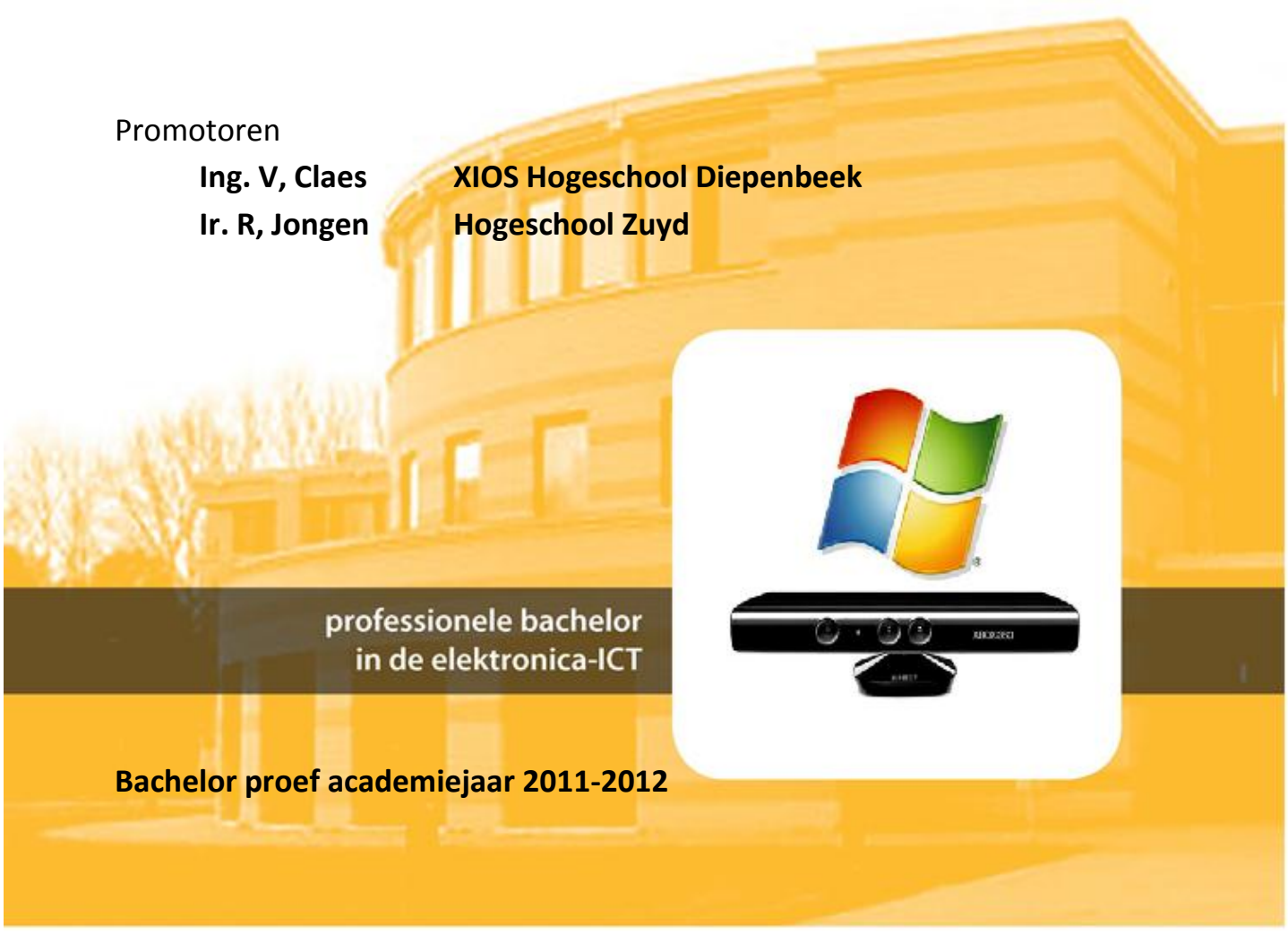
Departement N-Tech

XBOX KINECT-APPLICATIES

Bart Suelze
Kenny Colda



Promotoren
Ing. V, Claes **XIOS Hogeschool Diepenbeek**
Ir. R, Jongen **Hogeschool Zuyd**

professionele bachelor
in de elektronica-ICT

Bachelor proef academiejaar 2011-2012

Departement N-Tech

XBOX KINECT-APPLICATIES

Bart Suelze
Kenny Colda

Promotoren
Ing. V, Claes **XIOS Hogeschool Diepenbeek**
Ir. R, Jongen **Hogeschool Zuyd**

professionele bachelor
in de elektronica-ICT

Bachelor proef academiejaar 2011-2012

Dankwoord

Graag willen wij om te beginnen een dankwoord richten aan de XIOS Hogeschool Limburg en Hogeschool Zuyd voor het mogelijk maken van deze leerrijke stage. Zeker niet te vergeten zijn onze stagebegeleiders Vincent Claes en Ramon Jongen voor hun professionele bijstand wanneer deze vereist was. Ook bedanken wij Servaes Tilken van de informatica afdeling van de XIOS Hogeschool Limburg voor zijn tijd en hulp bij het programmeren in het algemeen en Hubert Hoen voor het verlenen van het geavanceerde robotica lokaal in Hogeschool Zuyd en alle nodige materialen om deze stage tot een goed einde te brengen. Ten laatste een dankwoord aan de vertegenwoordiger van de Biometrie afdeling van hogeschool Zuyd, de heer Hans Sauren, voor de interesse die hij toonde in ons project. Vooral door ons te wijzen op het feit ons project enorm gewaardeerd kan worden in de medische sector en ons te voorzien met waardevolle connecties. Daarom stonden wij hem ook graag bij met woord en programma-applicaties.

Inhoudstafel

1) Abstract

Titel: Xbox Kinect applicaties

Door: Suelze Bart & Colda Kenny
Promotoren:
Dhr. Ramon Jongen Hogeschool Zuyd
Dhr. Vincent Claes XIOS Hogeschool Limburg

De technologie staat niet stil. Het is aan de techneut van vandaag om verscheidene technologieën met elkaar te combineren en gebruiksvriendelijker te maken, niet om weer iets nieuws te ontwikkelen. Onze stage opdracht is in het teken van de Xbox Kinect, uitgebracht door Microsoft in november 2010. De Kinect is ontworpen als plug-in voor de Xbox 360 game console, gebaseerd op een webcam design. Hoewel deze alleen ontworpen is als game applicatie is hij, ondanks zijn goede prijs, tot zoveel meer instaat. Microsoft merkte dit op en daarom hebben ze ook engines ontworpen voor Windows. Nu is het softwarematig mogelijk om de Kinect aan te spreken via een .Net omgeving met C# - taal. Onze stage opdracht bestaat eruit om te onderzoeken of die Kinect ook geïmplementeerd kan worden in sectoren waarvoor hij niet bestemd is. Zowel op het gebied van software als hardware. Wij gebruiken de laatste nieuwe Kinect SDK die is vrijgeven door Microsoft, namelijk Beta 2. Deel één van onze stage:

- opzoekwerk naar de Kinect
- onderzoek naar de basisfuncties en programmeermogelijkheden
- schrijven van basis programma's die deze functies illustreren
- onderzoeken of de resultaten/prestaties die de Kinect weergeeft correct genoeg zijn om te implementeren in andere sectoren dan de game industrie
- nagaan wat de hardware limieten van de Kinect zijn

Na een korte onderzoeksperiode werd het vlug duidelijk dat de Kinect ook zijn nut kan bewijzen in de Industrie, maar vooral in de medische sector. Om dit te bewijzen worden er programma's geschreven, die applicaties nabootsen die in de medische sector geappreciëerd worden, dit wordt deel twee van onze stage. Dit deel van onze stage bestaat eruit aan patiëntbewaking te doen, er word voorkomen dat een zieke of zelfs demente patiënt uit bed valt, doormiddel van een alarmering of waarschuwing. We plaatsen de Kinect boven een zieken bed om zo het skelet van de patiënt nog te kunnen traceren. Het voordeel aan dit skelet is dat het de privacy van de patiënt niet meer schend in tegenstelling tot camerabeelden die dit wel doen. Een tweede applicatie is het meten van de hoek tussen bijvoorbeeld de boven en onderarm, of boven en onderbeen. Zo kan men controleren of het revalidatieproces van een patiënt met gebroken arm of been vorderingen maakt of niet.

2) Lijst met afkortingen

SDK	Software Development Tool
CMOS	Complementary Metal Oxide Semiconductor
RGB	Rood, Groen en Blauw
IC	Integrated circuit
kHz	Kilo Hertz
CD	Compact Disc
USB	Universele Seriële Bus
ISO file	International Organization for Standardization file
DLL	Dynamically Linked Library
PC	Personal Computer

3) Inleiding

Alles draait de dag van vandaag om technologie. Technologie geeft zekerheid, bespaart tijd en moeite, maar geeft vooral luxe. Technologie is ook niet te stoppen en evolueert iedere dag. Iedere programmeur of onderneming komt iedere dag wel op de proppen met een nieuwe vorm van technologie. Het is ook niet meer mogelijk voor één individu om in ons relatief korte levensbestaan alle vormen van technologie onder de knie te krijgen. Daarom is het aan de techneut van de toekomst om een bestaande technologie te implementeren in een gebied waarvoor deze technologie origineel niet was ontworpen, in de plaats van iets nieuws uit te vinden. Wanneer hij dit voor mekaar krijgt zullen we met onze huidige technologie veel meer geavanceerde doeleinden behalen en zal alles ook veel gebruiksvriendelijker worden.

Wij zijn Erasmusstudenten aan hogeschool Zuyd en beschouwen ons een beetje als technici van de toekomst. Omdat wij een stuk hardware, dat origineel enkel en alleen bedoeld was voor de gamesector, wisten te implementeren in de medische sector. We hebben het hier over de Kinect, ontwikkeld door Microsoft, een webcam gebaseerde plug–in voor de Xbox 360 game console. Hiervoor werden applicaties ontwikkeld die dit duidelijk zullen illustreren. Hoewel deze applicaties zich nog bevinden in de onderzoeksfase en dus nog prototypes zijn, zullen personen met de nodige kennis uit de desbetreffende sectoren erkennen dat deze applicaties nuttig zijn en meteen geïmplementeerd kunnen worden.

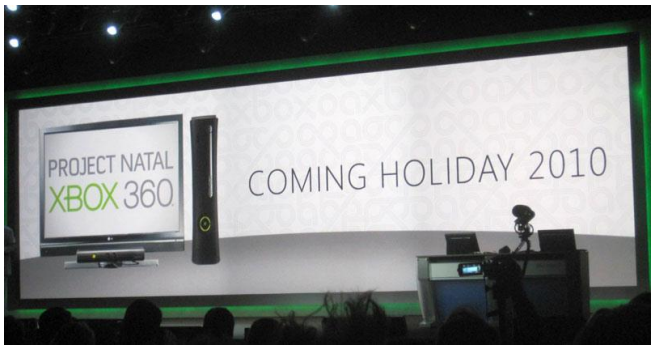
Allereerst wordt een volledige handleiding en beschrijving gegeven om de nodige voorkennis onder de knie te krijgen. Zowel hardware- en softwareonderdelen zullen uitgelegd en besproken worden aan de hand van een handleiding met de nodige eenvoud om stap voor stap een beeld te scheppen over hoe men de nodige drivers en programma's installeert. Verder wordt de programmacode uitgelegd met als doel alle programma's die we geschreven hebben te verklaren. Tot slot zal er ook een CD-ROM ter beschikking zijn met alle geschreven programma's voor de Kinect. Deze is als bijlage meegeleverd bij de scriptie

4) Hardware en omschrijving

In dit hoofdstuk dat volgt wordt verklaard wat een Kinect is en wordt de evolutie van de Kinect weergegeven. Verder zijn er nog allerlei hardware matige componenten die er in de Kinect allemaal aanwezig zijn en besproken worden. Tot slot bespreken we enkele mogelijke applicaties die mogelijk zijn om te programmeren met de Kinect.

4.1 Situatieschets

De Kinect is ontwikkeld door de software- en gameontwikkelaar Microsoft. De Kinect is een beweging detecterend input device voor de Xbox 360 game console. Hij is gebaseerd op een webcam design, maar is zoveel meer. Dankzij de Kinect kunnen Xbox 360- gebruikers interfacen met hun console zonder het gebruik van een externe controller. De Kinect herkent bepaalde handgebaren en spraakcommando's en maakt het zo mogelijk om de applicaties op de Xbox te sturen. Dit maakt hem zeer gebruiksvriendelijk.
Microsoft heeft de Kinect op de internationale markt gebracht in november 2010, op 10 november 2010 was de Kinect verkrijgbaar in Europa. Deze werd eerst project Natal genoemd, maar Natal werd vlug naar Kinect omgevormd als een verwijzing naar kinetische energie.



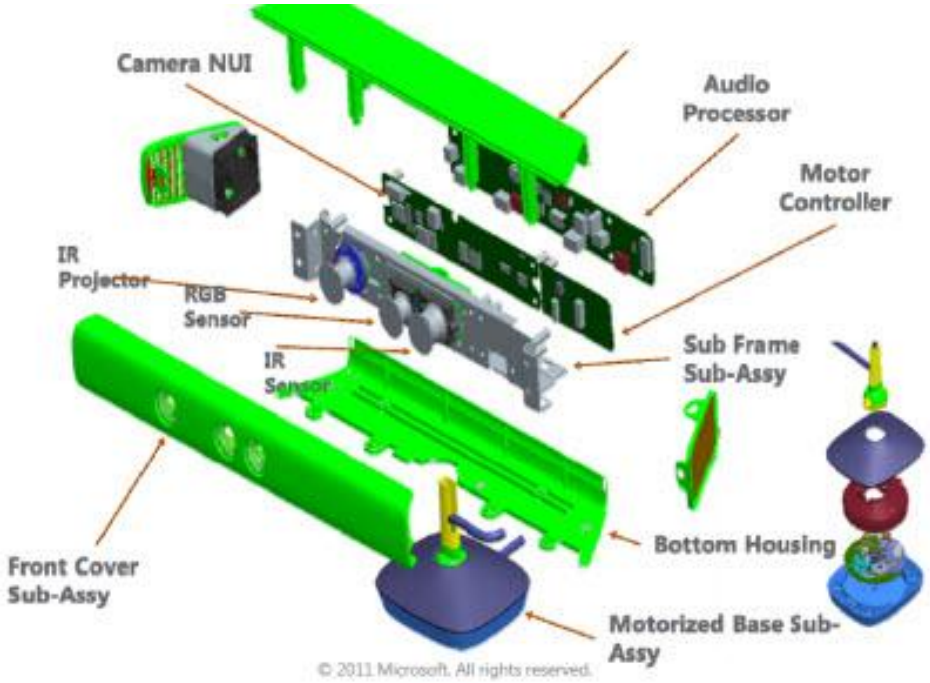
Figuur 1: Kinect project Natal

Nog geen 3 uur na de release van de Kinect werd deze al vlug gehackt. Microsoft wlde de creativiteit van deze programmeurs niet tegen gaan en gaf daarom op 16 juni 2011 SDK's vrij voor Windows 7, deze waren en zijn nog steeds gratis te downloaden. Dankzij deze engines kunnen ontwikkelaars applicaties schrijven in een simpele .Net omgeving. Voor deze stage werd er gebruik gemaakt van de Beta 2- versie van hun eerste SDK, deze is als laatste vrijgegeven toen de Kinect exact één jaar werd, op 4 november 2011. De applicaties worden geschreven in een C#- omgeving van Visual Studio. Deze SDK kan alleen gebruikt worden in een Windows 7- omgeving.

4.2 Hardware en specificaties

De Kinect kost vandaag in Europa ongeveer 144 euro. Het zal vlug duidelijk worden dat deze prijs niets is vergeleken met alles waartoe hij in staat is en de onderdelen waarmee hij is opgebouwd. De belangrijkste eigenschappen die de Kinect bevat zijn: 1) Dieptezicht, 2) Skelet opbouw, 3) Spraakherkenning.

De Kinect heeft deze eigenschappen omdat hij beschikt over een RGB- camera (gezichtsherkenning + video), een dieptesensor om beweging te detecteren (infraroodprojector met monochrome CMOS[1]- camera), verder zijn er nog 4 neerwaarts gerichte microfoon array's die spraak kunnen herkennen. Tot slot is er ook nog een aanstuurbare motor die geluidloos de Kinect op de juiste positie kan plaatsen.



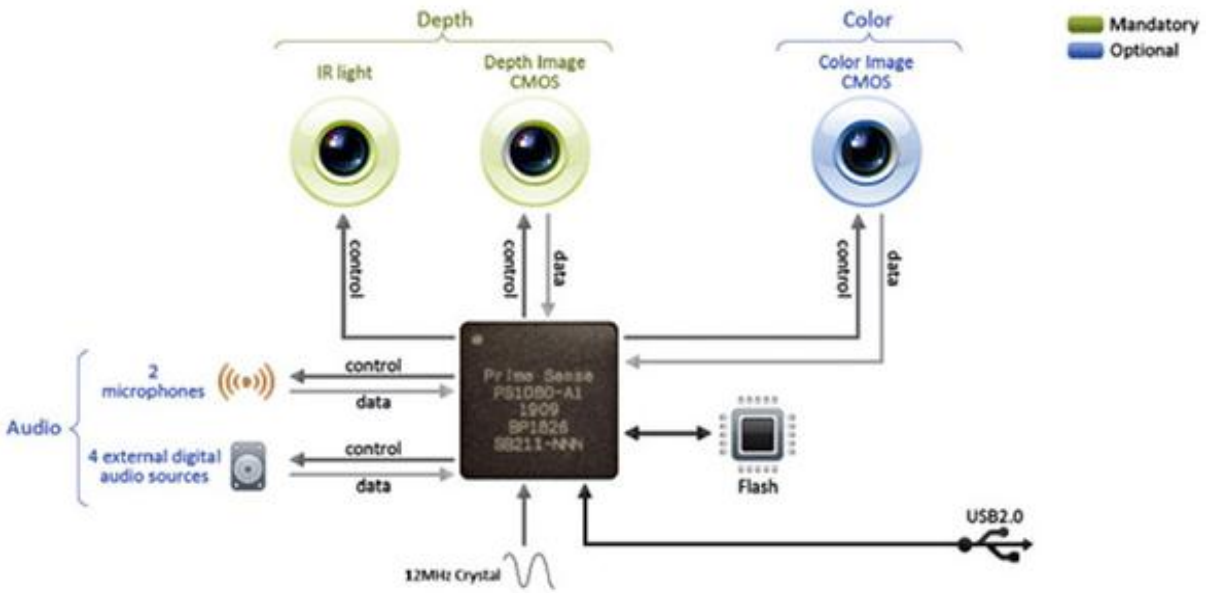
Figuur 2: Hardware onderdelen

Verder zijn deze slimme camera's en microfoontjes zeer compact verpakt, zoals te zien op de volgende afbeelding, dit alles met een aantrekkelijk buitenkant.

[1]Een actieve pixel sensor die een geïntegreerd systeem van pixel sensoren bevat.

Figuur 3: Hardware van de Kinect

Het dieptezicht wordt (3D) geregeld door het IR- raster en de dieptecamera. Verder ziet men ook nog de RGB-camera die een normaal beeld vormt in kleuren. Tevens wordt audio ook doorgestuurd naar de primesense IC. Deze IC zorgt ervoor dat er goede beelden en audio verwerkt worden en kunnen worden opgeslagen in het flashgeheugen. Daarnaast is het mogelijk om deze data door te sturen via USB naar een PC of XB0X 360. Op de onderstaande tekening kan je zien hoe de camera's en dergelijke verbonden zijn met hun geheugen en interface.



Figuur 4: Interface van de camera's

4.3 Diepte zicht

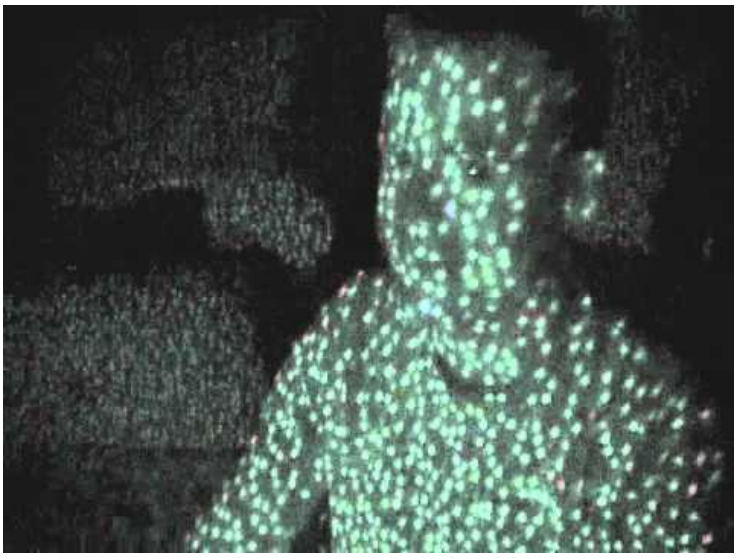
Zoals eerder te zien was, is er één slimme 3D- dieptesensor en één infrarood projector die gemoduleerde infrarood stralen uitzend, die het dieptezicht mogelijk maken. Een infraroodcamera van 1,3 megapixels de projector, die samen werken als een 3D- scanner. Om zo volgens het Time-Of-Flight-principe[2] de afstand van elke ontvangen pixel te meten.
Het principe is als volgt: door gemoduleerde infraroodstralen uit te sturen en dan de gereflecteerde stralen van objecten weer op te vangen kunnen onderzoekers door het tijdsverschil te meten een dieptewaarde bekomen. Met de ontvangen data worden verschillende objecten in lagen opgebouwd en deze worden op de z– as geplaatst. Op deze manier ontstaat er een diepte map met grijswaarden van het gebied zoals te zien op de volgende afbeelding.

Figuur 5: 3D – beeld

Bij een framerate van 60 frames per seconden kunnen er 3D– bewegingen van 1 cm geregistreerd worden. Let wel op dat deze framerate voorlopig gelimiteerd is op 30 frames per seconde op het gebied van software. Deze camera's zijn ontworpen door het Israëlische zusterbedrijf van Microsoft, genaamd PrimeSense.
De densiteit en hoeveelheid infraroodstralen die de Kinect uitzendt zijn eveneens revolutionair, de volgende afbeelding geeft hier een idee van. Deze is getrokken in het donker met een nachtcamera.

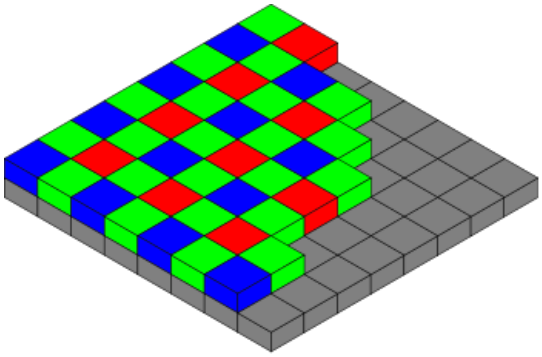
---

[2] Een Time-of-flightcamera of ToF-camera is een camerasysteem dat naast lengte en breedte ook diepte in beeld waarneemt, welke wordt berekend door middel van de reistijd die licht gebruikt. Zo vormt de camera geen tweedimensionaal beeld maar een driedimensionaal beeld.

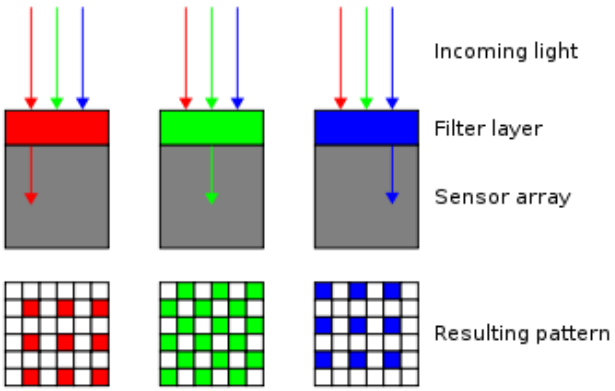


Figuur 6: Infrarood veld

Eén van de grote voordelen van het gebruik van gemoduleerde infraroodstralen is dat deze niet beïnvloed worden door licht, dit is vaak het probleem bij hogesnelheidscamera's. Het is dus mogelijk om een dieptezicht te krijgen in het donker. Hoewel licht geen invloed heeft op de dieptesensoren is dit natuurlijk niet het geval bij de RGB- camera. De sluitertijd van deze camera varieert wel degelijk bij overgang van dag naar nacht. Zo is de sluitertijd overdag 100 milliseconden en ligt deze gedurende de nacht veel hoger. Ook is voorlopig het aanpassen van deze sluitertijd niet mogelijk, deze wordt softwarematig bepaald.
De RGB- camera werkt verder volgens het principe van een Bayer- kleurenfilter. Een Bayerfilter is een regelmatig vierkant raster van kleurenfilters om zo de kleuren rood, groen en blauw van mekaar te onderscheiden. Deze wordt zoals verwacht vooral gebruikt bij digitale beeldverwerking. De volgende afbeeldingen illustreren het principe van de Bayerfilter.



Figuur 7: RGB Bayerfilter



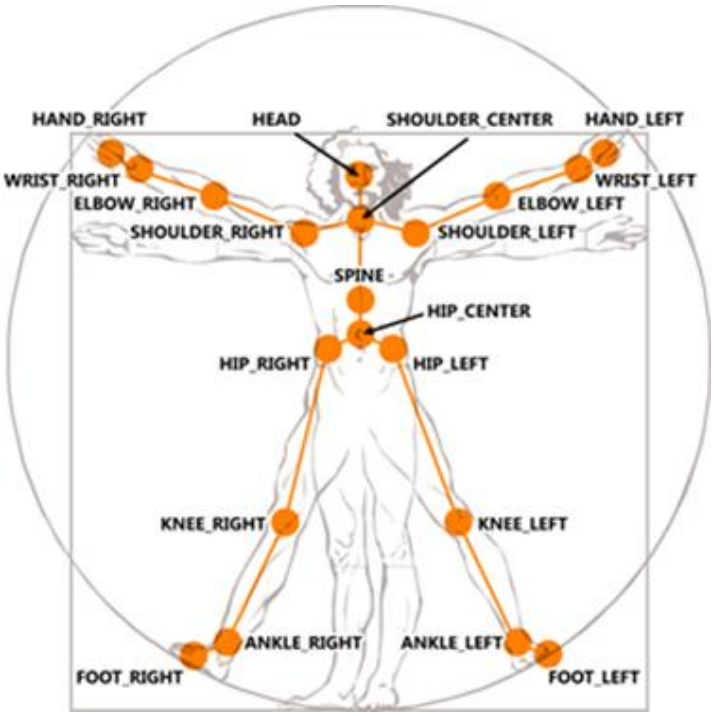
Figuur 8: onderscheiden van kleuren

De SDK 1.0 Beta 2, heeft de volgende technische specificaties op het gebied van de camera's. De RGB en dieptecamera's een resolutie hebben van 1280 x 1024 pixels, wat uitzonderlijk goed is voor die prijs, is de resolutie softwarematig gelimiteerd tot 640 x 480 pixels. Ook is de framerate softwarematig gelimiteerd tot 30Hz van deze camera's voor deze

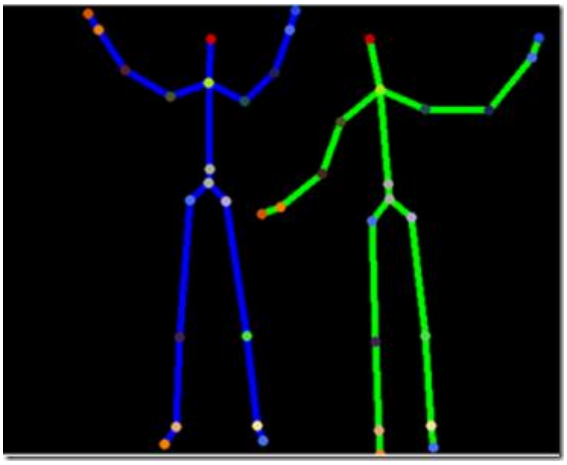
engine, ook al zijn ze op het gebied van hardware ontwikkeld met een framerate van 60Hz. Dit is hoogstwaarschijnlijk te wijten aan het feit dat de Kinect ontwikkeld is voor een game console, deze zijn op grafisch gebied veel sterker dan bijvoorbeeld een computer.
Er zijn nog een aantal andere hardwarelimieten waar rekening mee moet worden gehouden. Zo zullen de afstandsmetingen maar correct zijn vanaf een afstand van 1,2 meter en tot een afstand van 3,5 meter. In het algemeen heeft de Kinect een werkoppervlak van 6m². De sensor heeft een gezichtsveld van horizontaal 57° en verticaal 43°. Dankzij het motortje in de voet van de Kinect kan de Kinect nog van -27° tot 27° gekanteld worden. Dit geeft een pixel resolutie van 1,3 mm/pixel.

4.4 Skeletopbouw

De Kinect heeft een engine die het mogelijk maakt om het skelet van een speler te simuleren. Dankzij een inventieve bitberekening kan de Kinect tot 20 joints van het menselijke lichaam detecteren en volgen. De SDK geeft de exacte 3D- coördinaten ter beschikking, dit geeft veel nieuwe mogelijkheden.
De Kinect heeft de eigenschap om tegelijkertijd zes spelers te volgen, hiervan zijn er twee spelers die actief gevolgd kunnen worden, dus waarbij bewegingsanalyse mogelijk is. Dus de huidige SDK kan tegelijkertijd 40 “joints” of gewrichten met hun coördinaten ter beschikking geven van twee spelers. De volgende afbeelding geeft aan welke joints van het lichaam ter beschikking zijn.



Figuur 9: Gewrichtspunten van het skelet



Figuur 10: Skelet poppetjes

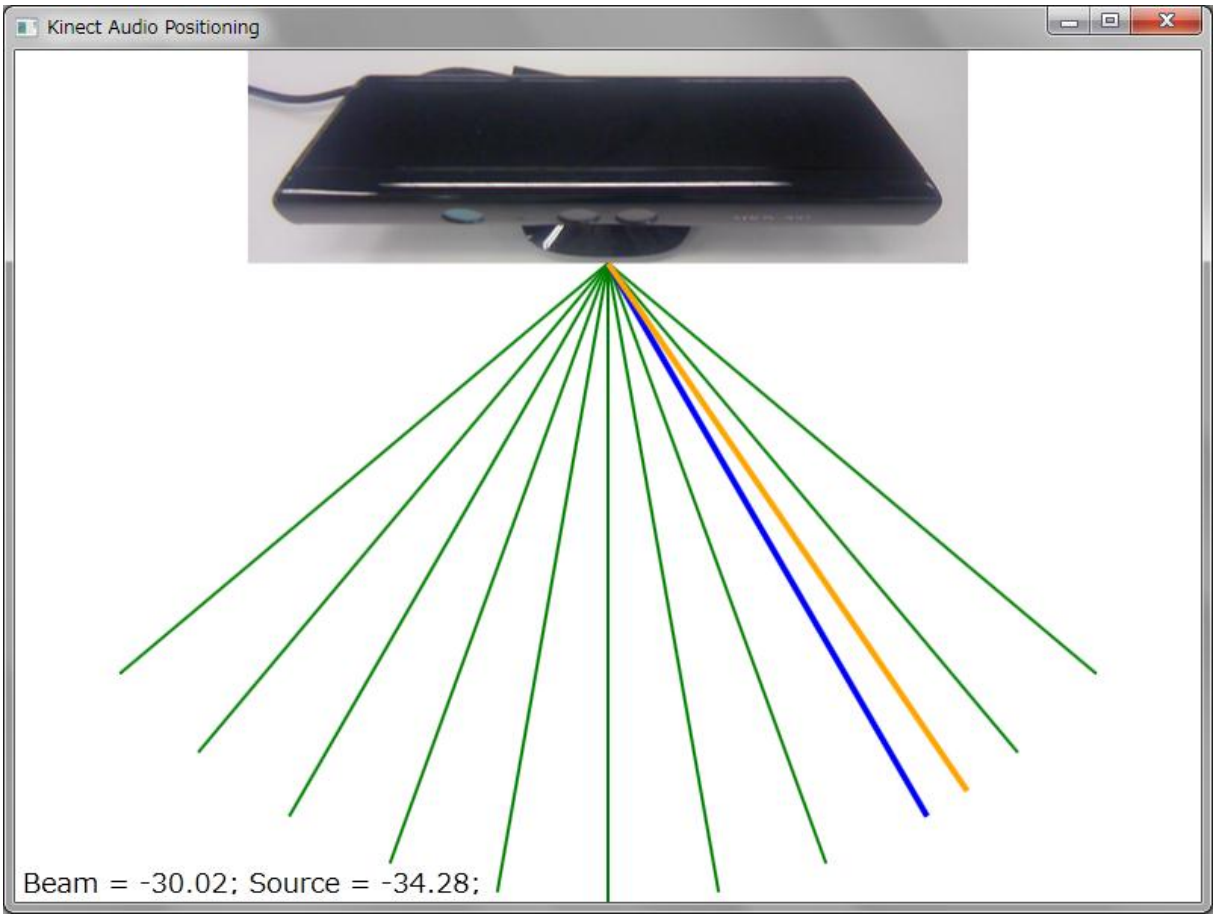
Hoewel we alleen de coördinaten ter beschikking krijgen, zal via een simpele DrawLine-opdracht, in een .Net – omgeving, tussen de coördinaten een skeletbeeld bekomen worden. Wanneer er dan twee spelers gevolgd worden, zullen de skeletten er zoals op de tekening uitzien op ons computerscherm

4.5 Spraakherkenning en audio

De Kinect is ook heel geavanceerd als het aankomt op audiotoepassingen en spraakherkenning. Tot nu werden al twee manieren besproken om spelers te lokaliseren: via een dieptebeeld en via joint- coördinaten. Een derde manier is via de stem iemand lokaliseren. Zoals eerder vermeld bevat de Kinect vier microfoonarray 's. De vier kanalen werken samen als één microfoonarray die elk 16 bit audio kunnen verwerken met een sample rate van 16 kHz.

Er worden niet zomaar vier microfoontjes gebruikt. De microfoontjes worden zo gepositioneerd dat de Kinect enkel audio accepteert onder bepaalde hoeken. Er ontstaan als het ware audio beams waarvan er telkens één wordt gebruikt om zo de rest van het omgevingsgeluid te neutraliseren. In het totaal worden er 11 audio beams gecreëerd. Van

-50° tot 50° en tussen elke beam een hoek van 10°. De volgende tekening geeft een idee hoe we dit kunnen visualiseren.



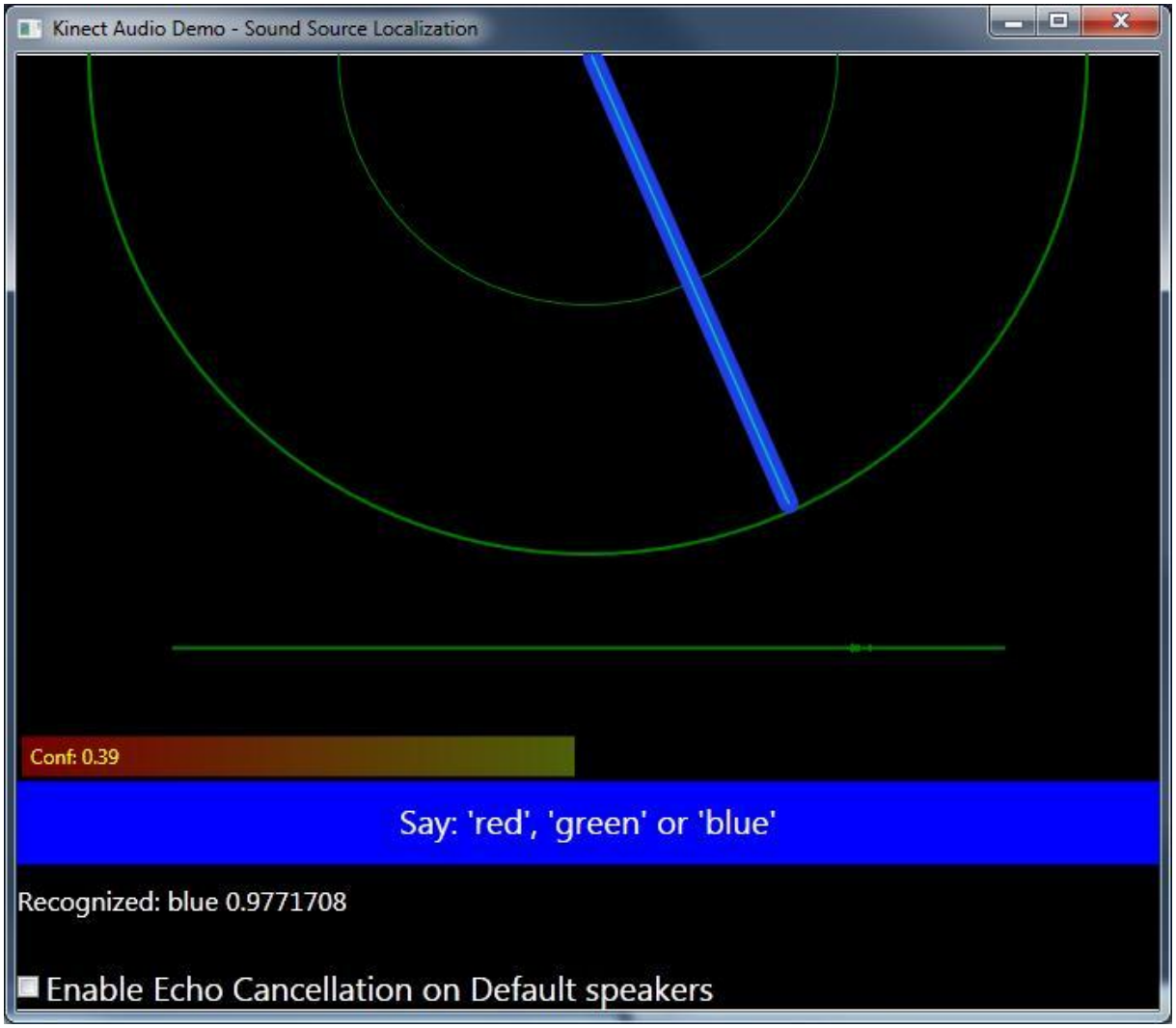
Figuur 11: Audio tracking

De groene stralen geven aan welke audio beams er ter beschikking zijn. De Kinect bepaalt waar het stemgeluid of de “source” zich bevindt, in dit geval aangeduid met de oranje straal. De Kinect bepaalt als volgt welke van zijn beschikbare beams het meest dichtbij de source ligt, dit wordt aangeduid met de blauwe straal. Op deze manier accepteert de Kinect alleen het geluid dat binnenkomt op zijn gekozen beam, dit noemt “acoustic source localization[3]”. Het achtergrondgeluid wordt nu verwaarloosd en kan dus niet interfereren met het gewenste geluid, dit noemt men “active noise control”.

Dankzij al deze audio-eigenschappen is het mogelijk voor Xbox 360- gebruikers om met elkaar “headset free” te communiceren. Ook voor spraakherkenning heeft dit veel voordelen, de makers van de Kinect wilden er zo zeker mogelijk van zijn dat de Kinect onze commando’s zo effectief mogelijk begrijpt en accepteert.

De SDK v1.0 Beta 2 heeft daarom ook een speech en speech recognition engine beschikbaar. De speech engine wordt gebruikt om de gewenste commando’s die de gebruiker wil gebruiken te declareren. Wanneer de commando’s ook effectief worden gezegd zal de speech recognition engine deze vergelijken met de gedeclareerde commando’s en een zekerheids- of “confidence” level aangeven. De volgende applicatie, vrijgegeven door Microsoft zelf, geeft een beeld van de beschikbare audiotoepassingen.

[3] Door gebruik van een microfoon bepalen waar geluid vandaan komt aan de hand van de geluidsdruk.



Figuur 12: spraak duidelijkheidsniveau herkenner

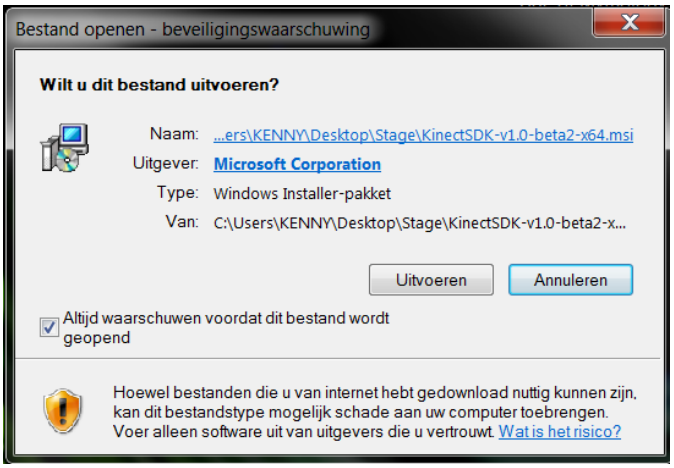
Dit geeft kort maar bondig een idee hoe de Kinect hardwarematig opgebouwd is en tot wat hij allemaal instaat is. De Kinect kan beschouwd worden als een enorm slimme camera, voor een relatief lage prijs. Vervolgens wordt er een handleiding geschreven die zal weergeven hoe men via C# - taal met de Kinect kan communiceren en zijn eigenschappen in een Windows-omgeving kan gebruiken.

5) Handleiding

We hebben een beknopte handleiding geschreven om een correcte opstelling te maken met de Kinect. Ook softwarematig worden er programma's geïnstalleerd om correct tussen computer en Kinect te communiceren.

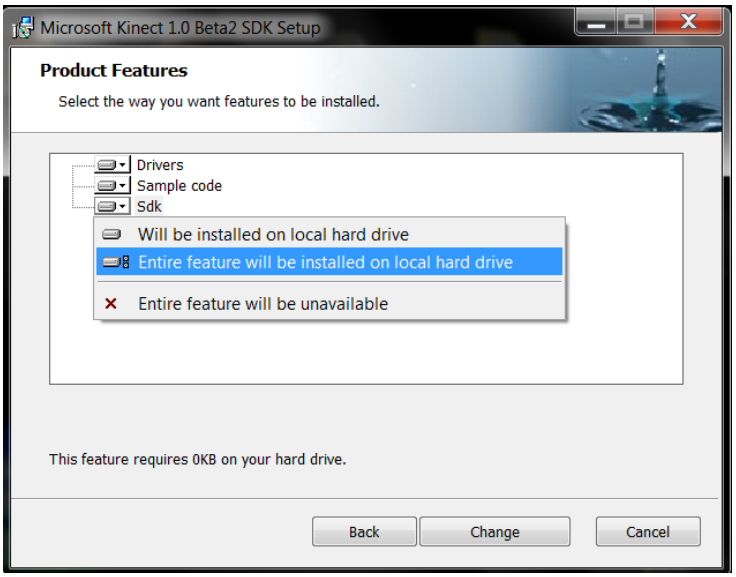
5.1 Microsoft Kinect SDK- handleiding

De Microsoft SDK Beta 2 is makkelijk om te gebruiken en te installeren. Op onze cd vind je de SDK manager "KinectSDK-v1.0-beta2-x64". Deze installeer je simpelweg door het bestand uit te voeren en vervolgens "next" te klikken bij de setup wizard. Let er wel op dat de SDK beperkt is en enkel op Windows 7 draait.



Figuur 13: software van de SDK uitvoeren

Vervolgens kan je de onderdelen selecteren om te installeren, Je installeert Drivers, voorbeeldcode en SDK volledig en gaat vervolgens verder om de setup af te sluiten. Nu is de SDK manager geïnstalleerd en klaar voor gebruik. Klik tot slot op Finish om de SDK setup af te sluiten.

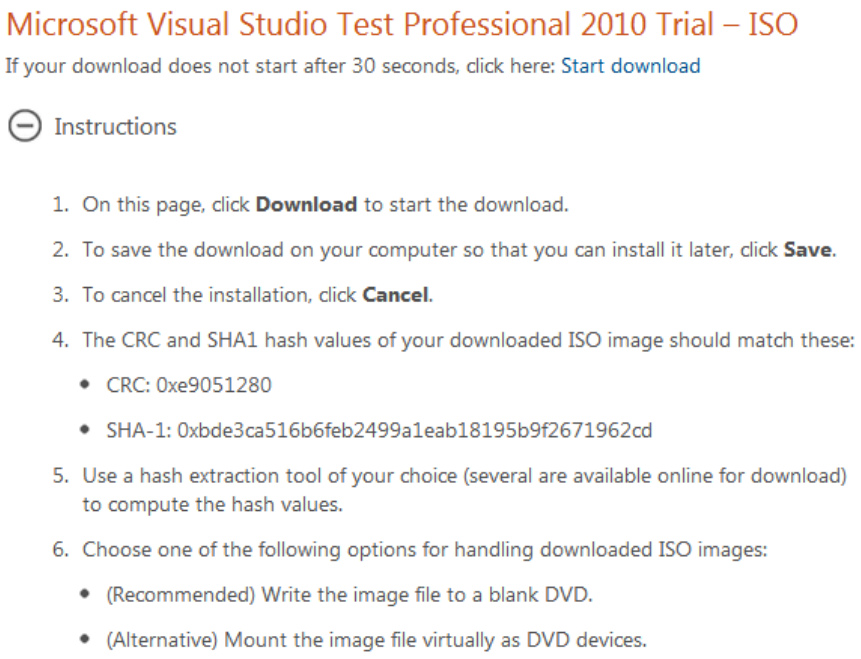


Figuur 14: Onderdelen installeren

5.2 Microsoft Visual Studio

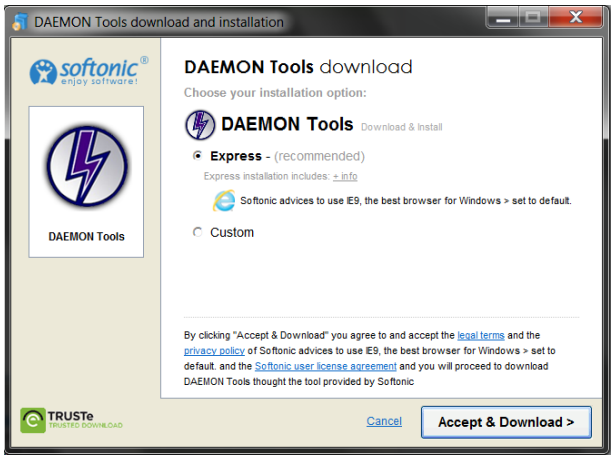
Allereerst heb je Visual studio nodig om de nodige programmacode te kunnen openen of zelf te schrijven. Op de volgende site kan je de software gratis downloaden en vervolgens installeren.
http://www.microsoft.com/visualstudio/en-us/products/2010-editions/express

Als bovenstaande link niet werkt heb je hier nog een back-up link.
http://www.microsoft.com/en-us/download/details.aspx?id=24988.
Je Download de Visual Studio 2010 Professional version trial. Volg de uitgelegde instructies om het programma te installeren.



Figuur 15: Instructies om Visual Studio te installeren

Als je problemen hebt bij het installeren van Visual Studio kan je ook Daemon tools downloaden en deze ISO hier virtueel op runnen. Hieronder de link waar je Daemon tools kan downloaden
http://daemon-tools.en.softonic.com/

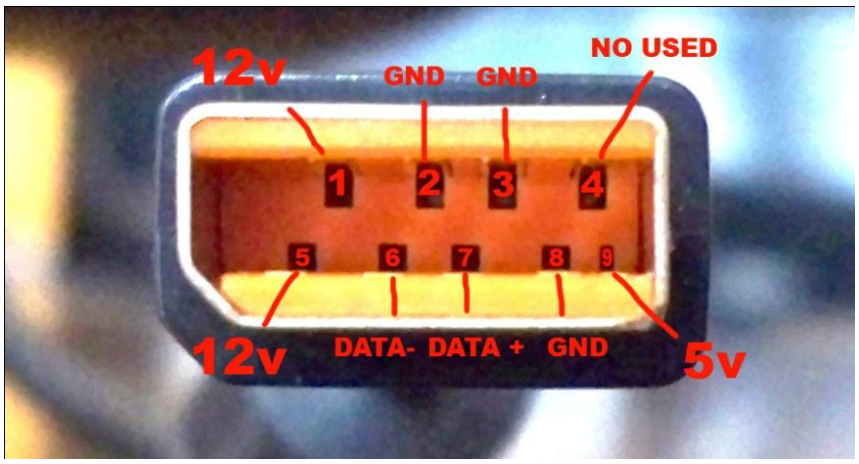


Figuur 16: DEAMON Tools installeren

5.3 De Kinect aansluiten

Als je eenmaal Visual studio hebt geïnstalleerd kan je de Kinect aansluiten via een connector die op een USB 2.0 connector lijkt. Deze is ontworpen om aangesloten te worden op de nieuwe black Xbox 360, maar de oudere modellen gebruiken een adapter die een converteerstuk heeft naar een gewone USB 2.0 connector.

Figuur 17: Kinect connector



Figuur 18: Kinect connector aansluitingen

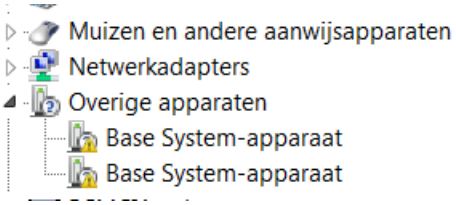
Pin	Description
1	+12V
2	Data -
3	Data +
4	Ground
5	+5V
6	+12V
7	Gnd
8	Gnd
9	N/C

Figuur 19: Pin Lay-out

Als de SDK geïnstalleerd is kan je bij apparaatbeheer of device manager kijken of de drivers automatisch geïnstalleerd zijn. Onder Microsoft Kinect zie je drie drivers die geïnstalleerd zijn, als je deze niet ziet kan het zijn dat de drivers bij de overige apparaten zijn toegevoegd als

Base System-apparaat. Deze drivers staan misschien wel op een andere plaats maar deze zullen even goed werken.

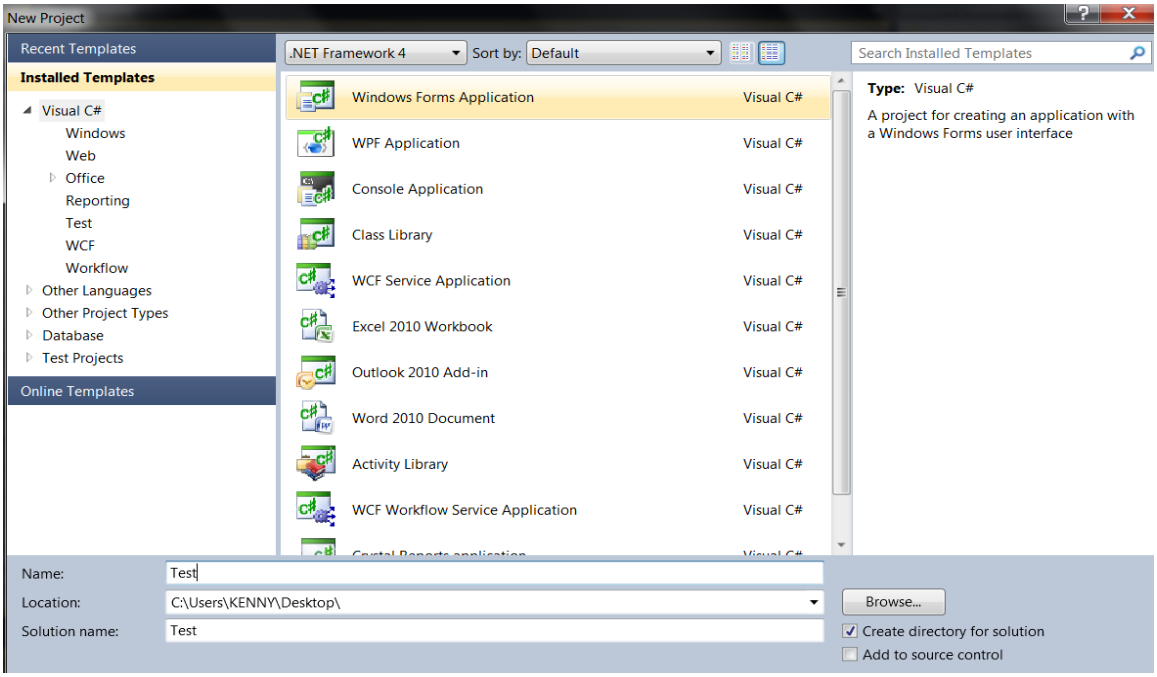
Figuur 19: Drivers van de Kinect



Figuur 20: Overige drivers Kinect

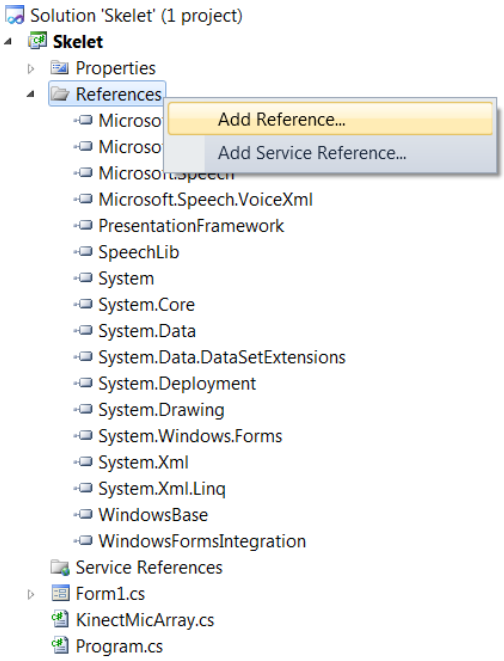
6) Software

Als je nu een c# project aanmaakt doormiddel van naar “File – New – Project” te kiezen, krijg je onderstaande tekening op je scherm. Vervolgens kies je Visual C# en dan een Windows Forms Application, verder moet je project onderaan ook een naam geven. In dit geval noemen we het programma Test.



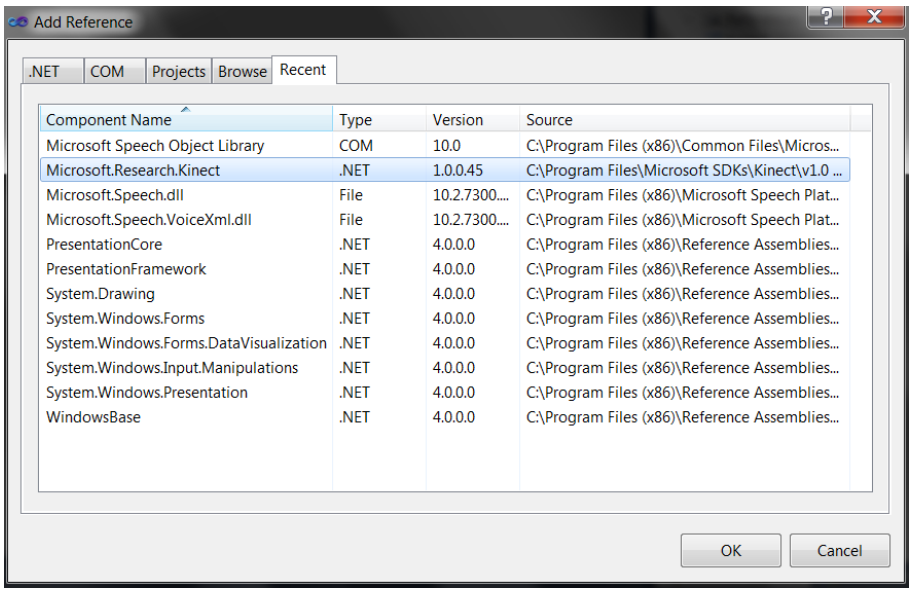
Figuur 21: Visual studio C# programma aanmaken

Als je dit hebt aangemaakt zal je de Kinect moeten toekennen aan het C# programma en in de rechterkolom kijken bij references. Klik vervolgens rechts op References e dan op Add Reference…



Figuur 22: References toevoegen aan je programma

Vervolgens ga je verschillende DLL files kunnen toevoegen aan je programma. Afhankelijk van je toepassing die je wil gaan schrijven heb je bepaalde DLL references nodig. De Microsoft. Research.Kinect DLL heb je altijd nodig in welk programma je ook mag schrijven. Deze DLL is nodig om de Kinect te herkennen. Verder zie je nog andere DLL files om Spraak toe te voegen of bepaalde tekeningen(Skelet) te tekenen. Later gaan we verder op deze DLL files in alsook hun specifieke functie.

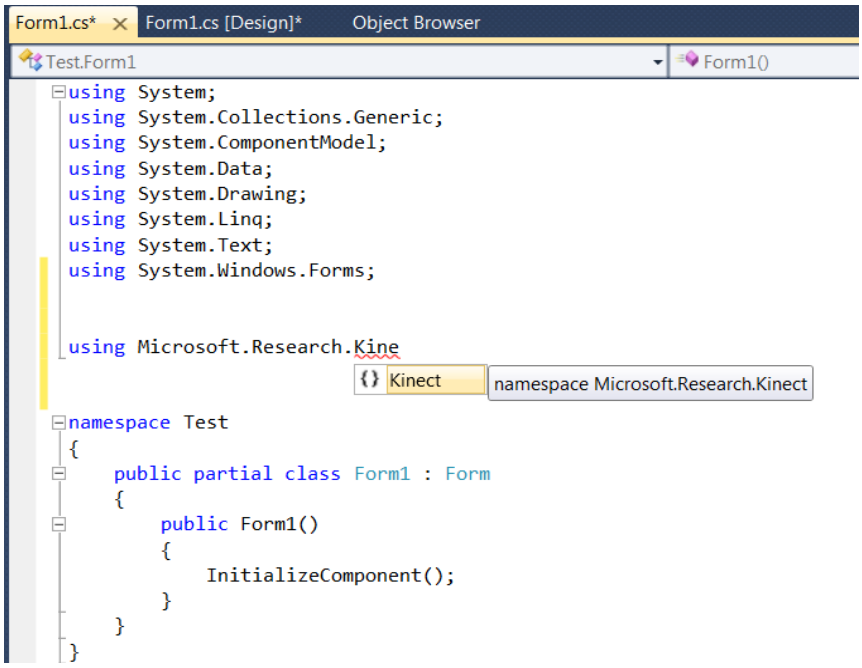


Figuur 23: Mogelijke references die je kan toevoegen

Als men nu een programma wil schrijven met de Kinect ga je, nadat je de DLL file hebt toegevoegd, de bibliotheek ervan toevoegen. `using Microsoft.Research.Kinect`

Verder kan men ook de volgende systeem bestanden toevoegen om audio of een camera (nui) te gebruiken.

```
using Microsoft.Research.Kinect.Audio;
using Microsoft.Research.Kinect.Nui;
```



Figuur 24: Bibliotheken toevoegen van de references

6.1 Voorbeeldprogramma met camera

Er word een voorbeeldprogramma geschreven om zo gemakkelijk de code uit te leggen. Elk Kinect programma heeft grotendeels hetzelfde begin qua programmeercode. Je moet een object aanmaken dat Runtime heet, deze weet of er één of meerdere Kinects zijn aangesloten op je PC.

```
Runtime nui = Runtime.Kinects[0];
```

De videocamera word gedeclareerd met de volgende Code. “UsColor” geeft aan dat hier alleen de RGB – camera word aangesproken.

```
nui.Initialize(RuntimeOptions.UseColor);
```

Vervolgens moet de stroom van beelden geopend worden zodat data kan gegenereerd worden.

```
nui.VideoStream.Open(ImageStreamType.Video,2,ImageResolution.Resolution640x480,ImageType
.Color);
```

De numerieke 2 wijst er op dat twee frames worden gebufferd voordat ze wegvallen. Omdat een PC snel genoeg is om dit te verwerken moet deze waarde niet hoger zijn. De beeldresolutie word ook bepaald in dit stukje code.

Nu dat de frames binnenkomen, moeten deze natuurlijk ook verwerkt worden, dit word in de volgende EventHandler gedaan.

```
nui.VideoFrameReady += new EventHandler<ImageFrameReadyEventArgs>(FrameReady);
```

In deze EventHandler word het duidelijk dat de data van elke frame word verwerkt tot een bitmap van 32 bits. De bits van elke frame kunnen aangesproken worden met de volgende code.

```
ImageFrame.Image.Bits
```

De ImageFrame.Image property heeft ook bruikbare data zoals het frame nummer en tijd, maar deze property wordt grotendeels gebruikt als image property. De data is in de vorm van een byte array samen met de image data verpakt in een bepaald formaat. Voor video is dit 4 bytes per pixel in RGBA formaat(32bit). De pixel data is opgeslagen in de array in een volgorde en met de volgende code gaan we deze wegschrijven

```
Size = height*width*bytesperpixel
```

De data komt uit onze frames, deze word door onze bitmap methode gestuurd via de volgende EventHandler.

```
void FrameReady(object sender, ImageFrameReadyEventArgs e)
                    {
                    PlanarImage Image = e.ImageFrame.Image;
```

Er word dus een frame van de vorm “PlanarImage” door de bitmap methode gestuurd. De bitmap methode doet simpelweg een berekening om van het binnenkomende beeld een gewenst bitmap formaat te maken. In deze berekening is de het pixelformaat volledig naar keuze. De bitmap methode stuurt dus de afgewerkte bitmap terug naar de EventHandler.

```
Bitmap PImageToBitmap(PlanarImage PImage)
                {
Bitmap bmap = new Bitmap(PImage.Width, PImage.Height, PixelFormat.Format32bppRgb);
BitmapData        bmapdata        =        bmap.LockBits(new        Rectangle(0,        0,
PImage.Width,PImage.Height),ImageLockMode.WriteOnly,bmap.PixelFormat);
IntPtr ptr = bmapdata.Scan0;
Marshal.Copy(PImage.Bits,0,ptr,PImage.Width * PImage.BytesPerPixel * PImage.Height);
map.UnlockBits(bmapdata);
return bmap;
```

```
}
```

Wanneer men nu in het FrameReady Event de bitmap in een picturebox plaatst is de volgende output zichtbaar.

```
pictureBox1.Image = bmap;
```

Figuur 25: Camera beeld

Wanneer dit 30 maal per seconde word gedaan, ziet men dus een real-time camerabeeld.
Er is ook code nodig om onze videostream te sluiten, anders sluit het programma zich fout af en zal er een foutmelding verschijnen.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
 nui.Uninitialize();
}
```

6.2 Depthview

Hieronder wordt het dieptezicht uitlegt. Dit lijkt qua code sterk op die van het gewone camerabeeld. Bijvoorbeeld kan de Kinect je vertellen hoe ver elke pixel in het beeld staat, verder nog kan je elke pixel labelen aan een persoon zodat hij deze personen afzonderlijk kan herkennen(dit noemt men ook wel indexeren).
De eerste stappen zijn hetzelfde zoals bij het normale camerabeeld.

```
Runtime nui = Runtime.Kinects[0];

nui.Initialize(RuntimeOptions.UseDepth);
```

Een stroom van camerabeelden is nu niet gewenst, maar wel een stroom van dieptebeelden.

```
nui.DepthStream.Open(ImageStreamType.Depth, 2, ImageResolution.Resolution320x240,
ImageType.Depth);
```

Alle binnenkomende diepteframes moeten weer netjes verwerkt worden, dit zal weer gebeuren in een EventHandler.

```
nui.DepthFrameReady += new EventHandler<ImageFrameReadyEventArgs>(DepthFrameReady);

void DepthFrameReady(object sender, ImageFrameReadyEventArgs e)
{
PlanarImage PImage = e.ImageFrame.Image;
pictureBox1.Image = DepthToBitmap(PImage);
}
```

De binnenkomende beelden moeten natuurlijk weer in het gewenste bitmap formaat staan, daarom is de bitmap methode ook weer van toepassing.

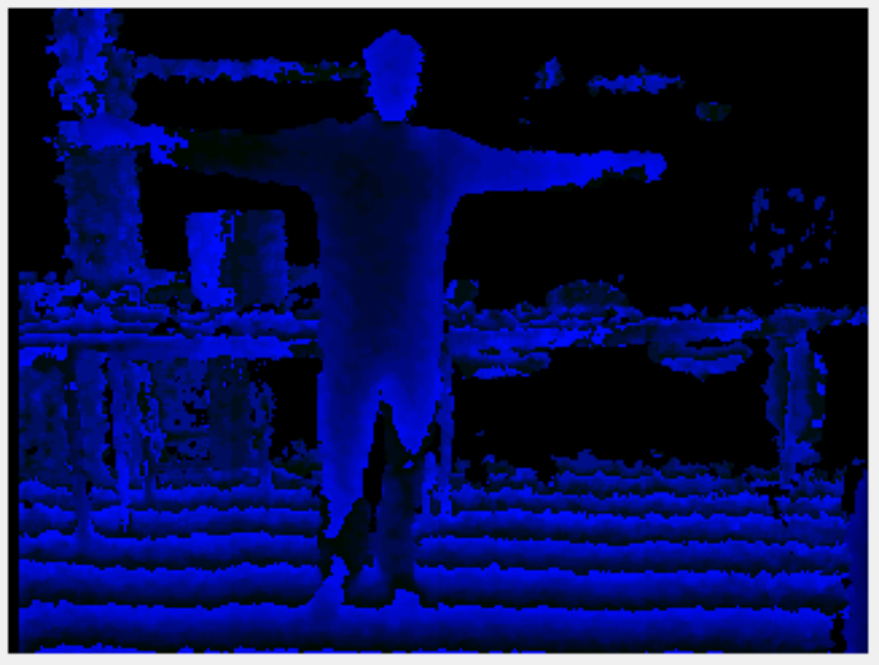
De bitmapcode is wel niet helemaal dezelfde als deze van het normale camerabeeld. Om het DepthFrame weer te geven gebruiken we Format16bppRgb555. Dit behandelt 16 bit waardes zodat voor elke basiskleur(R,G en B) elk 5 bits gebruikt kunnen worden. De laatste bit die overschiet wordt gewoon weggelaten omdat deze nutteloos is in dit geval.

```
Bitmap DepthToBitmap(PlanarImage PImage)
{
Bitmap bmap = new Bitmap(PImage.Width, PImage.Height, PixelFormat.Format16bppRgb555);
BitmapData bmapdata = bmap.LockBits(new Rectangle(0, 0, PImage.Width, PImage.Height),
ImageLockMode.WriteOnly, bmap.PixelFormat);
IntPtr ptr = bmapdata.Scan0;
Marshal.Copy(PImage.Bits, 0, ptr, PImage.Width * PImage.BytesPerPixel * PImage.Height);
bmap.UnlockBits(bmapdata);
return bmap;
}
```

De Camera moet bij het afsluiten ook afgesloten worden om onregelmatigheden te vermijden.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
 nui.Uninitialize();
}
```

De volgende output is zichtbaar.



Figuur 26: Depthview

6.3 Persoon Indexing

Net als in elk programma gaan we de Kinect opstarten met het Nui Runtime object.

```
nui = Runtime.Kinects[0];
nui.DepthStream.Open(ImageStreamType.Depth, 2, ImageResolution.Resolution320x240,
ImageType.DepthAndPlayerIndex);
nui.DepthFrameReady += new
EventHandler<ImageFrameReadyEventArgs>(nui_DepthFrameReady);
```

Omdat bij het indexeren verschillende mensen van mekaar onderscheiden moeten worden, is het belangrijk om de diepte en speler data in arrays te plaatsen. Dit word gedaan in de volgende EventHandler.

```
void nui_DepthFrameReady(object sender, ImageFrameReadyEventArgs e)
   {
   PlanarImage Image = e.ImageFrame.Image;
   int[] playercoded = new int[ Image.Width * Image.Height];
   int[] depth = new int[Image.Width * Image.Height];
   int[] player = new int[Image.Width * Image.Height];
```

```
    for (int i = 0; i < depth.Length; i++)
    {
    depth[i] = (Image.Bits[i * 2 + 1] << 5) | (Image.Bits[i * 2] >> 3);
```

Verder word in de volgende code elke speler die nieuw aan de speler array word toegevoegd, weergegeven met een willekeurig gekozen kleur.

```
player[i] = Image.Bits[i * 2] & 0x07;
                playercoded[i] = 0;
                if ((player[i] & 0x01) == 0x01)
                    playercoded[i] |= 0xFF0000;
                if ((player[i] & 0x02) == 0x02)
                    playercoded[i] |= 0x00FF00;
                if ((player[i] & 0x04) == 0x04)
                    playercoded[i] |= 0x0000FF;
            }
pictureBox1.Image = IntToBitmap(depth, Image.Width, Image.Height);
}
```

De bitmap methode is ook weer van toepassing.

```
Bitmap IntToBitmap(int[] array, int w, int h)
    {
    Bitmap bmap = new Bitmap( w, h,PixelFormat.Format32bppRgb);
    BitmapData bmapdata = bmap.LockBits(new Rectangle(0, 0,
        w,h),ImageLockMode.WriteOnly,bmap.PixelFormat);
    IntPtr ptr = bmapdata.Scan0;
    Marshal.Copy(array,0, ptr,array.Length);
    bmap.UnlockBits(bmapdata);
    return bmap;
    }
```

De volgende voorbeeld output is zichtbaar.

Bit 1 → R = 0xFF (helder rood)

Bit 2 → G = 0x00 (helder Groen)

Bit 3 → B = 0x00 (helder Blauw)

Deze kleuren gaan zich mengen en een rode kleur voortbrengen als er een persoon nu in beeld zal komen.

Figuur 27: Het indexeren van een persoon

6.4 Skelet Programma

Zoals gewent zijn de eerste stappen weer dezelfde als in de vorige situaties.

```
using Microsoft.Research.Kinect.Nui;

public Form1()
      {
       InitializeComponent();
       nui = Runtime.Kinects[0];
       nui.Initialize(RuntimeOptions.UseSkeletalTracking |
       RuntimeOptions.UseDepthAndPlayerIndex | RuntimeOptions.UseColor);
       nui.VideoStream.Open(ImageStreamType.Video, 2,
       ImageResolution.Resolution640x480, ImageType.Color);
       nui.VideoFrameReady += new
       EventHandler<ImageFrameReadyEventArgs>(FrameReady);
       nui.SkeletonFrameReady += nui_SkeletonFrameReady;
       }

void FrameReady(object sender, ImageFrameReadyEventArgs e)
        {
            videoImage = e.ImageFrame.Image;
        }
```

```
void SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
         if (videoimage.Bits == null) return;
        }
```

De binnenkomende planarimage zal ook weer geconverteerd moeten worden naar het gewenste bitmap formaat.

```
Bitmap PImageToBitmap(PlanarImage PImage)
{
Bitmap bmap = new Bitmap(PImage.Width, PImage.Height, PixelFormat.Format32bppRgb);
BitmapData bmapdata = bmap.LockBits(new Rectangle(0, 0, PImage.Width, PImage.Height),
ImageLockMode.WriteOnly, bmap.PixelFormat);
IntPtr ptr = bmapdata.Scan0;
Marshal.Copy(PImage.Bits, 0, ptr, PImage.Width * PImage.BytesPerPixel * Image.Height);
bmap.UnlockBits(bmapdata);
return bmap;
}
```

De volgende bibliotheken worden toegevoegd.

```
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
```

Om er voor te zorgen dat het programma zonder fouten sluit zal je een FormClosing EventHandler moeten schrijven.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            nui.Uninitialize();
        }
```

Nu gaan we het Skelet opbouwen uit een verzameling van verschillende “Joints” die overeenkomen met de werkelijke gewrichten van je lichaam. Onderstaande code beschrijft hoe we de Coördinaten gaan bepalen van 20 “joints” in het lichaam.

```
void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
            frame = e.SkeletonFrame;

            IEnumerable<SkeletonData> skeletonlist = (from s in frame.Skeletons
            where s.TrackingState == SkeletonTrackingState.Tracked select s);

            SkeletonData skeleton = skeletonlist.FirstOrDefault();
```

Wanneer de Joints beschikbaar zijn in het huidige frame zullen deze doorlopen worden met de parameter “s” en in de huidige skeletonlist geplaatst worden. De huidige skeletonlist krijgt zijn eigen parameternaam “Skeleton”. Ook kan je hier declareren welke voorrang je huidige skelet heeft. Hier heeft “Skeleton” de voorrang van het eerste skelet, dus de eerste persoon die in het beeld komt en getraceerd wordt blijft “Skeleton” behouden. De huidige SDK maakt het mogelijk om 2 spelers te traceren. Wanneer de eerste 20 punten zijn gegeven aan de parameternaam “Skeleton”, worden de volgende 20 punten van een speler 2 aan een andere parameternaam “Skeleton2” toegekend. De voorrang van dit skelet kan ook meegegeven worden.

```
SkeletonData skeleton2 = skeletonlist.Last();
```

We hebben nu onze twee getraceerde skeletten ter beschikking. Hoe we nu effectief elke joint apart aanspreken met zijn huidige coördinaten word gedaan met de volgende code.

```
Double HipX = Convert.ToDouble(skeleton.Joints[JointID.HipCenter].Position.X);

Double HipY = Convert.ToDouble(skeleton.Joints[JointID.HipCenter].Position.Y);

Double HipZ = Convert.ToDouble(skeleton.Joints[JointID.HipCenter].Position.Z);
```

In deze code is joint “HipCenter” aangesproken van skelet 1 (“Skeleton”). Het is mogelijk om zo elk joint aan te spreken door dit simpel aan te passen in de code:

```
skeleton.Joints[JointID. “ gewenste joint ”]
```

Om het skelet zo makkelijk mogelijk te tekenen hebben we ervoor gekozen om alle x en y coördinaten in een queue of rij te plaatsen.

```
private Queue<Point> jointPoints;
private Queue<Point> jointPoints2;
```

In onze EventHandler vullen we de rij aan per frame, met behulp van een For – lus.

```
for (int PtIndx = 0; PtIndx < joints.Count; PtIndx++)
                {
                    float x, y, x2, y2;
                    Point current;
                    Point current2;

  // Translate Joint from Kinect space to form space
nui.SkeletonEngine.SkeletonToDepthImage(joints[(JointID)PtIndx].Position, out x, out y);
nui.SkeletonEngine.SkeletonToDepthImage(joints2[(JointID)PtIndx].Position, out x2, out y2);
   // Use the client area of the form as the bounds for our drawing space
x = Math.Max(0, Math.Min(x * this.ClientRectangle.Width, this.ClientRectangle.Width));
y = Math.Max(0, Math.Min(y *this.ClientRectangle.Height,this.ClientRectangle.Height));
x2 = Math.Max(0, Math.Min(x2 *this.ClientRectangle.Width,this.ClientRectangle.Width));
y2 = Math.Max(0, Math.Min(y2 * this.ClientRectangle.Height,this.ClientRectangle.Height));
  // Keep only whole part of values
current = new Point((int)Math.Truncate(x), (int)Math.Truncate(y));
current2 = new Point((int)Math.Truncate(x2), (int)Math.Truncate(y2));
  // store translated point for painting
jointPoints.Enqueue(current);
jointPoints2.Enqueue(current2);
    }
}
```

Waar de skeletten worden getekend is geheel naar keuze. Wij hebben ervoor gekozen om dit in het formulier zelf te doen, maar dit kan eveneens in een PaintPanel. Het enigste wat nu nog moet gedaan worden is de rij aflopen en met een simpele DrawLine functie een lijn tekenen tussen de coördinaten die in de rij zijn opgeslagen.

```
Point p = jointPoints.Dequeue();
Point p2 = jointPoints2.Dequeue();
```

De enige moeilijkheid ligt erin dat de lijn moet getekend worden tussen het huidige punt p en het vorige punt in de rij. Daarom is het aan te raden om de punten in een array te plaatsen zodat de vorige punten ook nog ter beschikking zijn.

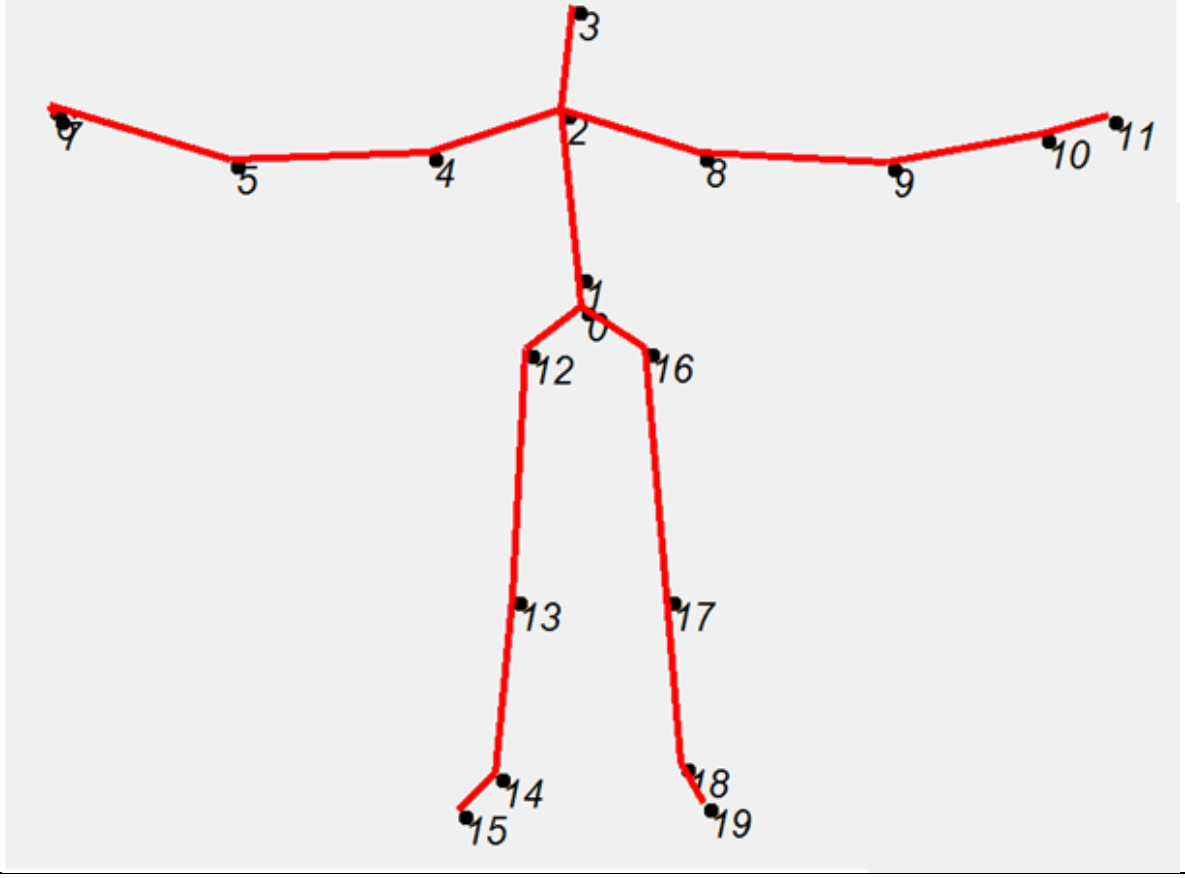
```
Point[] arraySkeleton = new Point[20];
Point[] arraySkeleton2 = new Point[20];

arraySkeleton [j] = new Point(p.X, p.Y);
arraySkeleton2[j] = new Point(p2.X, p2.Y);
j++;
```

Het is ook aan te raden om de arrays eerst volledig aan te vullen met 20 punten en daarna te tekenen. Om onregelmatigheden tegen te gaan.

```
while (jointPoints.Count > 0)
                {
                    if (j == 19)
                    {
                        g.DrawLine(pen, arrayx[0], arrayx[1]);
                        g.DrawLine(pen2, arrayx2[0], arrayx2[1]);

                        g.DrawLine(pen, arrayx[1], arrayx[2]);
                        g.DrawLine(pen2, arrayx2[1], arrayx2[2]);

                        g.DrawLine(pen, arrayx[2], arrayx[3]);
                        g.DrawLine(pen2, arrayx2[2], arrayx2[3]);

                        g.DrawLine(pen, arrayx[2], arrayx[4]);
                        g.DrawLine(pen2, arrayx2[2], arrayx2[4]);

                        g.DrawLine(pen, arrayx[4], arrayx[5]);
                        g.DrawLine(pen2, arrayx2[4], arrayx2[5]);

                        g.DrawLine(pen, arrayx[5], arrayx[6]);
                        g.DrawLine(pen2, arrayx2[5], arrayx2[6]);

                        g.DrawLine(pen, arrayx[6], arrayx[7]);
                        g.DrawLine(pen2, arrayx2[6], arrayx2[7]);

                        g.DrawLine(pen, arrayx[2], arrayx[8]);
                        g.DrawLine(pen2, arrayx2[2], arrayx2[8]);

                        g.DrawLine(pen, arrayx[8], arrayx[9]);
                        g.DrawLine(pen2, arrayx2[8], arrayx2[9]);

                        g.DrawLine(pen, arrayx[9], arrayx[10]);
                        g.DrawLine(pen2, arrayx2[9], arrayx2[10]);

                        g.DrawLine(pen, arrayx[10], arrayx[11]);
                        g.DrawLine(pen2, arrayx2[10], arrayx2[11]);

                        g.DrawLine(pen, arrayx[0], arrayx[12]);
                        g.DrawLine(pen2, arrayx2[0], arrayx2[12]);
```

```
                g.DrawLine(pen, arrayx[12], arrayx[13]);
                g.DrawLine(pen2, arrayx2[12], arrayx2[13]);

                g.DrawLine(pen, arrayx[13], arrayx[14]);
                g.DrawLine(pen2, arrayx2[13], arrayx2[14]);

                g.DrawLine(pen, arrayx[14], arrayx[15]);
                g.DrawLine(pen2, arrayx2[14], arrayx2[15]);

                g.DrawLine(pen, arrayx[0], arrayx[16]);
                g.DrawLine(pen2, arrayx2[0], arrayx2[16]);

                g.DrawLine(pen, arrayx[16], arrayx[17]);
                g.DrawLine(pen2, arrayx2[16], arrayx2[17]);

                g.DrawLine(pen, arrayx[17], arrayx[18]);
                g.DrawLine(pen2, arrayx2[17], arrayx2[18]);

                g.DrawLine(pen, arrayx[18], arrayx[19]);
                g.DrawLine(pen2, arrayx2[18], arrayx2[19]);
            }
        }
```

De output die rechtstreeks zichtbaar is op het formulier is als volgt.



Figuur 28: Skeletpoppetje met de gewrichtscoördinaten

Nu we het Skeletprogramma geschreven hebben, kunnen we overgaan tot spraak toepassingen.

6.5 Spraakherkenning

We hebben een programma dat op stemcommando een actie onderneemt, bijvoorbeeld die de Kinect omhoog of omlaag laat bewegen, een foto laat nemen en eventueel het formulier laat sluiten. De SDK v1.0 Beta 2 heeft twee engines die worden gebruikt voor spraak herkenning. Deze zullen in Visual Studio als reference moeten worden toegevoegd vooraleer we ze kunnen gebruiken.

Add reference → Microsoft.Speech.dll

Voeg de volgende bibliotheken toe.

```
using Microsoft.Speech.AudioFormat;
using Microsoft.Speech.Recognition;
```

We moeten de AudioSource en SpeechRecognitionEngine declareren vooraleer we deze kunnen gebruiken.

```
private KinectAudioSource audio;            // Interface to the Kinect audio subsystem.
private SpeechRecognitionEngine recognizer; // Speech recognizer

audio = new KinectAudioSource();
audio.FeatureMode = true;                   // Per sample; allows us to turn off AGC
audio.AutomaticGainControl = false;         // Per sample documentation, "Important to
turn this off for speech recognition"
audio.SystemMode = SystemMode.OptibeamArrayOnly;    // Per sample. Perceive this to be
"stereo"
```

De engine heeft een bepaalde woordenschat ter beschikking waar hij de ingestelde commando's mee vergelijkt. Deze woordenschat moet eerst opgehaald worden en ook gedeclareerd worden. De beschikbare woordenschat voor deze SDK is in het Engels.

```
 // Search for the installed recognizer. It must be "SR_MS_en-US_Kinect_10.0" for this
demo
RecognizerInfo ri = SpeechRecognitionEngine.InstalledRecognizers().Where(r => r.Id ==
"SR_MS_en-US_Kinect_10.0").FirstOrDefault();

GrammarBuilder builder = new GrammarBuilder();
```

Er kunnen meerdere woordenlijsten geïnstalleerd worden, en zoals bij het skelet kan er aan elke woordenlijst ook een voorrang meegegeven worden.

De recognizer word ook gedeclareerd, deze vergelijkt effectief de commando's met de woorden in de woordenlijst en zal ook het "confidence" level bepalen.

```
recognizer = new SpeechRecognitionEngine(ri.Id);    // Initializer a recognition engine
with the appropriate recognizer
```

De stemcommando's die de gebruiker wil dat de Kinect herkent moeten aan een lijst toegevoegd worden, eveneens dient deze lijst gedeclareerd te worden.

```
GrammarBuilder builder = new GrammarBuilder();
```

Nu kunnen de commando's toegevoegd worden.

```
whatIRecognize.Add("up", "down", "close", "photo", "eender welk gewenst commando");
// Add some words to be recognized
```

```
builder.Culture = ri.Culture;                                    // Set the culture of
the words
builder.Append(whatIRecognize);                                  // Pass the word list to
the GrammarBuilder

recognizer.LoadGrammar(new Grammar(builder));                    // Create a new Grammar
using the GrammarBuilder
```

Wanneer er een spraakcommando herkent wordt is het aan te raden om een EventHandler van start te laten gaan, deze geeft ons de mogelijkheid om ook effectief een actie aan een commando te binden.

```
recognizer.SpeechRecognized += recognizer_SpeechRecognized;      // Add a handler to the
SpeechRecognized event so we know when speech was recognized
```

Wanneer er een commando wordt herkend zal er overgeschakeld worden naar de EventHandler, ook kan het resultaat, dus het commando, meegenomen worden naar de EventHandler om daar elk commando de gewenste actie te geven.

```
void recognizer_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
        {
```

Het principe is als volgt, e.Result bevat het commando dat herkent was. We kozen ervoor om met een simpele if – functie te controleren aan welk commando e.Result gelijk is. De spraakherkenning is helaas niet foutloos, ondanks het “active noise control” denkt de Kinect soms commando’s te herkennen zelfs wanneer deze niet gezegd zijn. Hier komt het “confidence” level van de SpeechRecognitionEngine” zeer goed van pas. Zo is het mogelijk om effectief de actie pas uit te voeren wanneer we 90% zeker zijn dat het commando gezegd is geweest, dit geeft meer stabiliteit aan onze spraakherkenning.

```
if (e.Result.Confidence > 0.90)
```

Wanneer we bevestiging hebben wordt de actie die in de if – functie is beschreven uitgevoerd.

```
if (e.Result.Text.ToLower() == "up")
            {
                try
                {

                    nui.NuiCamera.ElevationAngle += 4;

                }
                catch (ArgumentOutOfRangeException ex)
                {
                    MessageBox.Show(ex.Message);
                }

            }
            else
                if (e.Result.Text.ToLower() == "down")
                {
                    try
                    {
                        nui.NuiCamera.ElevationAngle -= 4;
                    }
                    catch (ArgumentOutOfRangeException ex)
                    {
                        MessageBox.Show(ex.Message);
                    }
                }
                else
                    if (e.Result.Text.ToLower() == " eender welk gewenst commando”)
                    {      eender welk gewenst commando      };
```

Dit besluit de handleiding die ons een idee geeft hoe we de basiseigenschappen van de Kinect kunnen gebruiken in een Windows omgeving.

7) Testopstelling

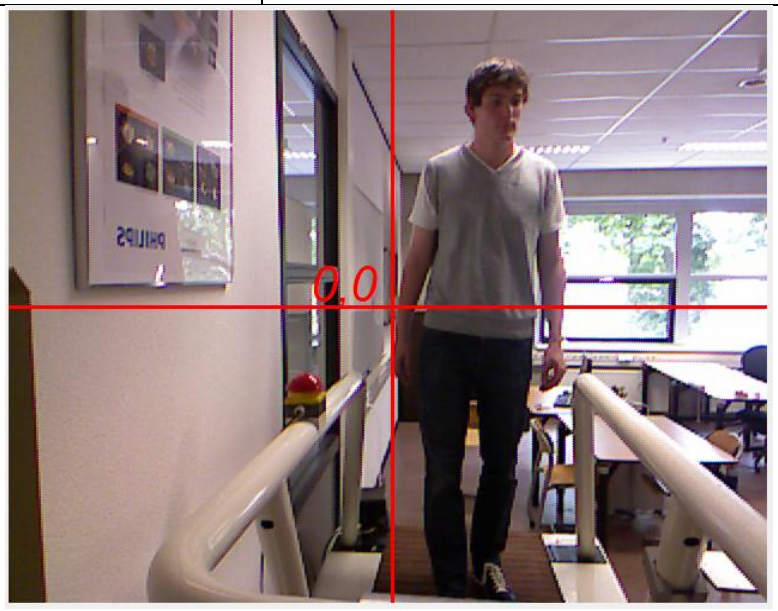
Nu de basiseigenschappen van de Kinect beschikbaar zijn, rest ons nog enkele onopgeloste vragen. Hoe bruikbaar zijn de eigenschappen van de Kinect voor real – time toepassingen? De Kinect is ontwikkeld voor de game wereld, maar is hij correct en punctueel genoeg om te implementeren in de industrie of medische sector?

Wanneer men naar de gebruiksaanwijzingen van de Kinect gaat kijken, word ons aangeraden om de Kinect boven het speelscherm te plaatsen, ligt naar beneden gekanteld en zo een 2 meter afstand van de speler, voor de optimale speel ervaring. Maar deze situatie is niet altijd mogelijk of van toepassing. Wil dat dan zeggen dat de waardes en coördinaten die de Kinect beschikbaar geeft niet meer correct zijn?

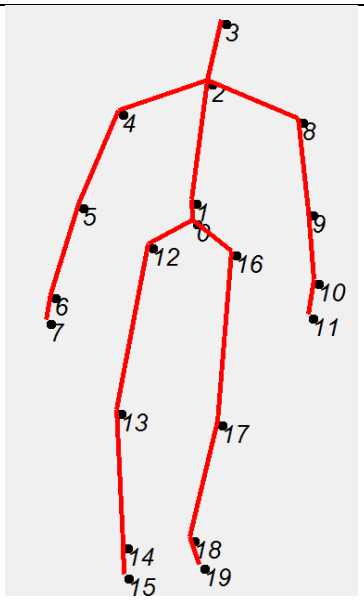
Laten we dit controleren met een paar simpele testopstellingen voor een paar alledaagse situaties. Er word een skelet opbouw gedaan van een persoon die zich op een loopband bevind en op een hometrainer. Wanneer het skelet bij elke benadering steeds goed word weergeven, kan ervan uitgegaan worden dat de coördinaten uitlezing correct en nog steeds bruikbaar is. Hierbij zal de afstand, hoogte en hoek t.o.v. het proefpersoon gevarieerd worden om te controleren welke opstelling kan effectief is en welke niet.

7.1 Testopstelling op de loopband

Positie 1:	0° t.o.v. de proefpersoon
Hoogte:	1,70 m
Afstand:	2,65 m t.o.v. de proefpersoon



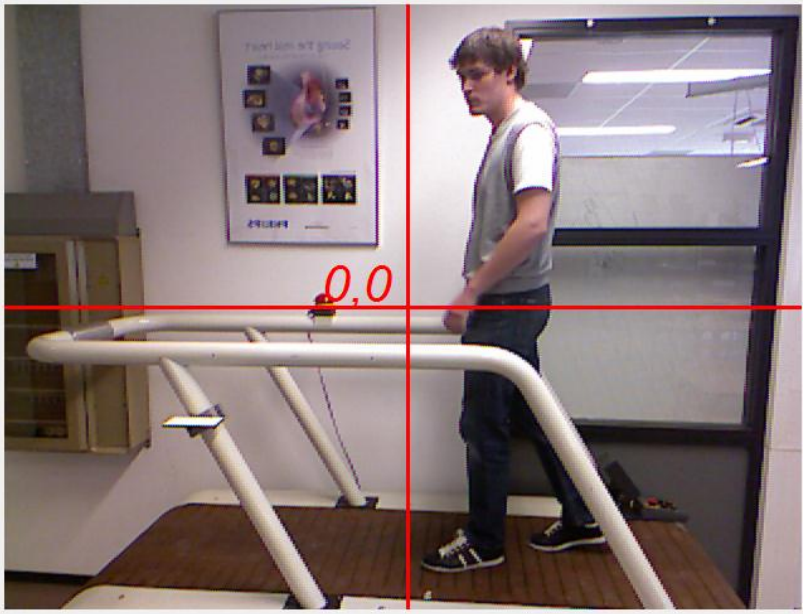
Figuur 29: opstelling loopband met 0° draaihoek



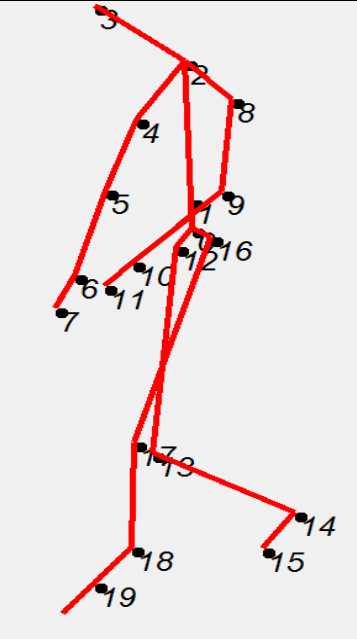
Figuur 30: opstelling loopband met 0° draaihoek skelet beeld

Conclusie: Dit is een opstelling volgens de normen die Microsoft geeft. Zoals te zien word het skelet stabiel opgebouwd en kan veronderstelt worden dat de coördinaten betrouwbaar en ook bruikbaar zijn.

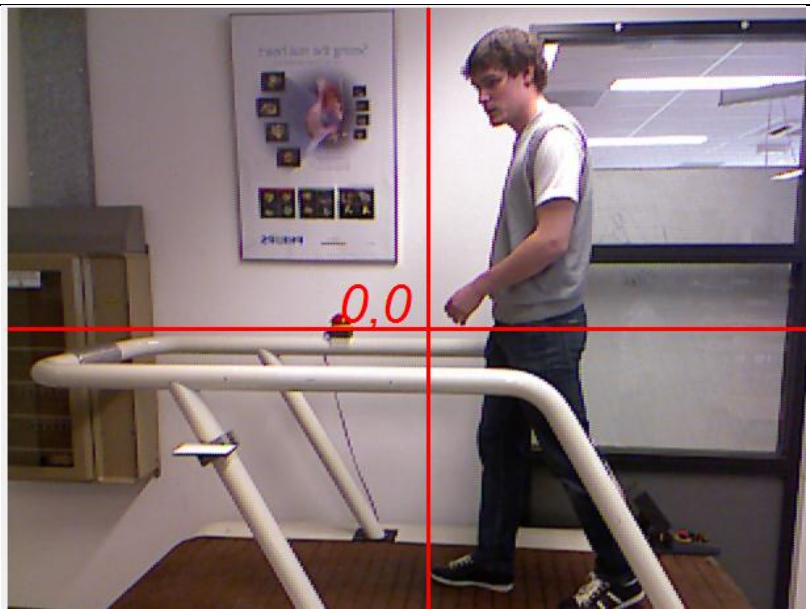
Positie 2:	90° t.o.v. de proefpersoon
Hoogte:	1,70 m
Afstand:	2,40 m t.o.v. de proefpersoon



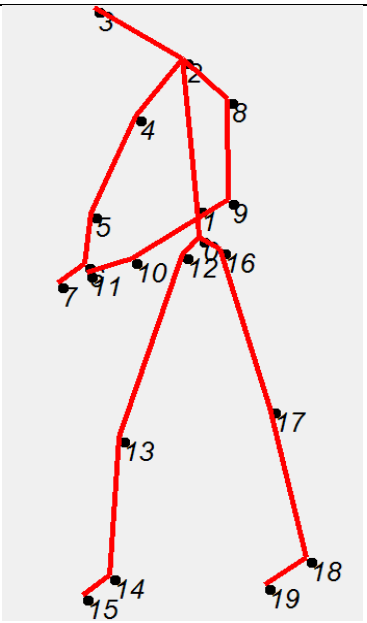
Figuur 31: opstelling loopband met 90° draaihoek



Figuur 32: opstelling loopband met 90° draaihoek skelet beeld



Figuur 33: opstelling loopband met 0° draaihoek skelet beeld

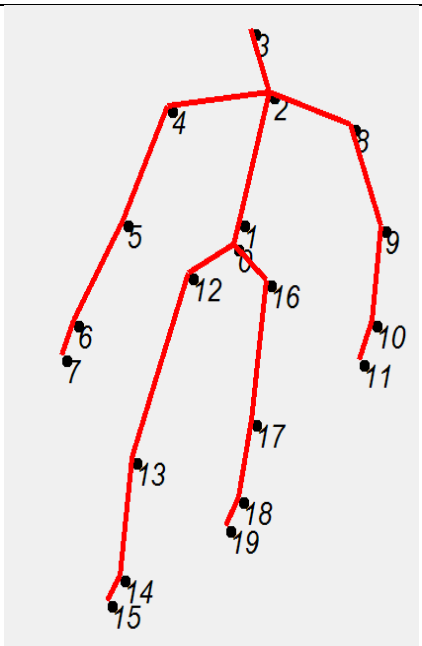


Figuur 34: opstelling loopband met 0° draaihoek skelet beeld

Conclusie: Geen slecht resultaat, de rechtervoet naar voren geeft wel een iets beter resultaat omtrent de benen. Er kan geconcludeerd worden dat de Kinect nog steeds meer als genoeg joints over heeft om een volwaardig skelet te geven.

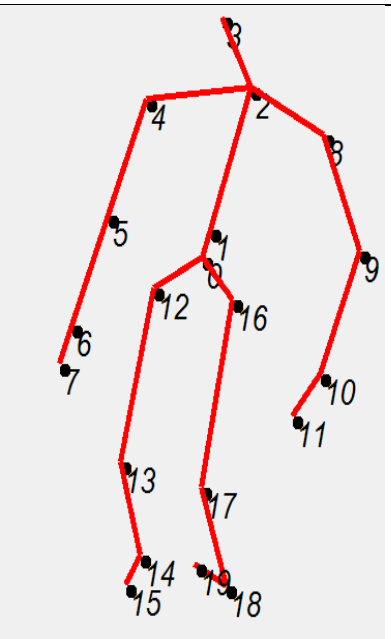
Positie 3:	30° t.o.v. de proefpersoon
Hoogte:	2,40 m
Afstand:	2,25 m t.o.v. de proefpersoon

Figuur 35: opstelling loopband met 30° draaihoek



Figuur 36: opstelling loopband met 30° draaihoek skelet beeld

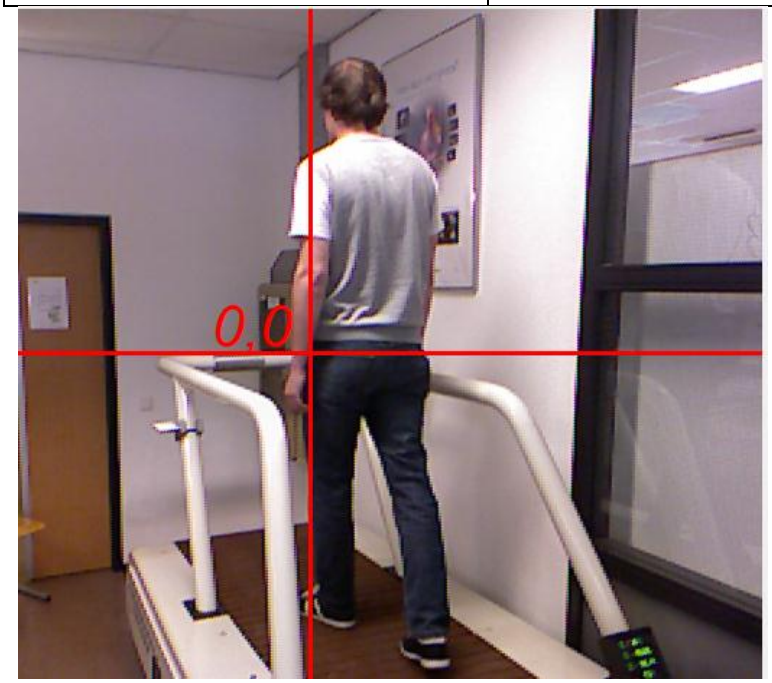
Figuur 37: opstelling loopband met 30° draaihoek



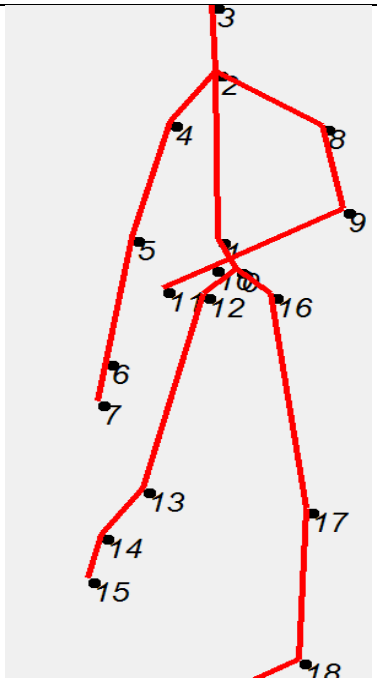
Figuur 38: opstelling loopband met 30° draaihoek skelet beeld

Conclusie: De verhouding tussen armen en benen zit duidelijk niet goed. De onderbenen worden niet realistisch weergegeven. Op het onderste beeld word een duidelijk slechte karakteristiek van de Kinect duidelijk, door het gebrek van zicht op de rechtervoet is er een verkeerde weergave van de voeten. De coördinaten van het onderlichaam zijn niet correct voor verder gebruik.

Positie 4:	150° t.o.v. de proefpersoon
Hoogte:	1,70 m
Afstand:	2,00 m t.o.v. de proefpersoon

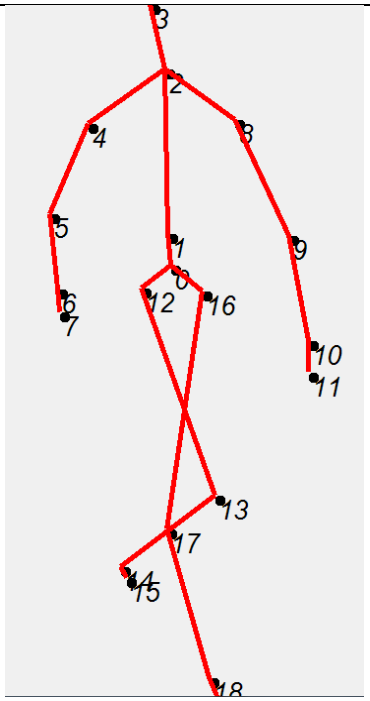


Figuur 39: opstelling loopband met 150° draaihoek



Figuur 40: opstelling loopband met 150° draaihoek skelet beeld

Figuur 41: opstelling loopband met 150° draaihoek

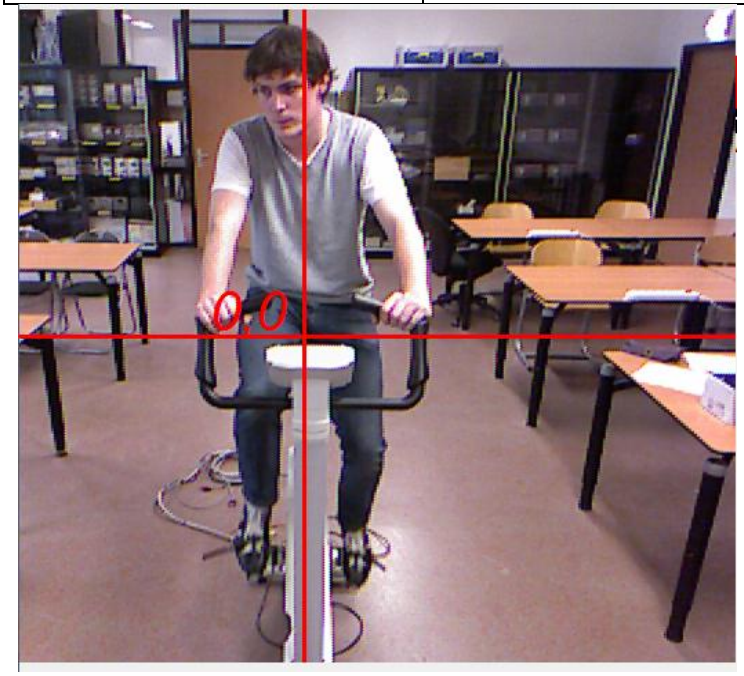


Figuur 42: opstelling loopband met 150° draaihoek skelet beeld

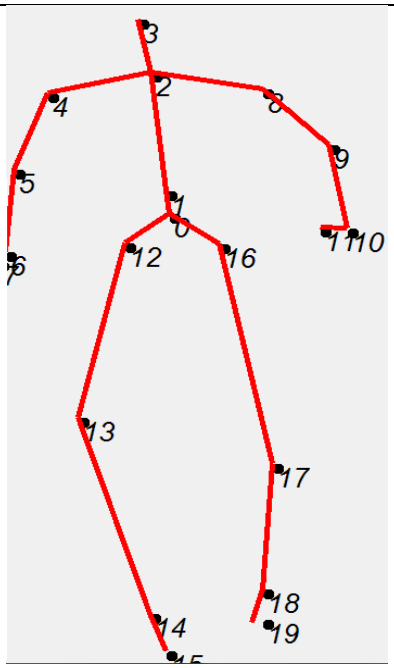
Conclusie: De Kinect is duidelijk niet ontworpen om iemand correct te volgen die met zijn rug naar de Kinect is gericht. Hij verward duidelijk de achterkant door deze als voorkant te tekenen. Hierdoor kunnen deze coördinaten zeker niet gebruikt worden.

7.2 Testopstelling op de hometrainer

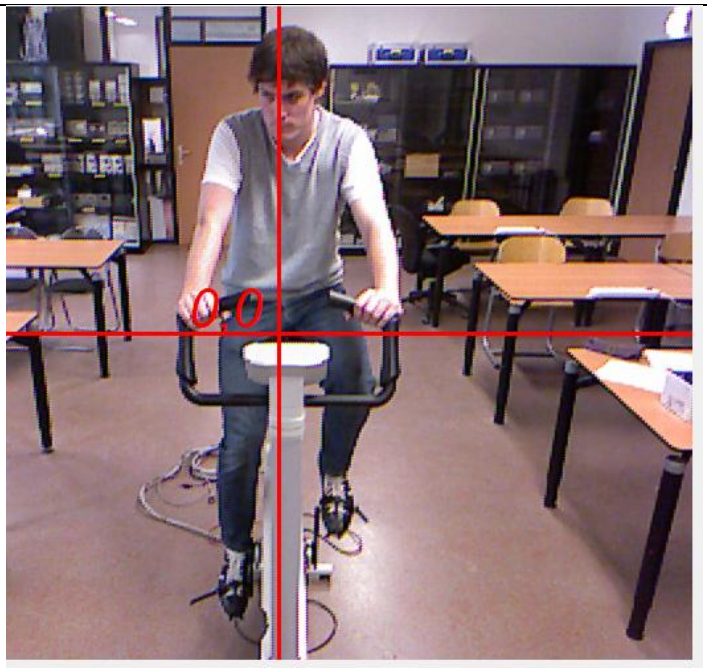
Positie 1:	0° t.o.v. de proefpersoon
Hoogte:	1,70 m
Afstand:	2,20 m t.o.v. de proefpersoon



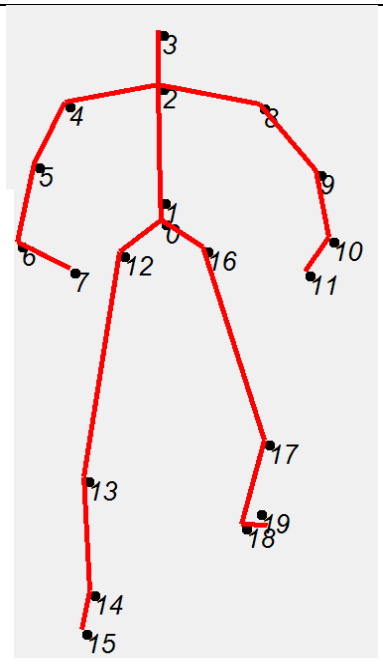
Figuur 43: opstelling hometrainer met 0° draaihoek



Figuur 44: opstelling hometrainer met 0° draaihoek skelet beeld



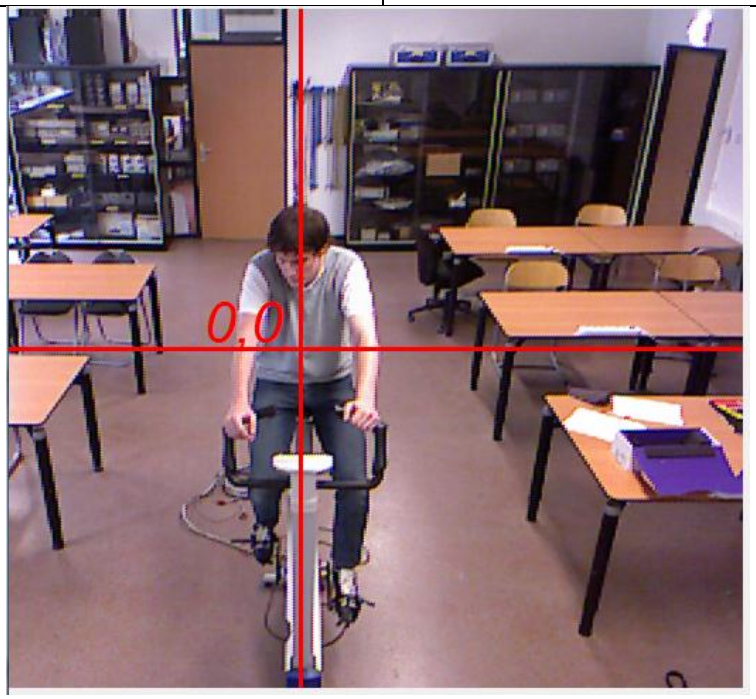
Figuur 45: opstelling hometrainer met 0° draaihoek



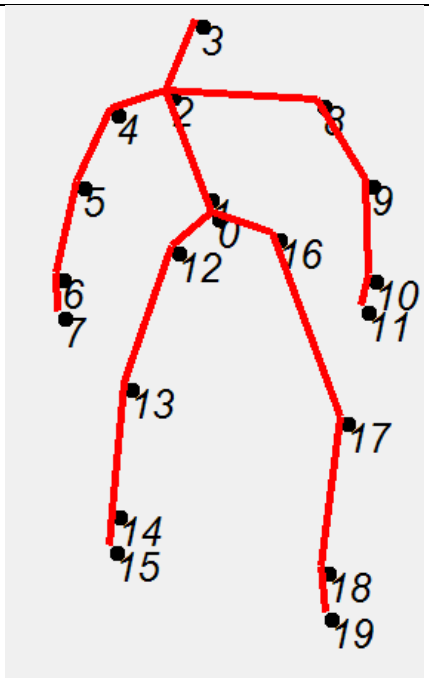
Figuur 46: opstelling hometrainer met 0° draaihoek skelet beeld

Conclusie: Dit zijn weer netjes de normen gevolgd die Microsoft aangeeft. Het skelet word weer goed weergegeven. Het frame van de hometrainer houd geen zicht op belangrijke joints tegen, de coördinaten zijn naderhand dus betrouwbaar.

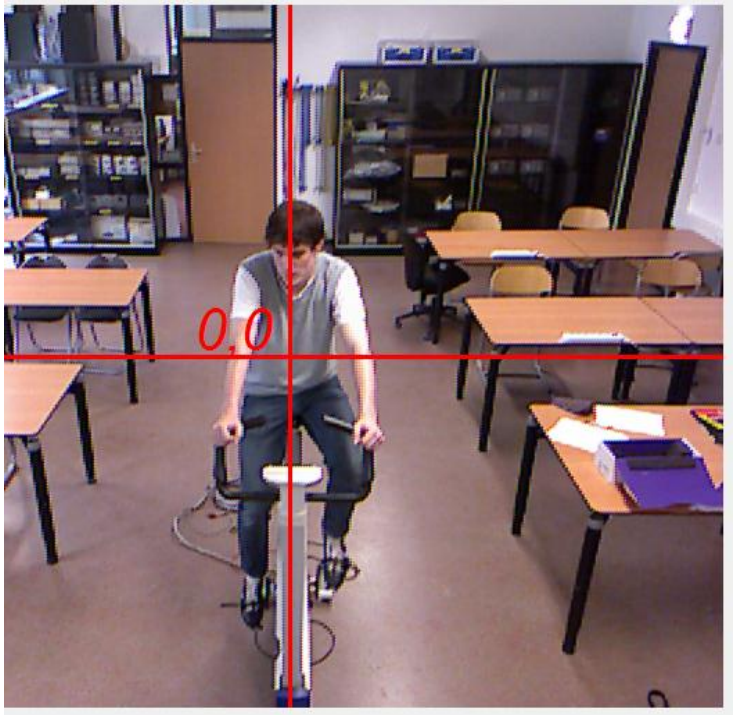
Positie 2:	0° t.o.v. de proefpersoon
Hoogte:	2,40 m
Afstand:	2,90 m t.o.v. de proefpersoon



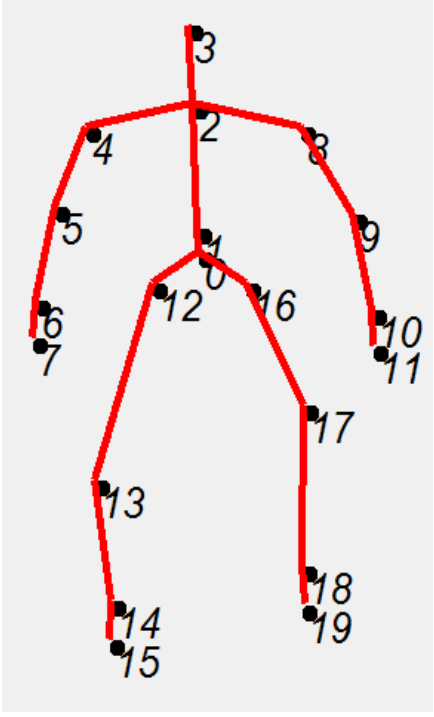
Figuur 47: opstelling hometrainer met 0° draaihoek



Figuur 48: opstelling hometrainer met 0° draaihoek skelet beeld



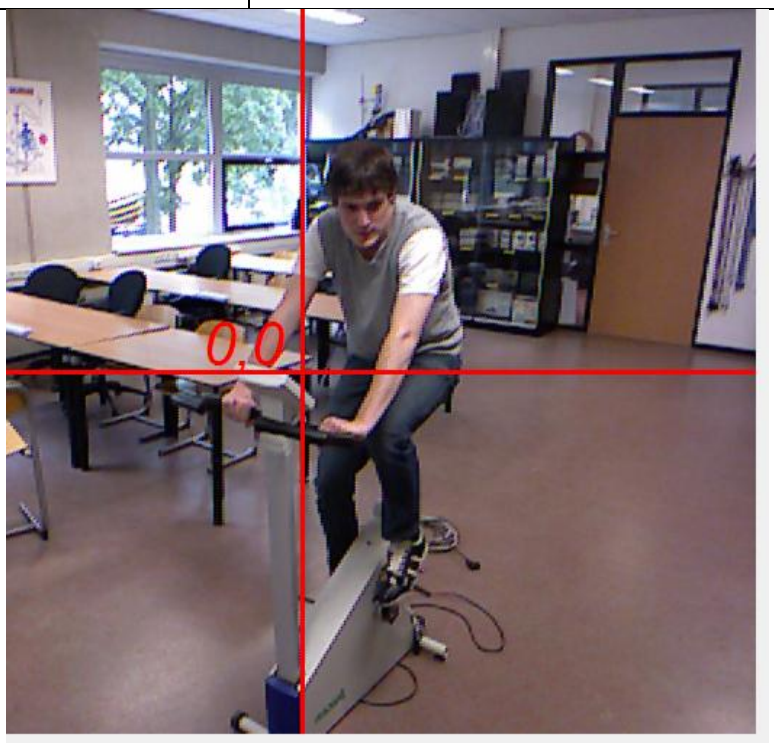
Figuur 49: opstelling hometrainer met 0° draaihoek



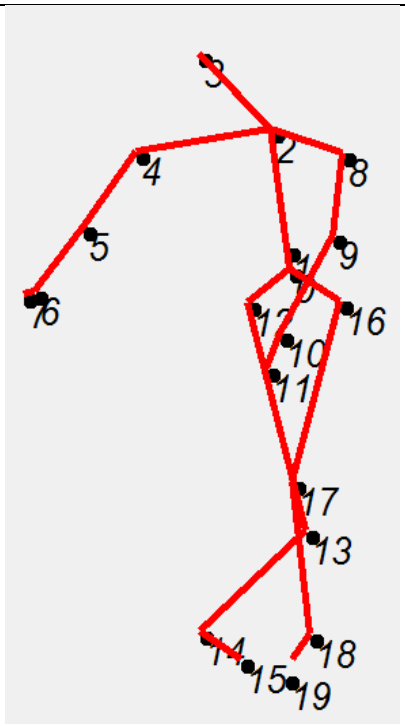
Figuur 50: opstelling hometrainer met 0° draaihoek skelet beeld

Conclusie: Er kan een volgende slechte eigenschap van de Kinect waargenomen worden bij het eerste beeld van positie 2. Omdat de rechtervoet in de lucht hangt voor de Kinect, gaat hij er van uit dat de persoon gekanteld is, daarom word de ruggengraat ook buigend naar rechts weergeven. Deze vergissing komt op onregelmatige tijdstippen voor en daarom kan een traject op een langere verloop van tijd als foutief beschouwd worden.

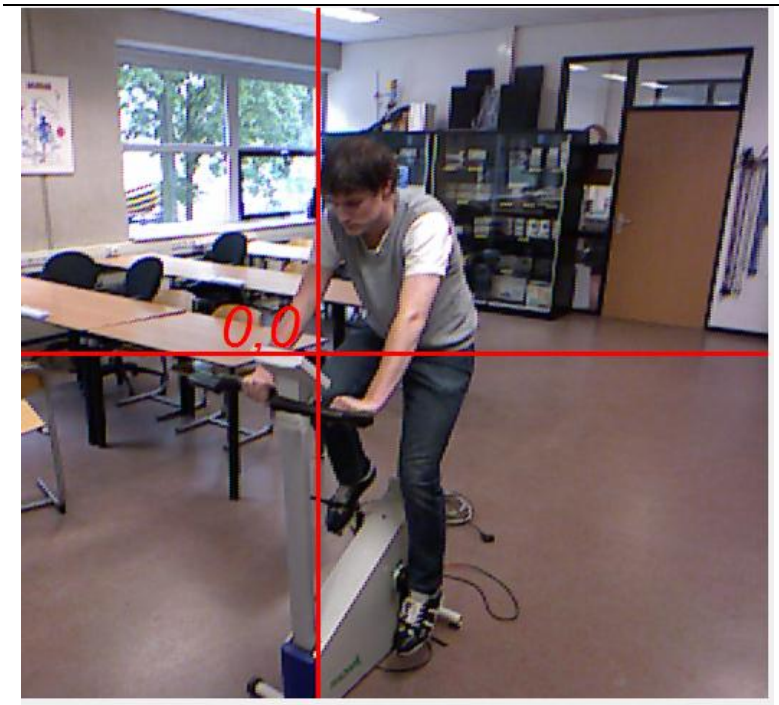
Positie 3:	30° t.o.v. de proefpersoon
Hoogte:	1,70 m
Afstand:	2,65 m t.o.v. de proefpersoon



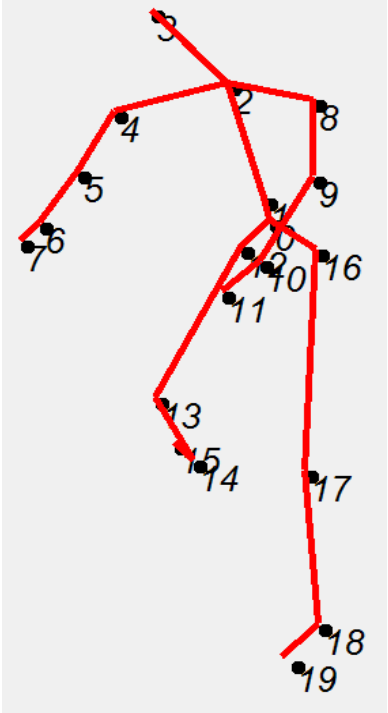
Figuur 51: opstelling hometrainer met 30° draaihoek



Figuur 52: opstelling hometrainer met 30° draaihoek skelet beeld



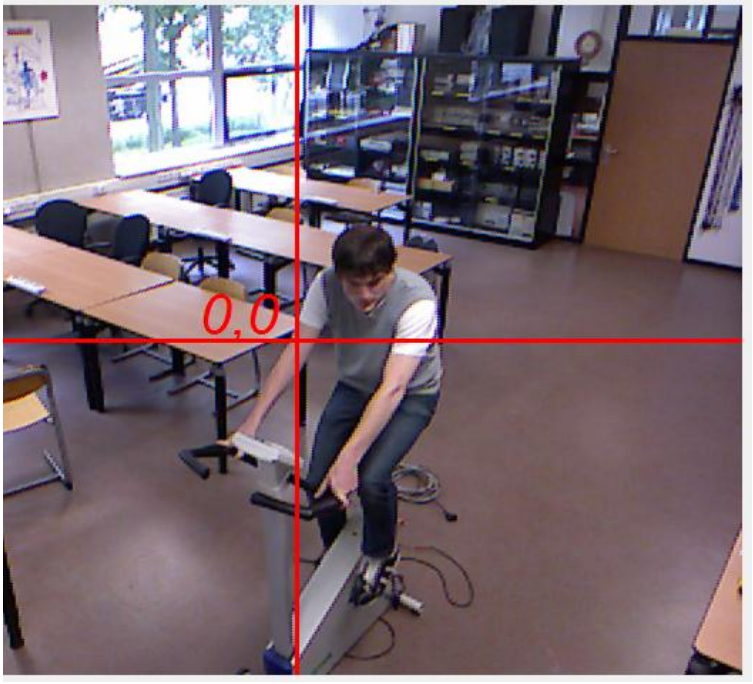
Figuur 53: opstelling hometrainer met 30° draaihoek



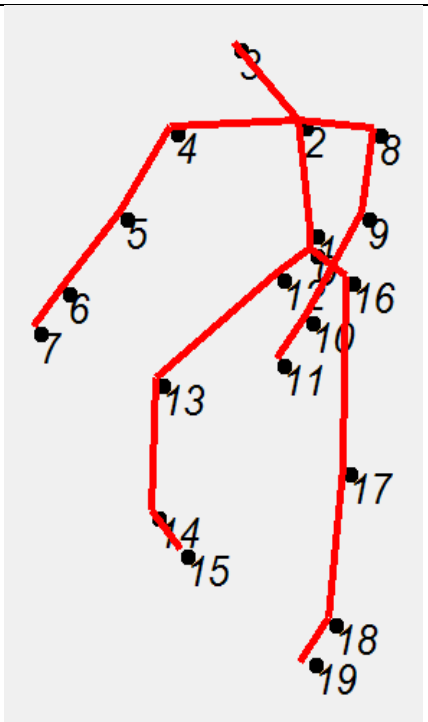
Figuur 54: opstelling hometrainer met 30° draaihoek skelet beeld

Conclusie: Er gaat duidelijk iets mis wanneer één van de voeten verdwijnt achter het frame van de hometrainer. Deze opstelling is dus maar de helft van de fietsbeweging correct.

Positie 4:	30° t.o.v. de proefpersoon
Hoogte:	2,40 m
Afstand:	2,60 m t.o.v. de proefpersoon

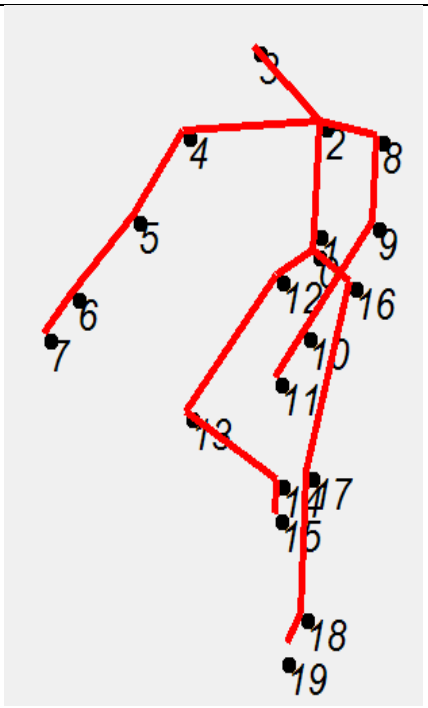


Figuur 55: opstelling hometrainer met 30° draaihoek



Figuur 56: opstelling hometrainer met 30° draaihoek skelet beeld

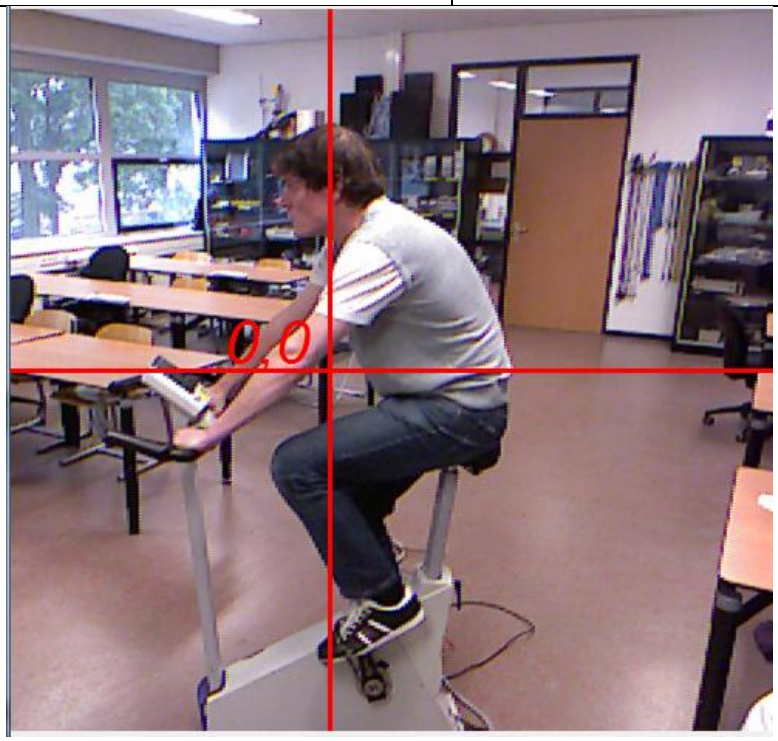
Figuur 57: opstelling hometrainer met 30° draaihoek



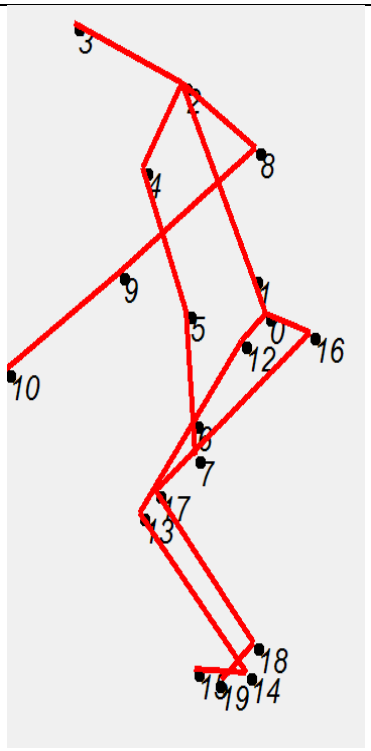
Figuur 58: opstelling hometrainer met 30° draaihoek skelet beeld

Conclusie: Onder deze hoek is de opstelling praktisch beter. We kijken als het ware over het hometrainer frame uit waardoor we nauwelijks zicht op onze joints verliezen. Toch zijn er nog korte momenten van de fietsbeweging die niet correct zijn.

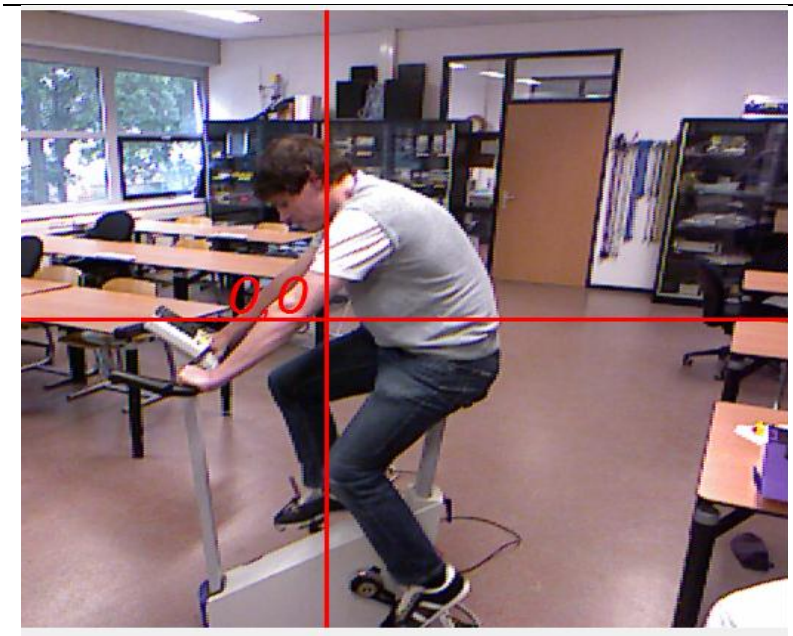
Positie 5:	90° t.o.v. de proefpersoon
Hoogte:	1,70 m
Afstand:	2,10 m t.o.v. de proefpersoon



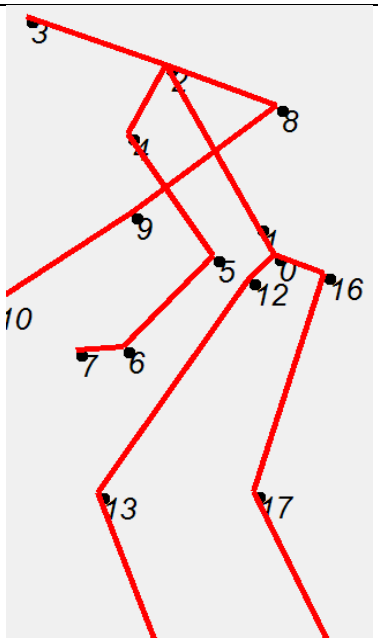
Figuur 59: opstelling hometrainer met 90° draaihoek



Figuur 60: opstelling hometrainer met 90° draaihoek skelet beeld

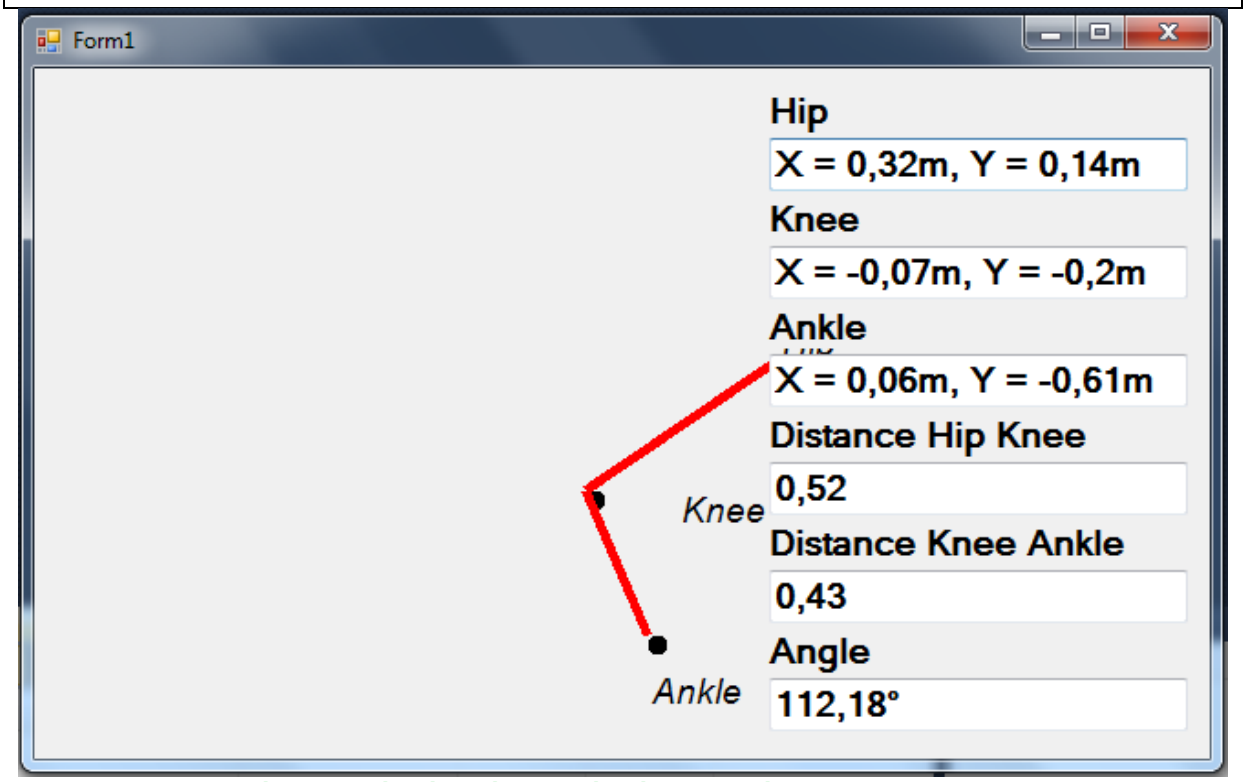


Figuur 61: opstelling hometrainer met 90° draaihoek skelet beeld



Figuur 62: opstelling hometrainer met 90° draaihoek skelet beeld

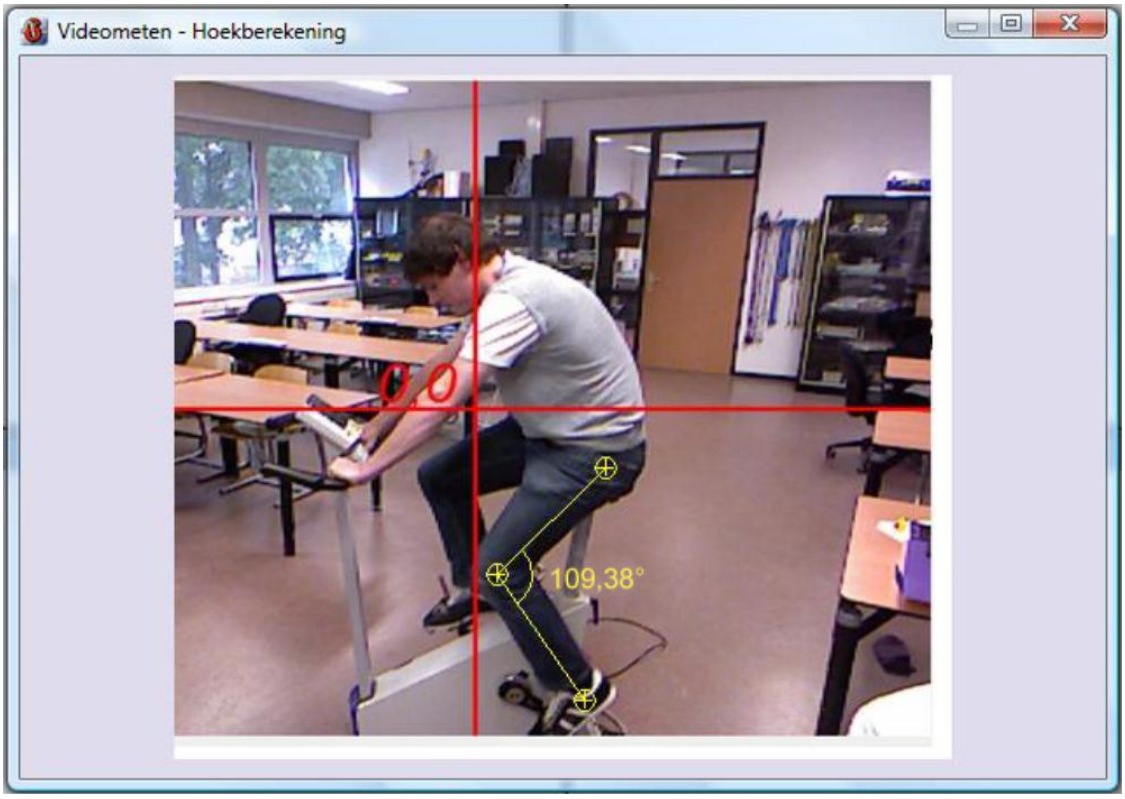
Onder deze opstelling was het ook zeer aantrekkelijk om een abstracte hoekmeting van het been te doen. Aangezien er loodrecht op het fietsvlak gekeken word, word er een 2D – situatie gecreëerd die het mogelijk maakt om met enkel de X– en Y – coördinaat een relatief simpele hoekmeting te doen. Deze ziet er als volgt uit.



Figuur 63: hoekmeting van het been met het C# programma

Conclusie: Een slecht resultaat, er kan weer geconcludeerd worden dat het zicht op teveel joints word geblokkeerd en daarom kan het skelet niet stabiel en correct weergegeven worden.

Wat de hoekberekening betreft, de hoek zal met coach 6 nogmaals bepaald worden, het resultaat is te zien in onderstaande afbeelding:



Figuur 64: hoekmeting van het been met coach 6

- Het resultaat dat verkregen is in Coach 6: 109,38 °
- Het resultaat verkregen met de Kinect: 112,18°

Een afwijking van ongeveer 3°. Er kan dus geconcludeerd worden dat de hoekberekening met een kleine afwijking correct is. Dit wil zeggen dat, zolang we een opstelling gebruiken die een 2D – situatie creëert , alle uitgelezen hoeken correct zullen zijn.

7.3 Algemene conclusie

Er kan geconcludeerd worden dat de normen die Microsoft opgeeft zeker niet achterwegen mogen worden gelaten. De beste weergave van het skelet word verkregen wanneer de Kinect voor de proefpersoon staat op een respectievelijke hoogte van ongeveer 1,70 m. De afstand mag variëren van 1,20 tot 3,50 m. In deze marge blijft de coördinaten uitlezing correct. Ook mogen er geen obstakels het zicht van de Kinect blokkeren t.o.v. de joints van de proefpersoon. Deze opstelling is misschien niet altijd even gewenst of beschikbaar waardoor we consequent in onze applicaties zullen moeten zijn.

Aan te raden positie	
Positie:	0° t.o.v. de proefpersoon
Hoogte:	1,70 m
Afstand:	1,20 – 3,50 m t.o.v. de proefpersoon

Nu deze normen bepaald zijn kan er verder gegaan worden met het laatste deel van de stag opdracht

8) Patiëntbewaking

Het eerste deel van de stage bestaat eruit om alle mogelijke voorbeelden van applicaties van de Kinect weer te geven in programma's, Deze applicaties zijn ook geïntegreerd in een basisprogramma. Deze programma's zijn ook op de CD-ROM terug te vinden. Het tweede deel van de stage bestaat er uit om een patiënt te bewaken die uit een ziekenbed dreigt te vallen. Voor deze applicatie hebben we een ziekenbed gebruikt samen met het skeletaltracking programma om zo te bepalen of een patiënt buiten de grenzen van het bed komt. In dit geval zal er een alarmering gebeuren die kan voorkomen dat de patiënt uit bed dreigt te vallen. Men kan dit met een soort van airbag of luchtkussen opvangen indien de patiënt toch te ver zou bewegen over de bedrand. Ook al is dit maar bedoeld als prototype toch word er geprobeerd om deze applicatie zo gebruiksvriendelijk als mogelijk te maken. Daarmee word bedoeld dat de Kinect zal reageren en bevestiging geven dat de patiënt veilig is. Het idee is als volgt. De Kinect word op een drie tal meter boven het voeteinde van het bed gepositioneerd. Dit zijn vaste instellingen want de grenzen van het bed liggen ook vast. Dus zal het bed zo geplaatst moeten worden dat deze mooi tussen de grenzen staat.

```
private Runtime nui;
PlanarImage PImage;
Pen pen = new Pen(Color.Red, 5);
private bool Bool=false;

nui = Runtime.Kinects[0];

nui.Initialize(RuntimeOptions.UseSkeletalTracking |
RuntimeOptions.UseDepthAndPlayerIndex |
RuntimeOptions.UseColor | RuntimeOptions.UseDepth);

nui.VideoStream.Open(ImageStreamType.Video, 2, ImageResolution.Resolution640x480,
ImageType.Color);

nui.VideoFrameReady += new EventHandler<ImageFrameReadyEventArgs>(FrameReady);

nui.SkeletonFrameReady += nui_SkeletonFrameReady;
```

Natuurlijk zal het inkomende beeld weer door een Bitmap gestuurd worden.

```
Bitmap PImageToBitmap(PlanarImage PImage)
        {
            Bitmap bmap = new Bitmap(PImage.Width, PImage.Height,
PixelFormat.Format32bppRgb);
```

```
            BitmapData bmapdata = bmap.LockBits(new Rectangle(0, 0, PImage.Width,
PImage.Height), ImageLockMode.WriteOnly, bmap.PixelFormat);
            IntPtr ptr = bmapdata.Scan0;
            Marshal.Copy(PImage.Bits, 0, ptr, PImage.Width * PImage.BytesPerPixel *
PImage.Height);
            bmap.UnlockBits(bmapdata);
            return bmap;
        }
```

De grenzen kunnen aangepast worden naar wens in de FrameReady EventHandler.

```
void FrameReady(object sender, ImageFrameReadyEventArgs e)
        {

            PImage = e.ImageFrame.Image;
            Bitmap bmap = PImageToBitmap(PImage);
            Graphics draw = Graphics.FromImage(bmap);
            Pen pen1 = new Pen(Color.Red, 3);
            Font myFont = new System.Drawing.Font("Helvetica", 20, FontStyle.Italic);
            Brush myBrush = new SolidBrush(System.Drawing.Color.Red);
            pictureBox1.Image = bmap;
            draw.DrawLine(pen1, ((PImage.Width / 2) -70), ((PImage.Height/2)),
((PImage.Width / 2) - 110), ((PImage.Height/2) +200));
            draw.DrawLine(pen1, ((PImage.Width / 2) + 70), ((PImage.Height / 2)),
((PImage.Width / 2) + 110), ((PImage.Height / 2) + 200));
            draw.DrawString("Grens", myFont, myBrush, ((PImage.Width / 2) - 150),
((PImage.Height / 2)-50));
            draw.DrawString("Grens", myFont, myBrush, ((PImage.Width / 2) + 80),
((PImage.Height / 2) - 50));
        }
```

De volgende programmacode is die van de patiëntbewaking en zal vervolgens ook uitgelegd worden.

Zoals eerder vermeld is zal de patiënt gevold worden dankzij zijn skeletopbouw en joints. Maar omdat de meerderheid van het lichaam bedekt zal alleen het hoofd gevolgd worden. Dit is wel niet zo efficiënt is gebleken. De Kinect bepaald namelijk al zijn coördinaten aan de hand van alle beschikbare joints. Dus hoe minder joints aanwezig zijn, hoe minder correct zijn de coördinaten van het hoofd. Toch zal dit voorlopig moeten volstaan in dit prototype.

Een Skelet tracking EventHandler word gedeclareerd.

```
nui.SkeletonFrameReady += nui_SkeletonFrameReady;

void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
       {

            frame = e.SkeletonFrame;

            IEnumerable<SkeletonData> skeletonlist = (from s in frame.Skeletons
                                                      where s.TrackingState ==
SkeletonTrackingState.Tracked
                                                      select s);

            IEnumerator<SkeletonData> enumerator = skeletonlist.GetEnumerator();

            SkeletonData skeleton = skeletonlist.FirstOrDefault();

            if (skeleton != null)
            {
                XR = Convert.ToDouble(skeleton.Joints[JointID.Head].Position.X);
                XR = Math.Round(XR, 2);
                YR = Convert.ToDouble(skeleton.Joints[JointID.Head].Position.Y);
                YR = Math.Round(YR, 2);
                ZR = Convert.ToDouble(skeleton.Joints[JointID.Head].Position.Z);
                ZR = Math.Round(ZR, 2);
                TxtLocatie.Text = string.Format("X = {0}m, Y = {1}m, Z = {2}m", XR, YR,
ZR);

JointsCollection joints = skeleton.Joints;
       }
```

De applicatie moet gebruiksvriendelijk zijn, we willen de patiënt zo afhankelijk mogelijk maken. De volgende situatie word gecreëerd. De patiënt plaatst zich in het bed en een spraakcommando om de bewaking te starten. De Kinect geeft bevestiging met het afspelen van een opname “Patiënt observation activated”. Wanneer de patiënt zich buiten de opgestelde grenzen gaat begeven, krijgt het bewakingscentrum een bevestiging “Patiënt possibly falling”. Dit is het teken waarop er gereageerd moet worden. Wanneer de patiënt ontwaakt geeft hij het commando om de bewaking te beëindigen. Er word bevestigd met een opname “patiënt observation deactivated”. De bewaking word afgesloten.

Omdat er spraakcommando's worden gebruikt zal er een audio en spraakherkenningsEventHandler worden opgestart.

```
private void InitializeAudio()
        {
            audio = new KinectAudioSource();
            audio.FeatureMode = true;                           // Per sample; allows us
to turn off AGC
            audio.AutomaticGainControl = false;                 // Per sample
documentation, "Important to turn this off for speech recognition"
            audio.SystemMode = SystemMode.OptibeamArrayOnly;    // Per sample. Perceive
this to be "stereo"

            InitializeRecognizer();
        }
        private void InitializeRecognizer()
        {

            // Search for the installed recognizer. It must be "SR_MS_en-US_Kinect_10.0"
for this demo
            RecognizerInfo ri = SpeechRecognitionEngine.InstalledRecognizers().Where(r
=> r.Id == "SR_MS_en-US_Kinect_10.0").FirstOrDefault();

            if (ri == null)
            {
                throw new ApplicationException("Could not locate speech recognizer.");
            }

            recognizer = new SpeechRecognitionEngine(ri.Id);                //
Initializer a recognition engine with the appropriate recognizer
            GrammarBuilder builder = new GrammarBuilder();
            Choices whatIRecognize = new Choices();
```

De comando's die de Kinect moet herkennen worden hier ingesteld.

```
            whatIRecognize.Add("Start","Stop");    // Add some words to be recognized
            builder.Culture = ri.Culture;                                   // Set the
culture of the words
            builder.Append(whatIRecognize);                                 // Pass the
word list to the GrammarBuilder

            recognizer.LoadGrammar(new Grammar(builder));                   // Create a
new Grammar using the GrammarBuilder
```

```
            recognizer.SpeechRecognized += recognizer_SpeechRecognized;     // Add a
handler to the SpeechRecognized event so we know when speech was recognized

            recogStream = audio.Start();                                    // Start
streaming audio from the Kinect
            recognizer.SetInputToAudioStream(recogStream,                   // Tell the
recognizer what stream to "listen" to and tell it how to sample
                                             new
SpeechAudioFormatInfo(EncodingFormat.Pcm, 16000, 16, 1, 32000, 2, null));
            recognizer.RecognizeAsync(RecognizeMode.Multiple);              // Start
recognizing words
        }
```

De volgende EventHandler stelt ons instaat acties aan onze commando's te binden.

```
void recognizer_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
        {


            textBox1.Text = e.Result.Text;
            if (e.Result.Confidence > 0.95)
            {
                if (e.Result.Text == "Start")
                {
```

Omdat er opnames worden afgespeeld heft visual studio een media speller ter beschikking. De boolean die word gedeclareerd zal zo meteen gebruikt worden in de effectieve grens controle.

```
                    System.Media.SoundPlayer audioPlayer = new
System.Media.SoundPlayer(@"activated_wav1.wav");

                    audioPlayer.Play();
                    Bool = true;
                }
                else
                    if (e.Result.Text == "Stop")
                    {
                        System.Media.SoundPlayer audioPlayer = new
System.Media.SoundPlayer(@"deactivated_wav1.wav");

                        audioPlayer.Play();
                        Bool = false;
```

```
                }
            }
        }
```

Nu dat de patiënt weet dat de bewaking is gestart moet er nog een grenscontrole gebeuren. Er word gekozen om dit via een Methode te doen. Ook word aangeraden om de methode te laten activeren door de EventHandler van de skelet engine. Zo zal de controle, wanneer er een skelet gevonden is tenminste, 30 maal per seconde worden uitgevoerd.

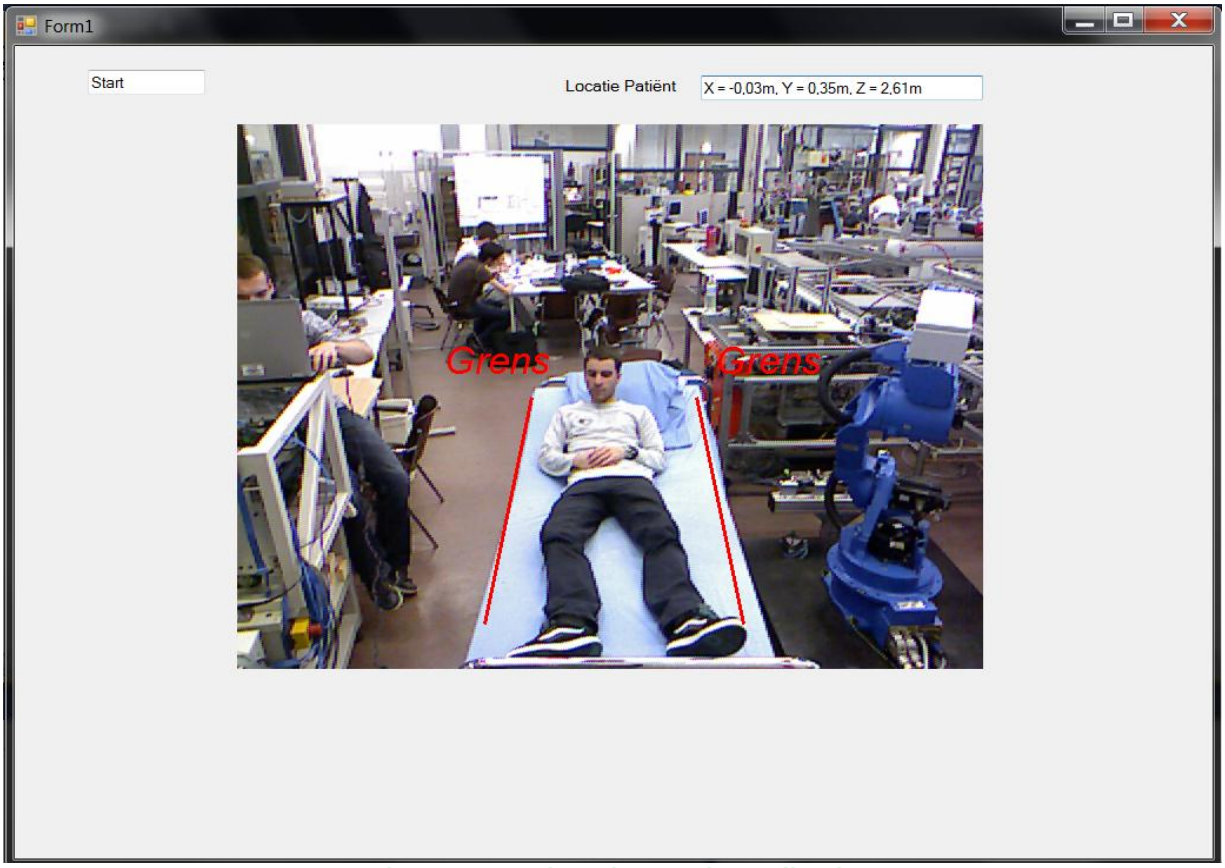
De Methode noemt hier “InitializeControle”.

```
private void InitializeControle()
        {
            if (Bool == true)
            {
                if ((XR < -0.40) || (XR > 0.40))
                {
                    if (k == 0)
                    {
                        System.Media.SoundPlayer audioPlayer = new
System.Media.SoundPlayer(@"possibly falling_wav1.wav");
                        audioPlayer.Play();
                        k = 1;
                    }

                }
                else
                {
                    k = 0;
                }
            }
        }
```

Dankzij de opstelling van deze applicatie is de patiënt geheel onafhankelijk van verplegers of dergelijk. Ook word in deze toepassing de patiënt gevolgd met normaal beeld, maar er kan voor gekozen worden om dit te vervangen door dieptebeeld of zelfs gewoon door het skelet poppetje om de privacy te bewaren.

De applicatie is ook instaat, met verdere uitbreiding, om de patiënt effectief op te vangen voor hij valt. 30 Maal per seconde word de controle uitgevoerd dus kan er een waarschuwingssignaal gegenereerd worden binnen 33 milliseconden.

Via USB komt het signaal met een vertraging van 250 milliseconden aan op het controle systeem. Veronderstel dat deze weer via USB een signaal verstuurd naar een pompsysteem van een airbag of luchtkussen. Dan kan op een halve seconde een airbag openvliegen om zo de patiënt op te vangen.



Figuur 65: Patiëntobservatie applicatie

9) Conclusie

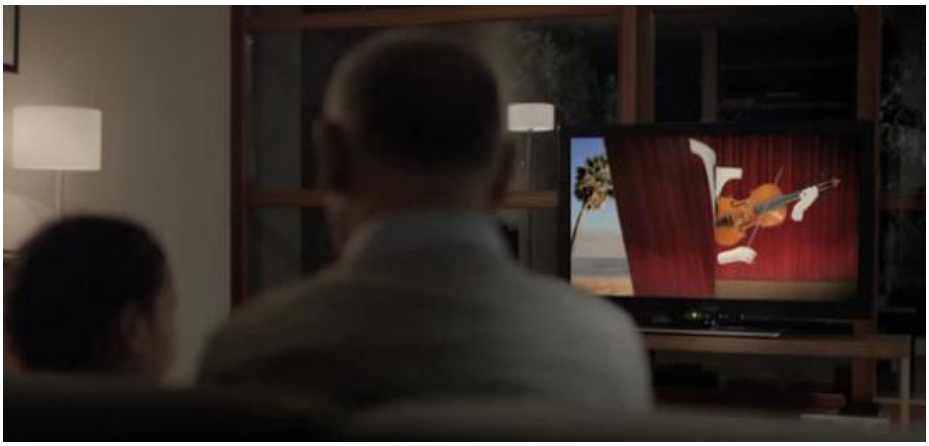
Zoals eerder vermeld is de Kinect uitzonderlijk goedkoop voor waar hij allemaal tot toe in staat is. De waardes die de Kinect nu ter beschikking stelt, zij pixelwaarden of coördinaten. Deze zijn relatief correct.en daarom zijn ze ook zeer bruikbaar in sectoren zoals biometrie.

Figuur 66: Lichaamscoördinaten volgen met een skelet

Want met deze lichaam coördinaten kan het hele lichaam gevolgd worden, net zoals makkelijk hoekberekeningen kunnen worden gedaan tussen de verschillende lichaamsdelen. Op deze manier kan men patiënten volgen die aan een revalidatie proces bezig zijn.

De Kinect was oorspronkelijk bedoeld als entertainment in de huiskamer. Gebruiksvriendelijk is het magische woord voor Microsoft.

Figuur 67: Gebruiksvriendelijke Kinect in de woonkamer



Figuur 68: Gebruiksvriendelijke Kinect in de woonkamer

Maar vlug werd duidelijk dat hij tot zoveel meer instaat is. Zo heeft hij zijn weg al gevonden in de operatiekamer, als hulp voor chirurgen.

Figuur 69: Kinect in de operatiezaal

Figuur 70: Kinect in de operatiezaal

De Kinect als onze sportcoach.

Figuur 71: De Kinect als sportcoach

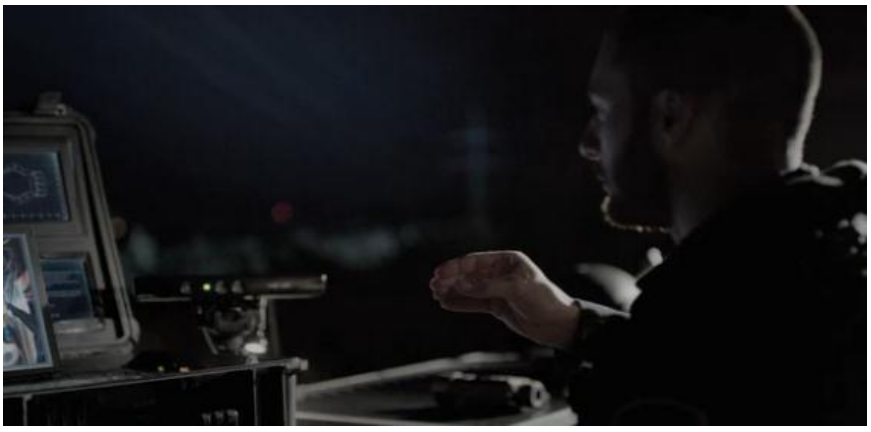
De Kinect als onze leraar.

Figuur 72: De Kinect als onze leerkracht

De Kinect als onze muziek coördinator.

Figuur 73: De Kinect als onze muziekleraar

De Kinect is enorm precies en klaar voor elke klus, zelfs robot aansturing.



Figuur 74: Robotarm aansturen met de Kinect

Figuur 75: Robotsturing met de Kinect

De Kinect zal letterlijk onze horizon verruimen …

Figuur 76: Horizon verruimen

We kunnen concluderen dat de Kinect een uitermate grote rol zal spelen in onze toekomst

10) Bijlage

CD-ROM: Alle programma-applicaties en installatieprogramma's zijn hier op te vinden

Journalistiek artikel: Deelname opdracht voor de Agoriaprijs en de Vlaamse scriptie prijs.

Xbox Kinect Applicaties

Bart Suelze & Kenny Colda,
XIOS Hogeschool Limburg en Zuyd Hogeschool Heerlen

De Kinect, aanvankelijk als Project Natal, gelanceerd door Microsoft in november 2010, was bedoeld als een leuke game- uitbreiding voor de Xbox 360. Toch is de Kinect tot veel meer instaat. Dat is alvast de conclusie van de Erasmus studenten aan de Hogeschool Zuyd in het Nederlandse Heerlen.

De twee Erasmus studenten, Bart Suelze en Kenny Colda, namen de uitdaging aan om te bewijzen dat de Kinect meer in zijn mars heeft, dit aan de hand van een sterk onderzoek en hun geavanceerde programmeer capaciteiten.

Het "Kinect" talent

Hardnekkige Xbox 360 - gebruikers herkennen ze wel, de Xbox -"Avatars", onze altijd lachende, virtuele representatie van ons zelf. Ze zijn bedoeld om een zo vertrouwd mogelijke omgeving en interface voor onszelf te creëren. Dankzij de Kinect lijken ze niet alleen op ons, maar bewegen ze ook zoals wij dat doen. Ze bootsen ons helemaal na! We hebben ons nog nooit zo verbonden gevoeld met onze favoriete games. En hoe natuurlijk dit ook overkomt, we staan er vrijwel nooit bij stil hoe ongelooflijk innovatief deze Kinect eigenschap wel is. Want zonder de Kinect of om het even welk andere controller aan te raken weet de Kinect precies waar wij ons in de speelruimte bevinden. Hij weet niet alleen waar wij ons bevinden, hij kan ook perfect onze lichaamsdelen van elkaar onderscheiden, zonder dat jij daarbij gebruik maakt van merkers of externe indicatoren. Geloof het maar, zoiets vraagt een enorme hoeveelheid aan rekenvermogen, iets wat we de kleine zwarte Kinect niet meteen zouden geven.

De bescheidenheid van Microsoft

Met de virtualisatie van ons hele lichaam hield het niet op voor Microsoft. We kunnen de Kinect ook commanderen want hij herkent onze stem commando's. Er is alles aan gedaan om ons van zoveel mogelijk luxe te voorzien en dit alles voor een zeer schappelijke prijs. Zonder dat Microsoft het echt doorhad sloegen ze een gat in de markt met hun Kinect. Want het was vlug duidelijk dat de Kinect z'n plaats verder dan de huiskamer is! Ofwel was Microsoft gewoon erg bescheiden … gelukkig zagen ze vlug het licht. Ze brachten een jaar na de release van de Kinect, in november 2011, een "software development kit" uit voor Windows. Voor elke geïnteresseerde programmeur ging er een nieuwe wereld open. Overal op het internet doken er filmpjes op van mensen die applicaties met de Kinect hadden ontworpen. Dit was ook het startschot voor Bart, Kenny en Hogeschool Zuyd.

Kinect soft- en hardware

Toen de software beschikbaar was, kon het programmeren beginnen. De laatste "sdk" die werd vrijgegeven. was de v1.0 Beta 2. Deze werd dan ook gebruikt door de Erasmus studenten. De versie werkt onder Windows 7 onder een .Net omgeving.

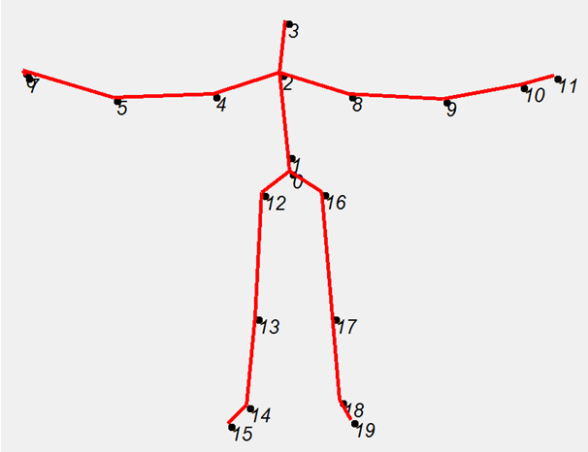
KINECT for Windows

Hoe de Kinect ons lichaam nu traceert is erg ingenieus maar niet moeilijk om te begrijpen. De Kinect heeft drie slimme cameras. Twee ervan verantwoordelijk voor wat wij “het dieptezicht” noemen en één voor het standaard- RGB-Beeld (red, green, blue). Eén van de dieptezicht cameras werkt als infrarood(IR) sensor. Deze stuurt een enorme hoeveelheid IR– stralen uit en die worden weer herkaatst door objecten die zich voor de Kinect bevinden. De tweede dieptezicht camera merkt de weerkaatste IR – stralen op en dankzij het tijdsverschil tussen de verzonden en ontvangen stralen berekent hij de afstand van het object ten opzichte van de Kinect. Zo krijg je het befaamde diepezicht, hier opgebouwd met blauw waarden. (tekening). De dieptezicht afbeelding geeft ons al een idee van hoe de Kinect ons uit de omgeving haal: de IR-stralen die op ons weerkaatsen keren veel vlugger terug.

De Kinect bevat ook een zogenaamde “Skelet Engine”. Deze bouwt op de achtergrond een skelet op en het is over deze skeletten dat de avatar wordt heen geplekt. Ook al verbergt Microsoft deze skeletten, toch kunnen ze van groot nut zijn.

Het Kinect skelet

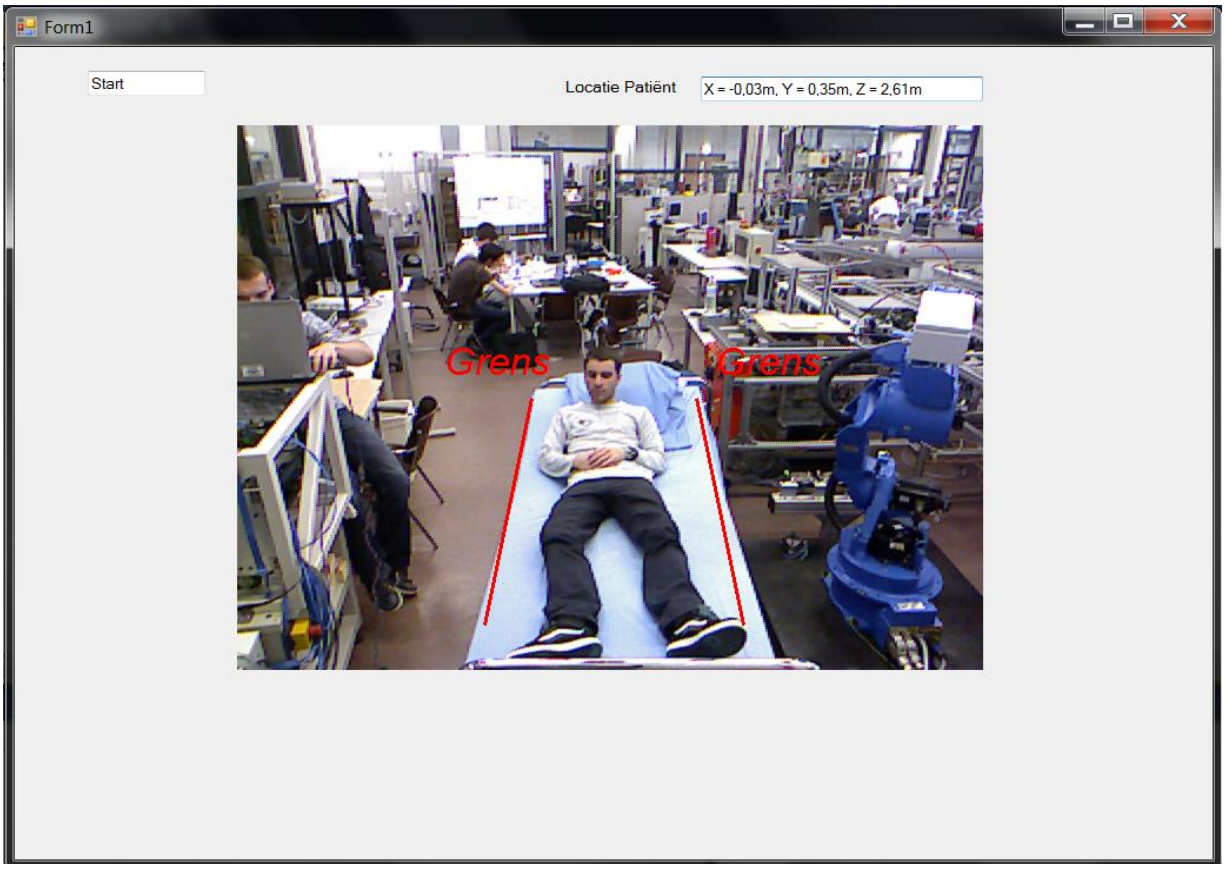
Zoals gezegd de Kinect heeft het talent om onze lichaamsdelen te onderscheiden. Hij onderscheid ze niet alleen, hij plaatst ook markeringen op de belangrijke knikpunten van ons lichaam, de “joints” genoemd. Het mooie hieraan is dat de Kinect de coördinaten van deze joints ter beschikking stelt. Elke programmeur die deze coördinaten van ons lichaam ter beschikking heeft kan tussen de punten lijnen trekken om zo een goede representatie van het skelet te maken. Dat zie je op de volgende tekening.



De mogelijkheden die deze coördinaten met zich meebrengen zijn hier eindeloos. Zo kan de Kinect ons volgen terwijl we sporten en ons zelfs onze techniek bijsturen om onze prestaties te verhogen. Kijk maar naar het game “Kinect Sports”.

De Kinect toepassing

We weten nu tot wat de Kinect allemaal in staat, maar wat kan men nu hier allemaal mee bereiken. Die vraag stelde ook Bart en Kenny aan zichzelf. Hiervoor hebben ze hun naar de medische sector gericht. Een alledaags probleem wat voorkomt in de medische sector is het toezicht houden op onrustige of demente patiënten. Dit word nu gedaan met camera toezicht en dat geeft ook weinig privacy voor de patiënten. De jongens hebben hierrond een leuke toepassing ontworpen. Ze gebruiken het draadpoppetje van de Kinect om de privacy mee te bewaren en stemcommando's om de applicatie te starten of beëindigen. Het idee is dus als volgt: De begeleiders van de patiënt leggen de patiënt s 'avonds in zijn bed. Met een simpel stemcommando word de applicatie gestart. De lichaamscoördinaten van de patiënt worden constant uitgelezen. Wanneer die coördinaten over een bepaalde grens gaan, wanneer dat de patiënt uit het bed begint te rollen, dan gaat er een alarmering af bij de geleiders. Deze kunnen dan vlug van het skeletbeeld overgeschakeld worden naar normaal beeld om te kijken wat er mis gaat. De uitlezing van de coördinaten gebeurd 30 maal per seconde. Aan deze snelheid is het dus mogelijk een patiënt ook effectief te redden wanneer hij uit het bed valt, eventueel door het open klappen van een airbag. De Kinect bied dus veiligheid, betrouwbaarheid en privacy. De volgende afbeelding geeft een idee van de userinterface.



Kinect voor een betere toekomst

We kunnen ons al geen toekomst zonder de Kinect meer voorstellen. Hij is gebruiksvriendelijk, goedkoop en betrouwbaar en dat motiveert ook Bart en Kenny om verder te gaan met het Kinect verhaal. Niet lang meer of er is een Kinect in elke huiskamer te vinden en veel verder ook. Zo heeft de Kinect zijn plaats al gevonden in de chirurgie, in de operatiezaal.

Chirurgen hoeven hun röntgen foto's niet meer vuil te maken om ze te bekijken en zelfs in en uit te zoomen op de cruciale punten.
Leraren gebruiken hem als hulp bij het lesgeven. Dankzij zijn sterke audio- eigenschappen, zal de Kinect ooit zijn plaats als muziek coördinator ook waarmaken.

We zien zelfs een toekomst waarin sterke en reusachtige robots, onze snelle en wendbare bewegingen nabootsen, dankzij de snelle coördinaat uitlezing van de Kinect. Conclusie: geen klus te zwaar voor de kleine Kinect. We hebben het laatste nog niet van hem gehoord.

www.xbox.com