna verdubbeld.

Git

Git is een versiebeheersysteem, ook wel broncode-beheersysteem genoemd. Git zorgt er voor dat twee of meerdere mensen gemakkelijk samen aan 1 stukje software kunnen werken.

Als er een verandering hebt aangebracht in de code commit je dit via Git. Committing is het bevestigen van je code. Als echter een de persoon waarmee je samenwerkt een ander stukje code gewijzigd en gecommit heeft, is Git zo slim en voegt deze 2 commits samen. En als er een conflict ontstaat, bijvoorbeeld alle beide programmeurs waren met dezelfde regels code bezig, kan dit makkelijk worden opgelost.

Wij gebruiken Git samen met een internet dienst genaamd GitHub (www.github.com). Op deze manier slaan wij onze code altijd online op, en kunnen wij er overal bij.

Commit History for tscheepers's StarDust-iPhone – GitHub	
jvbuurlage (author)	January 14, 2010
sunInfo, moonInfo toegevoegd	
jvbuurlage (author)	January 14, 2010
glow eng	
jvbuurlage (author)	January 14, 2010
vele verbeteringen ui	
jvbuurlage (author)	January 14, 2010
Kleine bug in slider (rood mode) opgelost	
jtscheepers (author)	January 14, 2010
Loading screen engels	
jtscheepers (author)	January 14, 2010
Zon ondergaan zon opkomen bg weer aangezet maar planetview bug fixed	
jtscheepers (author)	January 14, 2010
Slider wordt rood, en search weer op de juiste plaats	
jtscheepers (author)	January 14, 2010

Hier is een lijst met commits te zien. Elke commit kunnen wij benaderen en individueel weer opvragen. Dus als we ergens een fout in de code hebben geschreven kunnen wij gemakkelijk weer terug naar een oude versie van het programma.

tscheepers's StarDust-iPhone at master – GitHub	
git@github.com:tscheepers/StarDust-iPhone	tree 3c562eac6d1b1518ef766749a85d7575d66d645384
git@github.com:tscheepers/StarDust-iPhone	commit 430811acd3378f71b7e1c9083c93037340b858235
git@github.com:tscheepers/StarDust-iPhone	tree 3c562eac6d1b1518ef766749a85d7575d66d645384
git@github.com:tscheepers/StarDust-iPhone	parent ecb1acfe1887b03d28034cc0df21b6e74d2
git@github.com:tscheepers/StarDust-iPhone	parent u945e26d6f084837c317d52d8bd5c936e4dd2
git@github.com:tscheepers/StarDust-iPhone	commit 86d185d3b60f7a4446c8d8a34cc0df21b6e74d2
git@github.com:tscheepers/StarDust-iPhone	tree 5e6fd1802e9489735d5e151a4437242e929
git@github.com:tscheepers/StarDust-iPhone	parent bd1d5e5d1b91aco/f4028362e7f33f07e38d8f
git@github.com:tscheepers/StarDust-iPhone	commit bd1d5e5d1b91aco/f4028362e7f33f07e38d8f
git@github.com:tscheepers/StarDust-iPhone	tree 09d9ab7ea3f34ed526295d5742397f6243c1c1
git@github.com:tscheepers/StarDust-iPhone	parent 8a8185d8306977a404f73b9e1
git@github.com:tscheepers/StarDust-iPhone	commit 86d185d3b60f7a4446c8d8a34cc0df21b6e74d2
git@github.com:tscheepers/StarDust-iPhone	tree cba441391cc0a4845f777f34d27e2225c16
git@github.com:tscheepers/StarDust-iPhone	parent cf6793f7441792e328a297e2991df59bf9ff9a9

Naast het opslaan van de code via deze internet dient kunnen wij deze ook gewoon vanuit de browser bewerken. Wij hebben dus niet Xcode nodig om code te bewerken. Dit was erg handig voor het werken aan StarDust op school bijvoorbeeld.

Issues – tscheepers/StarDust-iPhone – GitHub	
git@github.com:tscheepers/StarDust-iPhone	Admin Unwatch Pull Request Download Source Graphs Branch: master
Source Commits Network (2) Fork Queue Issues (24) Downloads (3) Wiki (3) Graphs	Branch: master
Branches (2) Tags (0)	
Planetary iPhone application http://www.motefile.net/starburst	
Private git@github.com:tscheepers/StarDust-iPhone.git	git@github.com:tscheepers/StarDust-iPhone.git
Merge branch 'master' of git@github.com:tscheepers/Sternen	commit 430811acd3378f71b7e1c9083c93037340b858235
jvbuurlage (author)	tree 3c562eac6d1b1518ef766749a85d7575d66d645384
February 11, 2010	parent ecb1acfe1887b03d28034cc0df21b6e74d2
parent u945e26d6f084837c317d52d8bd5c936e4dd2	
StarDust-iPhone /	history
name age message	
Classes/ February 11, 2010 nieuwe [jvbuurlage]	
MainWindow.xib January 07, 2010 Zoon bug opgelost [tscheepers]	
Other/ January 18, 2010 Sternen xml herscheld [tscheepers]	
Resources/ February 11, 2010 Merge branch 'master' of git@github.com:tscheep... [jvbuurlage]	

GitHub heeft ons ook geholpen met de eenvoudige documentatie gereedschappen. GitHub is bijvoorbeeld uitgerust met een Issues-list. Hier kunnen wij dus alle mogelijke nieuwe functies en fouten in het programma opzetten en afkruisen als ze afgehandeld zijn. Op deze manier konden wij ook een goed overzicht houden op wie wat zou doen en wie wat gedaan had.

De fasen van het ontwikkelen

Natuurlijk moeten er veel verschillende fase afgewerkt worden om daadwerkelijk een programma in de handen van gebruikers te krijgen. Wij hebben in de volgende fase het project gerealiseerd.

Bedenkfase

1 oktober – 20 oktober

In onze eerste fase hebben wij vooral bedacht hoe het programma er uit moest gaan zien. Wij hebben alle functies van het programma eens goed op een papiertje gezet en zijn tekeningen gaan maken van de verschillende schermen waar de gebruiker in terecht zou kunnen komen.

In deze fase hebben wij ook overlegd of het nuttig is bepaalde functies te implementeren. Wij hebben bijvoorbeeld gedacht aan het implementeren van de huidige locatie van het International Space Station, een permanent bemand ruimte station. Natuurlijk zou deze functie erg aardig zijn. Echter zal de amateur astronoom nooit met zijn telescoop het ISS goed kunnen bekijken omdat deze erg snel door de lucht beweegt. Dus hebben wij besloten het ISS niet weer te geven in ons programma.

Ontwerp fase

21 oktober tot 10 november

In deze fase zijn wij begonnen met het ontwerpen van de interface van het programma in Photoshop. Daarnaast hadden wij een beginnetje gemaakt aan onze 3D omgeving. Wij hebben onder andere als oefening 3 dimensionale kubussen getekend op de iPhone.

Opzetfase

11 november tot 30 november

Wij zijn in deze periode druk geweest met het opzetten van het programma. Wij hadden bijvoorbeeld al een 3D omgeving gebouwd en hier al sterren in geladen. Tevens hadden wij de planeet posities berekend. Dit deden wij allemaal met Xcode

Implementatiefase

1 december tot 31 december

Eigenlijk is in deze periode het helle programma van de grond gekomen. We zijn vanaf deze fase begonnen met het gebruiken van GitHub. Wij hebben tijdens de implementatiefase alle functies van het programma gebouwd. Wij hebben de locatie en tijd rotaties ingebouwd, wij hebben de scherm interactie geprogrammeerd en interface elementen ontworpen.

Aan het eind van de implementatiefase hadden wij eigenlijk een volledig werkend programma.

Verbeterfase

1 januari tot 31 januari

Tijdens deze fase hebben wij het hele programma nog eens binnenstebuiten gekeerd en het een stuk sneller gemaakt. Gedurende januari hebben wij veel gebruik gemaakt van Instruments. Daarnaast hebben wij in januari de hele gebruikers interface nog een keer omgegooid.

Het programma is tot wel 3 keer sneller geworden in het opstarten en het herberekenen van de sterposities.

Afrondings- en gebruiksfase

1 februari tot heden

Wij hebben in deze fase nog een paar fouten in het programma opgelost en zijn bezig geweest met het programma in de AppStore krijgen. Daarnaast hebben wij een website voor het programma gebouwd zodat mensen het programma gemakkelijk kunnen vinden via het internet. Wij wachten nog steeds met smart tot dat onze applicatie in de AppStore ligt en door iedereen gedownload kan worden.

INTRODUCTIE IN COCOA

cocoa |'kōkō|

zelfstandig naamwoord

een object-georiënteerde API, geschreven in Objective-C.

Cocoa Touch is de programmeer omgeving die gebruikt wordt om iPhone applicaties mee te ontwikkelen.

Code word geschreven in Objective-C. Dit is een specifieke schrijfwijze van code. Veel programmeurs die Objective-C code voor het eerst zien schrikken een beetje omdat er in de syntax¹ veel blokhaakjes voorkomen.

```
NSObject * nieuwObject = [[NSObject  
alloc] initWithObject:[self delegate]];
```

Zoals je ziet komen er in een toewijzing van een nieuwe pointer een aantal blokhaakjes voor. Bij deze toewijzing word een stukje werkgeheugen vrijgemaakt om er vervolgens de waarde van het object delegate in te stoppen. En vervolgens wordt er een referentie gegeven naar dit stukje geheugen. Deze referentie noemen we een pointer.

Gericht op objecten

Als je aan het programmeren bent met Cocoa werk je vrijwel alleen maar met objecten. Dit word object-georiënteerd programmeren genoemd. Objective-C is op C gebouwd en ondersteunt alle C code. Er kan dus ook procedure gericht worden geprogrammeerd.

De term object is op zichzelf natuurlijk ietwat nietszeggend. En dat is een object in Cocoa eigenlijk ook. Je kan pas echt een object begrijpen als je de blauwdruk van het object kent. De blauwdruk wordt een class genoemd.

Bij deze een voorbeeld van een class:

```
@interface Persoon : NSObject {  
    string voornaam;  
    string achternaam;  
    int leeftijd;  
}  
@end
```

In dit geval is de class naam Persoon. En het object persoon kan een voornaam, achternaam en leeftijd bevatten. Voornaam en achternaam zijn strings omdat er als waarde een stuk tekst toegewezen moet worden. Leeftijd is een integer omdat de er als waarde een geheel getal toegewezen moet worden.

Methods

Op dit moment kan er echter nog niets worden toegewezen worden. Hiervoor moeten zogenaamde methods aangemaakt worden. Methods lijken heel erg op functies in het procedure gericht programmeren. Op functies komen wij later nog terug.

```
@interface Persoon : NSObject {  
    string voornaam;  
    string achternaam;  
    int leeftijd;  
}  
@end  
  
@implementation Persoon  
  
-(void)setVoornaam:(string)aVoornaam  
andAchternaam:(string)aAchternaam {  
    voornaam = aVoornaam;  
    achternaam = aAchternaam;  
}  
  
-(BOOL)setLeeftijd:(int)aLeeftijd {  
    if(aLeeftijd < 0) {  
        return NO;  
    }  
    else if (aLeeftijd > 150) {  
        return NO;  
    }  
    else {  
        leeftijd = aLeeftijd;  
        return YES;  
    }  
}  
@end
```

Nu zijn er 2 functies aangemaakt. Met eentje kan de voor- en achternaam toegewezen worden. En met de andere kan dit zelfde gebeuren met de leeftijd. De eerste method stuurt geen resultaat terug, dit wordt

aangegeven met void. Daarin tegen doet de tweede method dit wel. De tweede method stuurt namelijk een boolean terug. Een boolean is een variabele die een waarde kan aannemen van 1 of 0. Cocoa staat echter ook toe om YES en NO te gebruiken. In andere programmeertalen is het gebruikelijk om TRUE en FALSE te gebruiken voor booleans.

De tweede method zal ook checken of de persoon geen negatieve leeftijd heeft of dat de persoon niet ouder is dan 150. Dit gebeurt met een if-else structuur. Als een conditie waar is zal de code tussen de accolades uitgevoerd worden. Dus in dit geval als iemand jonger is dan nul word de code tussen de accolades van de eerste if uitgevoerd. Als de conditie echter niet waar is dan zal door worden gegaan met de else if conditie. En als die weer niet waar is zal de code tussen de accolades achter else worden uitgevoerd.

Om nu een nieuwe persoon aan te maken zal de volgende code uitgevoerd moeten worden. Dit is waar de hoekhaakjes weer terug komen.

```
Persoon *thijs = [[Persoon alloc] init];
[thijs setVoornaam: @"Thijs"
andAchternaam: @"Scheepers"];
[thijs setLeeftijd:18];
```

In de eerste regel code word het object thijs aangemaakt. En in de overige code word er informatie aan het object toegevoerd.

Hiërarchische verdeling van objecten

In het voorbeeld van het toewijzen van een class stond de volgende regel code:

```
@interface Persoon : NSObject {
```

In deze regel staat dat de class Persoon alle eigenschappen erft van de class NSObject. In dit geval is NSObject de super-class van Persoon. NSObject staat in Cocoa altijd boven aan de hiërarchie.

NSObject zorgt er bijvoorbeeld voor dat de alloc en init methods uitgevoerd kunnen worden op elk willekeurig object.

Hier volgt een voorbeeld dat het nut van een hiërarchische verdeling zal laten zien. Wij maken een nieuwe class aan getiteld Nederlander.

```
@interface Nederlander : Persoon {
    int burgerServiceNummer;
}
```

Nu kan er nog steeds een voornaam, achternaam en leeftijd worden toegewezen aan een object met de class Nederlander. Maar een Nederlander heeft een burgerServiceNummer maar dat heeft een Persoon niet.

OpenGL

In tegenstelling tot de meeste code geschreven voor StarDust is de code voor het creëren en manipuleren van de 3D-omgeving, proceduregeoriënteerde code. OpenGL staat voor Open Graphics Library en is een set met procedures voor het maken van 3D omgevingen en deze vervolgens op het scherm te projecteren.

OpenGL maakt veel gebruik van zogenaamde structs. Dit zijn verzamelingen van een vast aantal objecten. Als wij bijvoorbeeld een 3 dimensionale coördinaat willen defineren doen wij dat als volgt:

```
typedef struct {
    GLfloat x;
    GLfloat y;
    GLfloat z;
} Vertex3D;
```

Hier word een struct Vertex3D gedefinieerd. Deze struct bestaat uit een x-waarde, een y-waarde en een z-waarde. Vertex3D kan dus worden gebruikt als een coördinaat in het drie dimensionale vlak. Maar dit hoeft niet, je kan er namelijk elk willekeurig groepje van 3 getallen in gooien.

OpenGL heeft zijn eigen data types. Zoals we zojuist al int, bool en string hebben besproken hebben we in deze struct te maken met een float. Een float is een floating point value, wat betekend dat het dus een getal kan zijn met veel cijfers achter de komma. OpenGL heeft dus echter zijn eigen types. Dit is om beter

gebruik te maken van de beschikbare hardware en daarnaast zijn alle OpenGL types op alle platformen hetzelfde. Dus dat houdt in dat code uit een willekeurig iPhone programma zo naar een Desktop programma gekopieerd kan worden, en er niet veel veranderd hoeft te worden om het op dezelfde manier te laten werken.

Functies

Nu hebben wij zojuist een nieuwe struct aangemaakt. Wat in zichzelf ook een nieuw data type is. Echter hebben wij niet echt veel gewonnen met het aanmaken van alleen een struct. Het is namelijk pas efficiënt als er echt veel regels code mee bespaard kunnen worden. Vandaar dat we ook een functie moeten schrijven om een instantie van de struct te vullen. Een functie lijkt op een method, alleen is een functie niet specifiek aan een object gebonden.

```
static inline Vertex3D Vertex3DMake
(GLfloat inX, GLfloat inY, GLfloat inZ)
{
    Vertex3D ret;
    ret.x = inX;
    ret.y = inY;
    ret.z = inZ;
    return ret;
}
```

Hier maken wij een functie om een 3D coördinaat te vullen met waarden. We vragen aan de gebruiker 3 waarden in te voeren. Vervolgens instantiëren wij een nieuwe 3D coördinaat genaamd `ret` en vullen deze met de ingevoerde waarden. Vervolgens geven wij de aangemaakte 3D coördinaat terug.

Als we deze functie willen gebruiken zullen wij de volgende regel code moeten toepassen.

```
Vertex3D vertex1 = Vertex3DMake(0.0,
1.0, -3.0);
```

Nu hebben wij dus de nieuwe coördinaat genaamd `vertex1` met waarden $(0, 1, -3)$.

```
Vertex3D vertex2 = Vertex3DMake(1.0,
0.0, -3.0);
Vertex3D vertex3 = Vertex3DMake(-1.0,
0.0, -3.0);
Triangle3D triangle = Triangle3DMake
(vertex1, vertex2, vertex3);
```

Als wij de rest van de code nu ook toepassen kunnen wij op een driehoek construeren. Dit is een driehoek met 3 verschillende coördinaten. Daarnaast hebben wij hier ook een nieuwe struct gebruikt genaamd `Triangle3D` die weer op zijn beurt bestaat uit drie `Vertex3D`'s. De code voor het aanmaken van deze struct zal jullie bespaard blijven.

Tekenen

Nu om dit alles toe te passen en op het scherm te tekenen wordt de volgende code uitgevoerd. Wij zullen bij elke regel code uitleg geven.

```
glLoadIdentity();
```

Het eerste wat we doen is de identiteiten matrix inladen. Dit zorgt er voor dat dat alle matrix rotaties en transformaties terug gezet worden. Dit is eigenlijk een grote reset knop.

```
glClearColor(0.7, 0.7, 0.7, 1.0);
```

Hier stellen wij de kleur van de achtergrond in. De functie neemt 4 waarden waarvan de eerste 3 de rood-groen-blauw waarden zijn. Deze waarden hebben een domein van $[0;1]$, waarbij 1 wit en 0 zwart vertegenwoordigt. In dit geval zullen wij dus een lichte grijze kleur krijgen. De laatste waarden die de functie aanneemt heeft de transparantie van de kleur aan. Deze kleur is niet transparant want de waarde is 1.

```
glClear(GL_COLOR_BUFFER_BIT
GL_DEPTH_BUFFER_BIT);
```

Deze functie zorgt er voor dat de volgende buffers geleegd worden. De buffer voor kleur en de buffer voor diepte. Buffers zijn stukjes geheugen speciaal vrij gemaakt voor OpenGL, deze gebruikt OpenGL voor het tekenen van de objecten. De kleur buffer bevat informatie over alle pixels op het scherm. En de diepte buffer bevat alle informatie over objecten in OpenGL en bevat ook de informatie over hun status. Dit is nodig om te zien of de objecten getekend moeten worden of niet.

```
glEnableClientState(GL_VERTEX_ARRAY);
```

Nu activeren wij OpenGLs mogelijkheid om gemakkelijk te werken met vertex array's. Dit betekend dat OpenGL zal gaan werken met rijen van coördinaten. Een array is een rij met verschillende waarden er in. Onze structs kunnen dus ook als een array worden gebruikt. Dit levert echter veel onoverzichtelijker en onduidelijkere code op.

```
glColor4f(1.0, 0.0, 0.0, 1.0);
```

Nu stellen we de kleur van onze driehoek in. Dit zal rood zijn omdat de eerste waarde hoog is en de tweede en derde laag. glColor4f gebruikt net als glClearColor gebruik van 4 waarden waarvan de eerste drie de rood-groen-blauw waarden zijn en de laatste de transparantie.

```
glVertexPointer(3, GL_FLOAT, 0,  
&triangle);
```

Nu geven OpenGL aan dat triangle gebruikt moet worden voor het tekenen van de driehoek. In onze struct triangle hebben wij namelijk alle waarden gestopt. Daarnaast vertellen wij OpenGL dat hij moet rekenen op het gebruik van GLfloats in gegeven waarden.

```
glDrawArrays(GL_TRIANGLES, 0, 9);
```

Vervolgens vertellen wij OpenGL dat wij daadwerkelijk een 3 hoek willen tekenen en dat deze bestaat uit 9 waarden. Namelijk drie maal x,y en z waarden.

```
glDisableClientState(GL_VERTEX_ARRAY);
```

En om af te sluiten zetten wij de mogelijkheid uit voor het tekenen van coördinaten met behulp van rijen. Dit doen wij zodat er op een later moment in de code geen conflict kan ontstaan met een mogelijk andere ClientState.

En als wij deze code nu netjes uitvoeren krijgen wij het volgende resultaat:

Uit deze regels code is goed het verschil te zien tussen procedure georiënteerd programmeren en object

georiënteerd programmeren. Bij de code voor OpenGL is het van belang dat eerst de kleur ingesteld wordt en daarna pas de driehoek getekend wordt. Anders zal de driehoek geen kleur hebben. De volgorde hier is heel erg belangrijk. In tegenstelling tot object georiënteerd programmeren waarbij je een driehoek aanmaakt die vervolgens dimensies en kleur geeft en deze vervolgens in een keer tekent met telkens een verwijzing naar die specifieke driehoek. Je zal dus weten waar je mee bezig bent en welk object je een bepaalde kleur geeft. Bij de code die wij zojuist geschreven hebben is dat niet van belang maar alleen de volgorde.

Object georiënteerd programmeren is daardoor dus ook een stuk overzichtelijker. En als je te maken hebt met een programma met 10.000 regels code, een programma als StarDust. Dan kan procedure georiënteerd programmeren je aardig frustreren omdat je gewoon snel de draad kwijt bent.

De combinatie telt

Als wij nu OpenGL combineren met Cocoa Touch zijn er oneindig veel toepassingen te bedenken die gemakkelijk te bouwen zijn. Door de snelle performance van OpenGL te combineren met het gemakkelijk programmeren in Cocoa Touch is het heerlijk om zo'n programma te schrijven.

Dus hebben wij gekozen voor het gebruiken van deze 2 raamwerken om onze applicatie te laten sprankelen.



TOEGEPASTE WISKUNDE

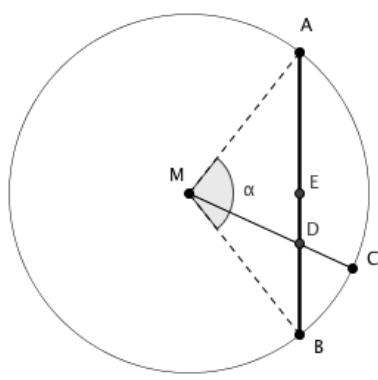
De wiskunde achter het aanklikken van een object in de 3D ruimte.

Het beeldscherm van de iPhone moet worden gezien als een venster in de 3-dimensionale wereld waarin alle sterren staan afgebeeld.

De sterren die hier in afgebeeld staan zijn allemaal geroteerd naar een bepaalde positie. Dit is voor verschil in tijd tussen nu en de epoch¹. Hiernaast moet er geroteerd worden naar de locatie van de iPhone op aarde.

Echter als het scherm wordt aangeraakt moet het overeenkomstige punt op de projectiebol gevonden worden om dit punt vervolgens terug te roteren met alle rotaties die gedaan zijn wat betreft locatie en tijd. Op deze manier kan een aangeklikt punt worden vergeleken met een punt in de sterrendatabase om vervolgens het programma te vertellen dat een bepaalde ster is aangeklikt.

Een versimpelde wiskundige uitwerking, 2D voorstelling.



Gegeven is een waarde p tussen $[-240 ; 240]$. Omdat het scherm van de iPhone 480 pixel breed is. Daarnaast is ook hoek α gegeven. We nemen voor de cirkel met middelpunt M een straal van 1.

Eerst bekijken we welke formules er gelden voor de lijnstukken.

$$AE = BE = \sin\left(\frac{\alpha}{2}\right)$$

$$ME = \cos\left(\frac{\alpha}{2}\right) = x_D$$

$$ED = \frac{p \cdot AE}{240} = y_D$$

Nu maken wij een nieuwe cirkel door punt D. Deze cirkel heeft hetzelfde middelpunt als onze originele cirkel.

$$\sqrt{x_D^2 + y_D^2} = r$$

$$x_C = \frac{x_D}{r} \wedge y_C = \frac{y_D}{r}$$

Nu gebruiken we de verhouding van de straal om punt C uit te rekenen.

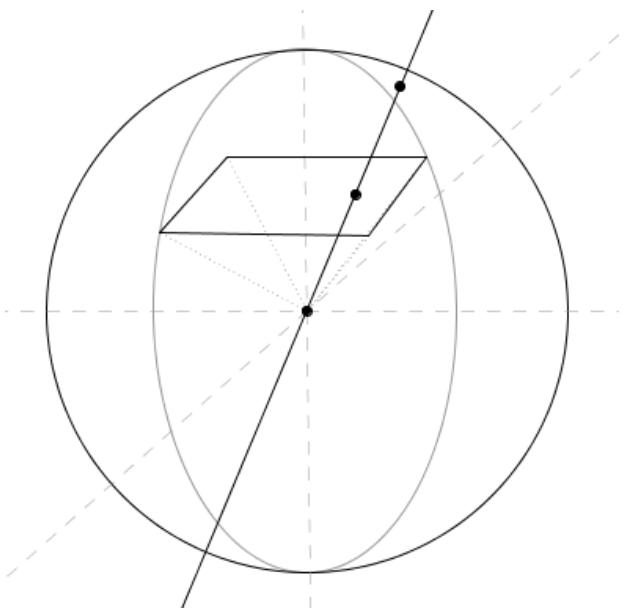
Als wij alle berekeningen samenvoegen resulteert dit in:

$$x_C = \frac{\cos\left(\frac{\alpha}{2}\right)}{\sqrt{\cos^2\left(\frac{\alpha}{2}\right) + \left(\frac{p \cdot \sin\left(\frac{\alpha}{2}\right)}{240}\right)^2}} \wedge y_C = \frac{240 \cdot \sin\left(\frac{\alpha}{2}\right)}{p \cdot \sqrt{\cos^2\left(\frac{\alpha}{2}\right) + \left(\frac{p \cdot \sin\left(\frac{\alpha}{2}\right)}{240}\right)^2}}$$

$$x_C = \frac{\cos\left(\frac{\alpha}{2}\right)}{\sqrt{\cos^2\left(\frac{\alpha}{2}\right) + \frac{p^2 \cdot \sin^2\left(\frac{\alpha}{2}\right)}{57600}}} \wedge y_C = \frac{240 \cdot \sin\left(\frac{\alpha}{2}\right)}{p \cdot \sqrt{\cos^2\left(\frac{\alpha}{2}\right) + \frac{p^2 \cdot \sin^2\left(\frac{\alpha}{2}\right)}{57600}}}$$

Een uitgebreide wiskundige uitwerking, 3D voorstelling.

Nu zullen wij dezelfde berekening uit moeten voeren in een driedimensionale ruimte. Dus gegeven is p tussen $[-240 ; 240]$. En q tussen $[-180;180]$. Opnieuw is de kijkhoek α .



Hierboven staat een voorstelling van de omgeving. En hoe deze via het vlak de bol snijdt.

$$a = \sin\left(\sqrt{\left(\frac{\alpha \cdot 480}{320}\right)^2 + \left(\frac{\alpha \cdot 480}{320}\right)^2}\right)$$

Variabele a rekenen we op deze manier uit. De stelling van Pythagoras gebruiken wij omdat hoek α de hoek is tussen het centrum van de bol en de diagonaal van het vlak.

$$x_D = \frac{p \cdot a}{\sqrt{\left(\frac{480}{320}\right)^2 + \left(\frac{480}{320}\right)^2}}$$

$$y_D = \frac{q \cdot a}{\sqrt{\left(\frac{480}{320}\right)^2 + \left(\frac{480}{320}\right)^2}}$$

$$z_D = \cos\left(\sqrt{\left(\frac{\alpha \cdot 480}{320}\right)^2 + \left(\frac{\alpha \cdot 480}{320}\right)^2}\right)$$

En op deze manier rekenen wij de coördinaten van punt D uit. Punt D is het punt op het vlak.

Vervolgens kunnen we met dezelfde truc ook punt C weer berekenen. Punt C is het punt op de bol. In dit geval gebruiken we geen cirkel maar natuurlijk zijn 3 dimensionale equivalent, een bol.

$$\sqrt{x_D^2 + y_D^2 + z_D^2} = r$$

$$x_c = \frac{x_D}{r} \wedge y_c = \frac{y_D}{r} \wedge z_c = \frac{z_D}{r}$$

Als we al deze functies samenvoegen krijgen we een nieuwe ontzettend lange formule. Deze is helaas niet echt overzichtelijk.

Tijdens het programmeren kunnen deze wiskundige vraagstukken ook in delen opgeschreven worden. In onze code wordt dit dus ook gedaan.

De wiskunde achter het dubbel klikken en inzoomen op een object in de 3 dimensionale ruimte.

In StarDust kan worden ingezoomd door dubbel op het scherm te klikken. StarDust onthoud bij deze handeling de positie op het scherm van de laatste aanraking.

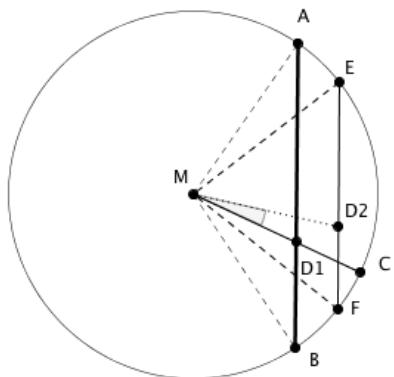
Dit inzoomen is zo gemaakt dat er op een object in de hoek geklikt kan worden en dit object onder de vinger van de gebruiker blijft staan. Er wordt dus niet simpel naar binnen gezoomd waarbij het centrum hetzelfde blijft.

Deze simpele actie wordt hierdoor in een klap wat gecompliceerder en er moet dus wat wiskunde aan te pas komen. Gelukkig hebben wij net al uitgelegd hoe coördinaten op een bol kunnen worden uitgerekend door middel van projectie.

Een versimpelde wiskundige uitwerking, 2D voorstelling.

Om de draai te maken zal de kijkhoek kleiner worden. En dus zal lijnstuk AB langzaam kleiner worden en verder van het middelpunt af komen te staan. En lijnstuk EF ontstaat. Met het aangeklikte punt wat we zojuist berekend hebben met de wiskunde in de vorige paragraaf kunnen wij punt D1 en C construeren. Lijnstuk ED2 staat in verhouding met lijnstuk AD1, en hetzelfde geld voor D1B en D2F. Nu om vervolgens moeten wij het nieuwe vlak EF roteren zodat lijn MD2 samenvalt met MD1. Nu

ontstaat er dus weer een nieuw lijnstuk dat het nieuwe venster moet voorstellen.



De wiskundige uitwerking is als volgt:

$$AE = BE = \sin\left(\frac{\alpha}{2}\right)$$

$$ME = \cos\left(\frac{\alpha}{2}\right) = x_{D1}$$

$$ED = \frac{p \cdot AE}{240} = y_{D1}$$

Eerst berekenen wij de lijnstukken weer op dezelfde manier als tijdens het selecteren.

$$\sqrt{x_{D1}^2 + y_{D1}^2} = r$$

$$x_c = \frac{x_{D1}}{r} \wedge y_c = \frac{y_{D1}}{r}$$

We bereken weer punt C op de bol. Ditzelfde doen we voor punt D2 en hier voor berekenen wij het bijbehorende punt op de bol. Deze noemen wij voor het gemak maar even C2. Met het middelpunt tussen de punten C en C2 kunnen wij vervolgens met behulp van sinus de hoek uitrekenen.

$$CM = \sqrt{(x_c - x_{c2})^2 + (y_c - y_{c2})^2}$$

$$\gamma = 2 \sin(CM / 1)$$

Nu kunnen wij met behulp van deze rotatie het venster roteren en een nieuwe lijn E'F' creëren.

Ditzelfde principe hebben wij toegepast in de 3 dimensionale omgeving. En wij hebben dit op dezelfde manier uitgerekend als het vorige 3D voorbeeld.

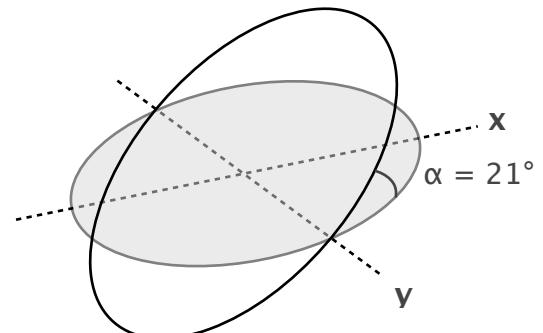
De hemelbol

Alle objecten aan de lucht, van de planeten tot de zon en de maan, hebben een positie op een denkbeeldige bol die we de hemelbol noemen. In werkelijkheid staan sterren verder dan planeten, en planeten verder dan de maan – ze zijn echter allemaal te zijn als lichtbronnen en hun afstand tot de aarde is voor hun positie aan de hemel verder helemaal niet belangrijk.

In dit hoofdstuk trachten wij wiskundig te beschrijven hoe je de positie van verschillende objecten op deze hemelbol uit kan rekenen.

De verschillende vlakken

Binnen de hemelbol zijn twee vlakken te onderscheiden:



figuur 1: Het equatoriale vlak, en de

Het equatoriale vlak, en de ecliptica. De ecliptica is het vlak die de baan van de aarde maakt als hij om de zon draait. Het equatoriale vlak is het vlak dat gemaakt wordt wanneer de evenaar doorgetrokken wordt, en deze geprojecteerd wordt op de hemelbol. Omdat de as van de aarde niet loodrecht op zijn baan staat zijn deze twee vlakken niet gelijk, en hebben ze een bepaalde hoek ten opzichte van elkaar (α in figuur 1). Bij dit figuur is het assenstelsel gekozen die de ecliptica als fundamenteel heeft.

De wetten van Kepler

De baan die de planeten maken om de zon worden beschreven door Kepler's drie wetten, die ook al in de introductie genoemd zijn.

- I. De planeten draaien rond de zon in elliptische banen, met de zon als centrum in een van de twee brandpunten van de ellips.
- II. Een denkbeeldige lijn van de zon naar een planeet toe legt gelijke oppervlakken af in gelijke tijd terwijl de planeet zijn baan maakt om de zon.
- III. Het kwadraat van de periode van een planeet is evenredig met zijn afstand tot de zon tot de derde macht. Wanneer er voor afstand gekozen wordt voor de Astronomische Eenheid (AU = de (gemiddelde) afstand van de aarde tot de zon) is dit te schrijven als:

$$(1): P^2 = a^3$$

Deze wetten betekenen dat je met een paar gegevens over een planeet al snel zijn baan kan berekenen.

De positie van de planeten

Bij het berekenen van de positie van een bepaalde planeet maak je gebruik van de vergelijking van Kepler:

$$(2): M = E - e \sin(E)$$

Hierbij is e de excentriciteit, E de anomalie van de excentriciteit, en M de middelbare anomalie.

De excentriciteit van een ellips is gegeven als

$$(3): e = \sqrt{1 - \frac{a^2}{b^2}}$$

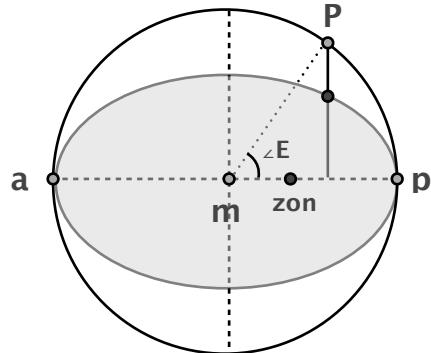
Hierbij is a de afstand van het middelpunt tot het lange eind van de ellips (**mp** in figuur 2) en b de afstand tot het korte eind van de ellips. Bij een ellips geldt:

$$(4): c^2 = a^2 - b^2$$

Hierbij is c de afstand van het middelpunt tot een van de twee brandpunten (**mz** in figuur 2), en geldt dus: $b = \sqrt{(mp^2 - mz^2)}$.

Het punt P is het punt waar de loodrechte die op de lange as van de ellips staat en die door de planeet heen loopt, de omschrijvende cirkel snijdt. Het punt p , de perihelion, is het punt waar de planeet in zijn baan het dichtste bij de zon staat. De aphelion, punt a , is het punt waar de planeet het verste van de zon weg staat.

De excentrische anomalie, E , is gedefinieert als de hoek tussen de perihelion, het middelpunt van de ellips m , en punt P .



figuur 2: Kepler's vergelijking

De hoek M , de middelbare anomalie, is de hoek die een denkbeeldige planeet af zou hebben gelegd op het tijdstip t , wanneer zijn hoeksnelheid gelijk zou zijn aan 2π rad gedeeld door de periode. M is deze snelheid maal de tijd die verstrekken is sinds hij voor het laatst bij de perihelion is geweest.

$$(5): M = \frac{2\pi(t - T)}{P}$$

Hierbij is t de tijd waarop je de hoek die de denkbeeldige planeet heeft afgelegd wilt weten. T het tijdstip waarop hij voor het laatst punt p gepaseerd heeft. En P is in dit geval de periode (omloopijd) van de planeet. Nu M bekend is kan je E benaderen d.m.v. een numerieke methode.

Om de plaats van de echte planeet te weten moet je de ware anomaliteit, v , berekenen. Dit is de werkelijke hoek tussen de perhelium, de zon, en de planeet.

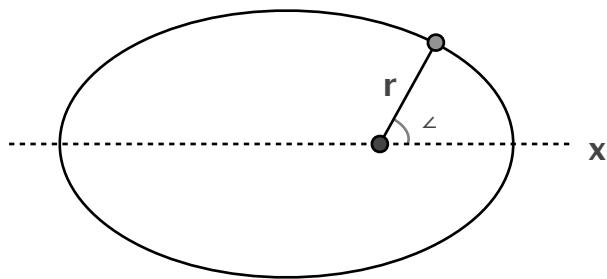
v volgt uit de excentrische anomalie volgens de formule:

$$(6): v = 2 \arctan \left(\sqrt{\frac{1+e}{1-e}} \tan \left(\frac{E}{2} \right) \right)$$

Om nu de afstand van deze planeet tot de zon te vinden gebruik je de volgende formule:

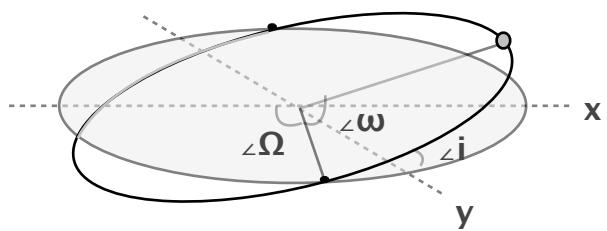
$$(7): r = \frac{a(1-e^2)}{1+e\cos(v)}$$

We hebben nu de poolcoördinaten van het punt waar de planeet zich bevindt in het vlak van zijn baan. (zie III: Poolcoördinaten)



figuur 3:
poolcoördinaten van de

Het vlak dat beschreven wordt door de baan van elke willekeurige planeet die rond de zon draait heeft een hoek t.o.v. de ecliptica. Deze hoek noem je de inclinatie i .



figuur 4: toelichting bij
translatie tussen vlakken

De poolcoördinaten zetten we om in carthesische coördinaten:

$$(8): x_p = r \cos(v + \omega)$$

$$(9): y_p = r \sin(v + \omega)$$

Als we de zojuist gevonden coördinaten om moeten zetten naar de coördinaten die beschreven worden in het assenstelsel van de ecliptica moeten we deze roteren, dit doen we aan de hand van een rotatiematrix. Hiermee vinden we de coördinaten X , Y , Z op het equatoriale vlak van de zon.

We willen de x -as zo draaien dat hij richting punt a (het punt van aries) wijst. Voor die draai naar rechts krijgen we een rotatie van de x -as naar de y -as toe over de hoek van Ω , en een rotatie van de z -as richting de x -as over de hoek i , de inclinatie.

Om de rotatiematrix te vinden vermenigvuldigen we de twee rotatiematrixen. (zie I: Matrixen), we gebruiken een rechtshandig coördinaten systeem waardoor we de twee hoeken negatief moeten maken.

We vinden:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-i) & -\sin(-i) \\ 0 & \sin(-i) & \cos(-i) \end{pmatrix} \begin{pmatrix} \cos(-\Omega) & -\sin(-\Omega) & 0 \\ \sin(-\Omega) & \cos(-\Omega) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(i) & \sin(i) \\ 0 & -\sin(i) & \cos(i) \end{pmatrix} \begin{pmatrix} \cos(\Omega) & \sin(\Omega) & 0 \\ -\sin(\Omega) & \cos(\Omega) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\begin{pmatrix} \cos(\Omega) & \sin(\Omega) & 0 \\ -\sin(\Omega)\cos(i) & \cos(\Omega)\cos(i) & \sin(i) \\ \sin(i)\sin(\Omega) & -\sin(i)\cos(\Omega) & \cos(i) \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (10)$$

Uitwerken geeft:

$$X_p = x \cos(\Omega) - y \sin(\Omega) \cos(i)$$

$$Y_p = x \sin(\Omega) + y \cos(\Omega) \cos(i)$$

$$Z_p = y \sin(i)$$

x en y invullen en herleiden (z is nul, en valt dus weg) geeft:

$$X_p = r(\cos(\omega + v) \cos(\Omega) - \sin(\omega + v) \sin(\Omega) \cos(i))$$

$$Y_p = r(\cos(\omega + v) \sin(\Omega) + \sin(\omega + v) \cos(\Omega) \cos(i))$$

$$Z_p = r \sin(\omega + v) \sin(i)$$

Omdat i tenopzichte van het ecliptische vlak staat, en we de equatoriale coördinaten willen hebben moeten we nogmaals met een rotatie matrix vermenigvuldigen, hier is i_g de inclinatie van de ecliptica t.o.v. het equatoriale vlak (α in **figuur 1**):

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-i_g) & -\sin(-i_g) \\ 0 & \sin(-i_g) & \cos(-i_g) \end{pmatrix} \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} = \begin{bmatrix} X_e \\ Y_e \\ Z_e \end{bmatrix}$$

Zo vinden we de coördinaten t.o.v. het equatoriale vlak, nadat de negatieve hoeken herleid zijn:

$$\begin{aligned} X_e &= X_p \\ Y_e &= Y_p \cos(i_g) - Z_p \sin(i_g) \\ Z_e &= Z_p \cos(i_g) + Y_p \sin(i_g) \end{aligned}$$

Aangezien we de positie van de planeten vanuit de aarde willen weten moeten we deze van heliocentrisch (de zon als oorsprong) naar geocentrisch (de aarde als oorsprong) omzetten. We doen dit d.m.v. de coördinaten van de aarde (die we vinden met de voorgaande methode) bij X_e , Y_e en Z_e op te tellen.

$$\begin{aligned} x_{\text{planeet}} &= X_e + x_\odot \\ y_{\text{planeet}} &= Y_e + y_\odot \\ z_{\text{planeet}} &= Z_e + z_\odot \end{aligned}$$

Nu we de geocentrische coördinaten gevonden hebben, willen we weten in welke windrichting de planeet staat, en hoe hoog hij staat boven de horizon.

Om dit te berekenen moeten we eerst de positie van de planeten op de hemelbol berekenen. Deze worden gegeven in bolcoördinaten.

We gebruiken de formules om carthesische coördinaten om te zetten in bolcoördinaten (zie **II: Bolcoördinaten** voor bewijs en uitleg).

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\varphi = \arctan 2(y, x)$$

$$\theta = \arccos\left(\frac{z}{r}\right)$$

Hiermee vinden we, in bolcoördinaten, de positie van een planeet op de hemelbol. Dit noemen we de rechte klimming (horizontale coördinaat), en de declinatie (verticale coördinaat). Naast deze coördinaten heb je nog een derde coördinaat: de afstand r , die je als waarnemer, vanwege de enorme afstanden, niet kunt schatten. In het programma hebben wij voor elke ster een vaste waarde van r gekozen, zo staan deze dan ook daadwerkelijk op één en dezelfde bol.

Sterrenposities

Nu we een wiskundig model hebben voor het berekenen van de posities van de planeten t.o.v. de aarde, willen we deze planeten natuurlijk weten waar deze planeten staan ten opzichte van de sterren.

De positie van een ster wordt gegeven met enkel twee coördinaten, de bij de planeten al genoemde rechte klimming en declinatie. Deze twee hoeken geven de positie aan op de hemelbol, net als de coördinaten die we eerder vonden bij de planeten. Dit zijn dus eveneens bolcoördinaten.

Omdat de positie van de sterren in het tijdspan waarin wij geïnteresseerd zijn (tussen 2000 en 2100) nagenoeg niet verandert, hoeven we verder geen berekeningen te doen.

Sterrentijd

Om te berekenen waar de sterren staan op een bepaald tijdstip maak je niet gebruik van de tijd die wij in het dagelijkse leven gebruiken. Je maakt gebruik van een ander soort tijd die afhankelijk is van de draaiing van de aarde om zijn as, de sterrentijd.

Deze tijd wordt gemeten door de stand van de sterren. De rechte klimming van een ster die op dat moment door de hemelmeridiaan gaat, geeft de sterrentijd aan. Bij het berekenen van de sterrentijd zijn de volgende variabelen van belang: Allereerst ΔJ dit is het aantal dagen

sinds 00:00 1 Januari 2000, deze datum nemen we dus als referentie punt. Vervolgens L , dit is de heliocentriche gemiddelde longitude van de aarde. We gaan verder niet in op wat dit precies is, maar in de formule komt hij voor als een combinatie van 3 constanten (de anderen verwaarlozen we). l_w is de longitude van je locatie op de aarde. t is de kloktijd, in de lokale tijdzone gemeten in uren. t_z is je tijdzone. Uit eindelijk krijgen we θ , dit is de sterrentijd op jouw locatie, in graden. Om van standaard tijd over te gaan in sterrentijd gebruik je de volgende formules:

$$\theta \equiv \theta_0 + \theta_1 t \bmod 360^\circ$$

$$\theta_0 \equiv L_0 + L_1 \Delta Jd + \theta_p \bmod 360^\circ$$

$$\theta_p \equiv L_2 \Delta Jd^2 + L_3 \Delta Jd^3 - l_w + \theta_1 t_z$$

$$\theta_1 \equiv M_0 + M_1 \Delta Jd + M_2 \Delta Jd^2$$

De waarden van L_n en M_n zijn te vinden in **tabel 1**.

We gaan verder niet in op totstandkoming van deze formule, maar we zullen hem in het getallen voorbeeld wel gebruiken.

Locatie op de aarde

Afhankelijk van waar je bent op de aarde ziet de hemel er erg anders uit. Nu we de positie van de sterren, planeten en de maan hebben op de hemelbol, willen we weten hoe hoog ze staan boven, of hoe laag onder de horizon. En in welke windrichting het object staat. Om dit te berekenen moet je eerst je geografische locatie op de aarde weten.

Je geografische locatie bestaat uit twee waarden, de geografische breedtegraad φ , en je geografische lengtegraad l_w die al eerder aan bod kwam bij het berekenen van de sterrentijd. Verder gebruiken we de uurhoek H waarvoor geldt: $H = \theta - \alpha$. Dit is de hoek die het object voorbij het lente punt is. Ook gebruiken we de declinatie van het object: δ .

De positie aan de hemel wordt gegeven door twee coordinaten: de azimut, en de hoogte. De azimut geeft aan in welke richting het object staat, de hoogte geeft aan hoe hoog het object zich aan de hemel bevindt.

$$\sin A \cos h = \sin H \cos \delta$$

$$\cos A \cos h = \cos H \cos \delta \sin \varphi - \sin \delta \cos \varphi$$

$$\sin h = \sin \varphi \sin \delta + \cos \varphi \cos \delta \cos H$$

Uit deze vergelijkingen volgen de volgende twee formules:

$$A = \arctan(\sin H, \cos H \sin \varphi - \tan \delta \cos \varphi)$$

$$h = \arcsin(\sin \varphi \sin \delta + \cos \varphi \cos \delta \cos H)$$

De arctangens is overigens de al genoemde arctan2 functie, die ervoor zorgt dat (x,y) en $(-x,-y)$ niet dezelfde hoek geven.

Getallen voorbeeld

Als we b.v. willen weten waar Jupiter op de hemelbol staat om 12:00 's middags, op 12 maart moeten we eerst de sterrentijd op dat tijdstip bepalen. Vervolgens lopen we de berekeningen door die gegeven zijn in deze paragraaf:

Voor Jupiter en de Aarde gelden de waarden in **tabel 2**, die achteraan dit hoofdstuk te vinden is.

Dit zijn waarden die gelden vanaf het jaar 2000, en zijn enkel een aantal decennia nauwkeurig. Voor onze doeleinden voldoet dit echter prima.

We berekenen eerst de sterrentijd op 12 maart 2010, om 12 uur 's middags. Hiervoor hebben we het aantal dagen nodig sinds J2000.

$$\Delta J = 2455267.95843 - 2451545 = 3723,95843$$

Invullen in de formules geeft een sterrentijd van:

$$\theta = 339,9083333^\circ$$

Vervolgens berekenen we welke hoek de planeet gemiddeld aflegt in zijn baan, gezien vanaf de zon. Hiervoor geldt, vanwege de 3e wet van Kepler (hierin is (met $c = 0,985607668$ graden = n aarde ($= \pm 360/365,25$, omdat de omloopstijd van de aarde rond de zon ongeveer 365,25 dagen is)):

$$n = c / (a\sqrt{a})$$

voor Jupiter geldt: $n = 0,083056$
 Voor de Aarde geldt: $n = 0,985608$

vervolgens berekenen we de gemiddelde Anomalie d.m.v. M op de begindatum (M_0). De begindatum, d_0 is wanneer de planeet het dichtst bij de zon stond (perihelium).

$$M = M_0 + n(d - d_0)$$

Voor Jupiter geldt: 328,408637
 Voor de Aarde geldt: 57,090192

Met de middelbare anomalie kunnen we de ware anomalie uitrekenen, d.m.v. de vergelijking van Kepler. Deze lossen we op d.m.v. een iteratie. De formule is namelijk niet in een zodanige vorm te stoppen dat je E kan schrijven als functie van M .

$$M = E - e \sin(E)$$

Voor Jupiter geldt: 326,891051
 Voor de Aarde geldt: 57,9011072

Nu berekenen we de afstand en de ware anomalie van de planeet:

$$v = 2 \arctan \left(\sqrt{\frac{1+e}{1-e}} \tan \left(\frac{E}{2} \right) \right)$$

$$r = \frac{a(1-e^2)}{1+e \cos(v)}$$

Voor Jupiter geldt: $v = -34,658828$, $r = 4,991287$
 Voor de Aarde geldt: $v = 58,715790$, $r = 0,991121$

vervolgens gebruiken we poolcoördinaten (zie **III: Poolcoördinaten**) om tot de carthesische coördinaten (x,y) in het ecliptische vlak te komen)

$$x_p = r \cos(v + \omega)$$

$$y_p = r \sin(v + \omega)$$

Deze gebruiken we in de vergelijkingen:

$$X_p = r(\cos(\omega + v)\cos(\Omega) - \sin(\omega + v)\sin(\Omega)\cos(i))$$

$$Y_p = r(\cos(\omega + v)\sin(\Omega) + \sin(\omega + v)\cos(\Omega)\cos(i))$$

$$Z_p = r \sin(\omega + v)\sin(i)$$

en

$$\begin{aligned} X_e &= X_p \\ Y_e &= Y_p \cos(i_g) - Z_p \sin(i_g) \\ Z_e &= Z_p \cos(i_g) + Y_p \sin(i_g) \end{aligned}$$

Om tot de driedimensionale, en heliocentrische coördinaten van de planeet te komen.

Voor Jupiter geldt: $x = 4,679340$, $y = -1,734131$, $z = -0,097508$
 Voor de Aarde geldt: $x = -0,940738$, $y = 0,311979$, $z = 0$

vervolgens berekenen we de geocentrische coördinaten

$$\begin{aligned} x_{\text{planeet}} &= X_e + x_{\odot} \\ y_{\text{planeet}} &= Y_e + y_{\odot} \\ z_{\text{planeet}} &= Z_e + z_{\odot} \end{aligned}$$

Voor Jupiter geldt: $x = 5,620079$, $y = -2,046110$, $z = -0,097599$

De bolcoördinaten van Jupiter:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \varphi &= \arctan 2(y, x) \\ \theta &= \arccos \left(\frac{z}{r} \right) \end{aligned}$$

Voor Jupiter geldt: RA = 341,885684°, Dec = -8,86039°

Deze waarden geven de positie van Jupiter aan op de hemelbol, op het gegeven tijdstip.

AANVULLENDE WISKUNDE

Er komt bij het berekenen van planeet- en sterrenstanden nogal wat wiskunde kijken die het examenprogramma van Wiskunde B op het VWO niet gehaald hebben. Hier zullen we deze wiskunde kort behandelen.

Matrixvermenigvuldigen

Een matrix is een tweedimensionale lijst met waarden. Met matrixen kan je rekenen. Je kunt ze optellen, vermenigvuldigen, etc. In dit profielwerkstuk zijn we alleen geïnteresseerd in matrixvermenigvuldigingen. Deze gebruiken we om de plaats van een punt in een assenstelsel, om te zetten naar een plaats in een ander assenstelsel. Dit gebeurt met behulp van zogenaamde standaard(rotatie)matrixen. Voor een omwenteling om een bepaalde as gelden de volgende matrixen.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Twee matrixen vermenigvuldigen gaat als volgt. We bekijken de volgende twee matrixen:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} x & y \\ z & q \end{pmatrix}$$

Om deze twee matrixen te vermenigvuldigen gebruik je de volgende regels. De dimensie van de matrix die uit deze vermenigvuldiging komt heeft de lengte (het aantal kolommen) van de 1e matrix, en de lengte (het aantal rijen) van de tweede matrix. Het aantal rijen van de eerste matrix, moet overeenkomen met het aantal kollommen in de tweede matrix anders is de vermenigvuldiging ongeldig (onbepaald)

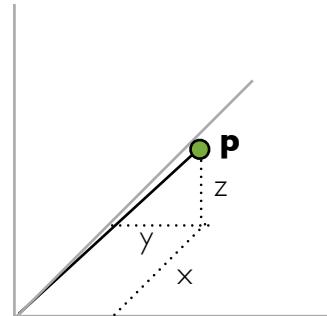
Om de waarde van x te vinden doe je de eerste uit de eerste rij van de eerste matrix, en vermenigvuldig je deze met de 1e uit de eerste kolom van de tweede matrix. Hier tel je het product van het tweede getal van de eerste rij van de eerste matrix, en het tweede getal van de eerste kolom van de tweede matrix bij op. Zo werk je y, z en q af.

Uit deze vergelijking komt dus de volgende matrix:

$$\begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}$$

Bolcoördinaten

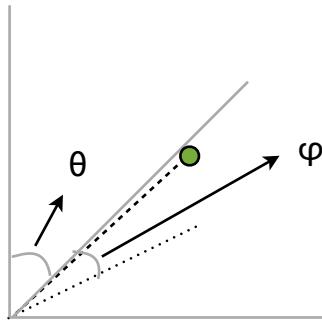
Een bolcoördinaat bestaat uit 3 waarden. Een afstand r, en twee hoeken phi en theta. Om de carthesische coördinaten (x, y, z) om te zetten naar bolcoördinaten bekijken we het volgende figuur:



Uit dit figuur blijkt dat voor de afstand d(O,P) = r geldt:

$$r = \sqrt{\sqrt{x^2 + y^2}^2 + z^2} = \sqrt{x^2 + y^2 + z^2}$$

Nu we de afstand hebben gevonden gaan we op zoek naar de hoeken. De volgende hoeken geven samen de bolcoördinaten:



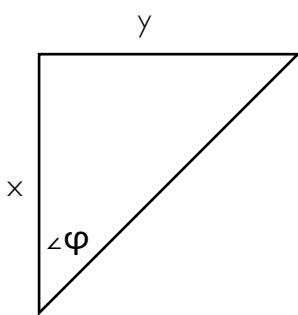
Om van (x,y,z) naar (r,φ,θ) te gaan gebruik je dus het volgende drietal formules:

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\varphi = \arctan 2(y,x)$$

$$\theta = \arccos(z/r)$$

voor φ , de hoek tussen de x as en de lijn die O verbindt met de projectie van p op het x,y -vlak, geldt de volgende driehoek:



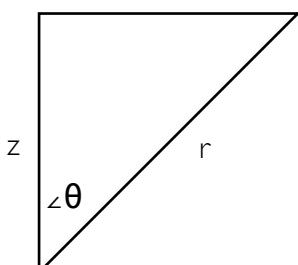
hieruit blijkt:

$$\varphi = \arctan(y/x)$$

omdat dit dezelfde hoek geeft voor (x,y) en $(-x,-y)$ gebruiken we de atan2 functie, die dit probleem verhelpt:

$$\varphi = \arctan 2(y,x)$$

voor theta geldt de volgende driehoek:



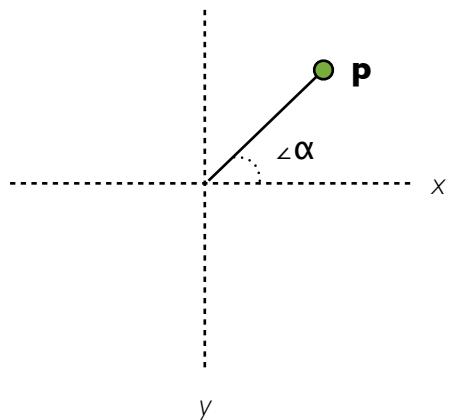
uit deze driehoek blijkt:

$$\theta = \arccos(z/r)$$

Poolcoördinaten

Poolcoördinaten zijn een manier om een punt in een plat vlak, weer te geven met een hoek en een afstand.

De formules voor de omzetting van een punt (x,y) naar zijn poolcoördinaten volgen uit het volgende figuur:



Uit dit figuur is afleiden dat punt $p(x,y)$ de volgende poolcoördinaten heeft:

$$r = \sqrt{x^2 + y^2}$$

$$\angle \alpha = \arctan\left(\frac{y}{x}\right)$$

Omgekeerd geldt voor de coördinaten (x,y) van punt p :

$$x = r \cos(\alpha)$$

$$y = r \sin(\alpha)$$

Tabellen:

Tabel 1:

	0	1	2	3
M	15,041068	$2,42 \cdot 10^{-14}$	$-6,528 \cdot 10^{-23}$	
L	99,9677947	360,985647	$2,907 \cdot 10^{-13}$	$-5,302 \cdot 10^{-22}$

Bron: <http://www.astro.uu.nl/~strous/AA/en/reken/sterrentijd.html>

Tabel 2:

	a	e	i	ω	Ω	M₀
Jup.	5,2026	0,4849	1,303	273,87	100,46	20,020
Aar.	100.000	1.671	0	288.064	174.873	357.529

Bron: <http://www.astro.uu.nl/~strous/AA/nl/reken/hemelpositie.html>

CONCLUSIE

Wij hebben de afgelopen maanden veel geleerd niet alleen over de nachtelijke hemel maar ook over het ontwikkelproces van het bouwen van een applicatie. Wij zijn wijzer geworden en hebben onze programmeer vaardigheden uitgebreid. We hebben naast het programmeren veel geleerd over wiskunde die we voor dit werkstuk nooit gezien hadden, of in ieder geval nooit in toepassing hadden gebracht.

Wij zijn begonnen met het idee een werkstuk te schrijven over de wiskunde achter de hemelmechanica. Toen wij besloten een programma te bouwen hadden wij nooit gedacht dat het zo uitgebreid zou worden als het nu is. Het werkstuk bevat minder wiskunde dan we vooraf verwacht hadden, en de nadruk is langzaam gedurende de afgelopen maanden verschoven van de wiskunde het ontwikkelen van een programma.

Wij hopen van harte dat het programma door veel mensen gedownload en gebruikt gaat worden zodra hij in de App Store belandt in de komende weken.

BRONNEN

Wikipedia

- Johannes Kepler
- Kepler's vergelijking
- Excentriciteit (wiskunde)
- Poolcoördinaten
- True Anomaly
- Object-Georiënteerd Programmeren
- Cocoa (API)

Internet

<http://www.davidcolarusso.com/astro/>
<http://stjarnhimlen.se/comp/ppcomp.html>
<http://developer.apple.com>
<http://iphonedevlopment.blogspot.com>
<http://www.astro.uu.nl/~strous/AA/nl/reken/hemelpositie.html>

Tijdschriften

NWT, November 2009. Blz. 66. – "Oerkosmologie"

Sci. Am., Juli-Augustus 2009. Blz. 56 – "Het verleden van het verre-kijken"

Astronomy, October 2009. Blz. 64. – "Predicting Neptune's existence."

Astronomy, January 2010. Blz. 34 . – "What was the star of Bethlehem?"

Boeken

Van supernova's tot zwarte gaten. Blz. 53. – "De ster van Bethlehem"

50 physics ideas you really need to know. Blz 12. – "Kepler's laws"

50 physics ideas you really need to know. Blz 16. – "Newton's law of gravity"

De bètacanon. Blz. 113. – "Vader van de zwaartekracht"

De bètacanon. Blz. 193. – "Zonnestelsel"

UREN REGISTRATIE

Datum (Thijs)	Uren (Thijs)	Datum (Jan-Willem)	Uren (Jan-Willem)
14 okt 2009	3	14 okt 2009	2
16 okt 2009	1	17 okt 2009	3
20 okt 2009	8	20 okt 2009	8
21 okt 2009	6	21 okt 2009	6
23 okt 2009	3	25 okt 2009	3
28 okt 2009	1	26 okt 2009	3
31 okt 2009	2	31 okt 2009	3
4 nov 2009	1	4 nov 2009	3
5 nov 2009	3	12 nov 2009	2
15 nov 2009	3	18 nov 2009	5
16 nov 2009	1	19 nov 2009	2
22 nov 2009	2	22 nov 2009	1
29 nov 2009	3	29 nov 2009	2
3 dec 2009	1	4 dec 2009	4
9 dec 2009	3	12 dec 2009	3
25 dec 2009	8	25 dec 2009	2
29 dec 2009	5	29 dec 2009	5
2 jan 2010	4	2 jan 2010	2
13 jan 2010	2	13 jan 2010	3
16 jan 2010	4	16 jan 2010	4
17 jan 2010	1	17 jan 2010	3
20 jan 2010	2	7 feb 2010	2
27 jan 2010	3	11 feb 2010	3
6 feb 2010	5	20 feb 2010	1
7 feb 2010	4	21 feb 2010	3
11 feb 2010	3	23 feb 2010	2
20 feb 2010	5	26 feb 2010	3
21 feb 2010	2	27 feb 2010	4
23 feb 2010	2	28 feb 2010	5
26 feb 2010	2	1 mrt 2010	7
27 feb 2010	1	2 mrt 2010	1
28 feb 2010	3		
	97		100

Wij hebben beide op de Havo al een profielwerkstuk gemaakt en waren hierdoor maar verplicht tot het vullen van 40 uur wij hebben echter zo genoten van het maken van het werkstuk en het bouwen van de applicatie dat wij uiteindelijk veel meer uren in het werkstuk hebben gestoken dan aanvankelijk nodig was.