Procedurele animatie van menselijke interactie door middel van Inverse Kinematics en Fuzzy Logic

Gaétan Deglorie

Promotor: prof. dr. ir. Sofie Van Hoecke
Begeleider: dhr. Koen Samyn

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: elektronica-ICT

Vakgroep Industrieel Systeem- en Productontwerp
Voorzitter: prof. Kurt Stockman
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2013-2014

UNIVERSITEIT
GENT

Procedurele animatie van menselijke interactie door middel van Inverse Kinematics en Fuzzy Logic

Gaétan Deglorie

Promotor: prof. dr. ir. Sofie Van Hoecke
Begeleider: dhr. Koen Samyn

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: elektronica-ICT

Vakgroep Industrieel Systeem- en Productontwerp
Voorzitter: prof. Kurt Stockman
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2013-2014

UNIVERSITEIT
GENT

Voorwoord

Toen ik dit onderwerp zag staan tussen de lijst van masterproeven, wist ik dat dit iets voor mij was. Al sinds het middelbaar onderwijs had ik interesse in Computer Graphics. *Hoewel animaties niet mijn sterkste kant waren, wou ik toch de uitdaging aangaan. Het pad om te komen tot deze scriptie was niet zonder moeilijkheden, ik zou er wellicht niet gekomen zijn zonder de steun van verscheidene mensen.*

Ik zou daarom allereerst mijn thesisbegeleider Koen en mijn promotor Sofie willen bedanken. Koen in het bijzonder voor de grote hoeveelheid tijd die hij heeft gestopt in de begeleiding bij het afwerken van de praktische realisatie. Tevens wil ik mijn familie en vrienden bedanken voor de steun die ze me hebben geboden door de jaren heen.

Gaétan Deglorie, juni 2014

Toelating tot bruikleen

"De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik.
Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef."

Gaétan Deglorie, juni 2014

Procedurele animatie van menselijke interactie door middel van Inverse Kinematics en Fuzzy Logic

door

Gaétan Deglorie

Scriptie ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen:
elektronica-ICT (multimedia en informatietechnologie)

Academiejaar 2013–2014

Promotor: Prof. Dr. Ir. S. Van Hoecke
Begeleider: Dhr. K. Samyn
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

Vakgroep Industriële Systeem- en Productontwerp
Voorzitter: Prof. K. Stockman

Samenvatting

In dit werk proberen we animaties van menselijke interacties te modelleren op basis van procedurele technieken. Er wordt onderzocht hoe animaties van dit type kunnen uitgewerkt worden met behulp van Inverse Kinematics en Fuzzy Logic. De twee modellen worden bestudeerd en vergeleken. Hierbij wordt de handdruk bestudeerd als simpele interactie. Het doel van dit werk is een beter inzicht te krijgen in de opbouw van animaties van menselijke interacties in het algemeen.

Trefwoorden

procedurele animatie, fuzzy logic, inverse kinematics, menselijke interactie

Procedural Animation of Human Interactions using Inverse Kinematics and Fuzzy Logic

Gaétan Deglorie

Supervisor(s): Sofie Van Hoecke, Koen Samyn

Abstract— **This article tries to model animations of human interactions using procedural techniques. A comparative study of the use of Inverse Kinematics and Fuzzy Logic to achieve these animations is conducted. More specifically a handshake is virtually modelled and investigated.**

Keywords— **procedural animation, Fuzzy Logic, Inverse Kinematics, human interaction**

I. Introduction

THE animation of interactions is often avoided because of the difficulties that are accompanied by it. Matching specific parts of an animation with their respective environment is only just being implemented into mainstream digital content creation tools. An example is the adaptation of the walk cycle of a human character with regard to a sloped surface. (Even on non-flat surfaces.) Procedural animations offer a solution to these problems, one of the most popular techniques being Inverse Kinematics. In this research Inverse Kinematics will be compared with Fuzzy Logic as a technique for animating human interactions. Fuzzy Logic is a technique which was designed as an alternative to the standard logic controller. It uses 'vague' logic to determine the best solution for a given problem. It has rarely been used to solve animation problems. In this abstract, we will propose a model of a handshake in which each subject participating in the interaction can use either Inverse Kinematics or Fuzzy Logic.

II. Case study

To determine a model of any animation it is always crucial to start from the real life situation. A short study was made of handshakes in real life to produce a basic set of rules to define a handshake virtually.

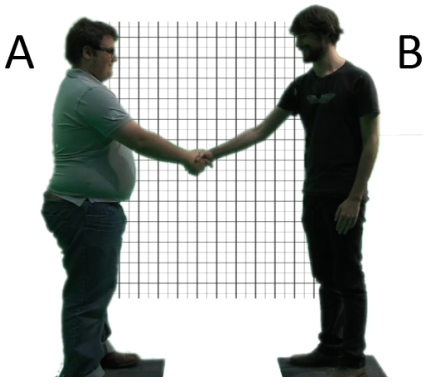


Fig. 1. Case study

On examining the results it was clear that the motions of different people can differ greatly. The personality of each human is unique and influences the animation in its own way. The average model extracted from this data makes some assumptions based on symmetry to make the base animation. These assumptions include but are not limited to: people grab each others hands in the center between both of them, the shake action itself will move perpendicularly to the floor. The determined rules specify the proportions of the different paths taken by both hands during the animation.

III. Framework

To achieve realistic interaction between two virtual subjects, a framework for asynchronous interactions has to be created first. This framework provides information about each other for both subjects participating in the interactions. This allows the subject to see the other subject during the animation. This information can then be used for any interaction animation built on top of it, in this case a handshake.

The framework provides a way to give each virtual subject a personality. Each trait of a subject is described using a value pair, binding a name to numeric value. These pairs are combined into dictionaries that represent a subject's personality. The handshake model uses a set of traits to personify each animation according to the personalities of each subject.

IV. Handshake

The handshake includes a part where both hands are touching, referred to as the grip. To mimic this grip, without using collision detection, synchronization is required. To synchronize only part of the animation we split the animation up into distinct phases.

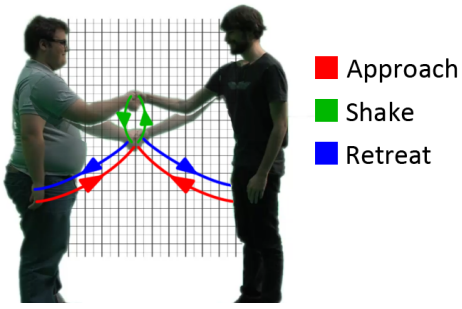


Fig. 2. Handshake phases

Each phase determines a spline segment, the proportions of each segment are calculated at the start of that phase. This makes the model slightly responsive to outside changes, like moving the subjects. As the phases themselves are not responsive to these changes, it is recommended not to enact any such changes to the subjects while the animation is playing. A snapshot of the finished model can be viewed in figure 3.

Fig. 3. Handshake model

Each subject can have its own animation technique: Inverse Kinematics or Fuzzy Logic. Even though an interaction animation fundamentally requires information about both parties, each party can decide for themselves how they enact the rules to produce a handshake animation.

A. *Inverse Kinematics*

Inverse Kinematics is a technique designed to produce a set of joint rotations on a robotic arm for a given target position of the end effector. To use this technique for animation, animation paths have to be used. An animation path describes the location of point in space for a period of time. The algorithm solves towards this point for several points in time, thus creating the illusion of natural motion.

Each phase of the handshake animation builds its own animation path. The usage of paths makes it easier to synchronize both hands during the shake phase, as both hands can use the same animation path.

A downside of Inverse Kinematics is its lack of respect to joint limits. It will solve to a mathematically viable solution, but that doesn't mean a physically possible one. Every frame after a solution is calculated, the programmer has to manually check whether the joint limits have not been exceeded and rotate back to a valid rotation if necessary.

B. *Fuzzy Logic*

A Fuzzy Logic controller does not have a specific implementation, it is programmed using a set of rules that give numeric output based on numeric input. Each set of rules is used to control the state of a single joint. To simplify the control of a joint, each rotation is split up in Euler Angles which are individually controlled. Figure 4 displays a simplified control scheme for a single rotation axis.

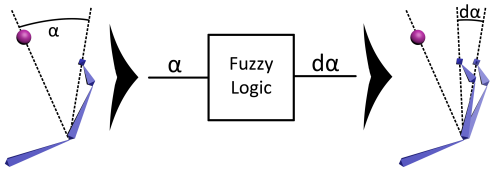


Fig. 4. Fuzzy Logic controller

As shown, the controller does not immediately solve to the target position, it takes the controller a number of iterations to get there. This makes it possible to animate with only using a static target position, instead of the animation paths discussed earlier. In the final model however, animation paths are used only for the shake phase. This is to ensure that both hands remain together during the shake phase.

With Fuzzy Logic there are two ways of respecting joint limits: fixing limit crossing after rotations are applied (equivalent to Inverse Kinematics) or limiting the rotations calculated from the controller before applying them.

V. Discussion

Both types of controllers, Inverse Kinematics and Fuzzy Logic, can be calibrated in wide arrange of manners. In the end Inverse Kinematics delivers a higher precision than Fuzzy Logic. However Fuzzy Logic's independence of paths gives it great potential. Especially to relieve the burden of figuring out what exact route a limb might take during an animation for the animator. If calibrated correctly the limb will find the most natural way to move towards a target position.

Because the model uses procedural techniques and each handshake phase is built according to the dimensions of each subject, the animation automatically adapts to its subjects. Figure 5 displays a frame of the handshake animation where both subjects have a significant difference in size.

Fig. 5. Adaptability of animation to its subjects

VI. Conclusion

The model that has been proposed here adapts the animation based on not only the personalities but also the sizes and positions of both subjects. The model is far from perfect, but this work provides the foundation for future models of human interactions. Fuzzy Logic proved to be a powerful animation technique, that requires more attention in the future.

Inhoudsopgave

Gebruikte afkortingen

CCD	Cyclic Coordinate Descent
CG	Computer Graphics
FCL	Fuzzy Control Language
GUI	Graphical User Interface
IK	Inverse Kinematics
PID	Proportioneel, Integrerend en Differentiërend
TCP	Transmission Control Protocol
WPF	Windows Presentation Foundation

Hoofdstuk 1

Inleiding

Deze masterproef is gekaderd in het domein van computer animatie en is uitgevoerd in Universiteit Gent Campus Kortrijk.

Computer graphics heeft de laatste decennia veel vooruitgang geboekt, op vlak van animatie is men op zoek naar modellen die meer realisme bieden en minder rekenintensief zijn. Dit creëerde de groep van procedurele animaties, deze animaties worden automatisch gegenereerd met omgevingsinformatie en tijd als parameters.

Er wordt onderzoek gedaan naar de implementatie van animatie bij interactie tussen verschillende karakters, zoals het animeren van de handdruk. Hierbij wordt gekeken welk animatie algoritme het best geschikt zou zijn en welke problemen zich eventueel kunnen voordoen. Er wordt ook onderzocht hoe de animatie zelf verandert afhankelijk van de karakters die in interactie gaan.

Er wordt een raamwerk gecreëerd om verschillende algoritmes te testen en te vergelijken. Het raamwerk zal slechts één animatie testen, die vooral concentreert op interactie met armen en handen.

De scriptie is als volgt opgebouwd: in de literatuurstudie wordt een overzicht gegeven van animatie en animatie algoritmen met de bijkomende problemen van multi-karakter animatie. In het hoofdstuk “Probleem definitie” wordt kort de probleemstelling naar voor gebracht. Hierop volgt een hoofdstuk “Casestudy” die vooronderzoek over menselijke interactie naar voor brengt. Vervolgens wordt kort de achterliggende structuur van het raamwerk beschreven in het hoofdstuk “Realisatie”. In het voorlaatste hoofdstuk “De handdruk” wordt de opbouw van de handdruk binnen het raamwerk uiteengezet samen met de bekomen resultaten. Ten slotte wordt er afgesloten met het hoofdstuk “Conclusie en toekomstperspectieven”.

Hoofdstuk 2

Literatuurstudie en technologieonderzoek

> Animation - The technique of photographing successive drawings or positions of puppets or models to create an illusion of movement when the film is shown as a sequence.
> *Oxford Dictionary 2013*

2.1 Voorgeschiedenis

Een korte voorgeschiedenis die wat basisprincipes van animatie zal verklaren. Eerst wordt het oudste type van animatie besproken: animatie met de hand. Gevolgd door het effect van de computer op animatie, het ontstaan van video games en een korte bespreking van de verdere groei.

2.1.1 Animatie met de hand

De eerste animaties waren geconstrueerd uit handgetekende afbeeldingen, de vroegst te vinden animatie was een video gemaakt door J. Stuart Blackton in 1906. [1] De tekeningen werden gemaakt op een schoolbord met krijt, deze tekeningen werden snel achtereenvolgens afgespeeld om zo de illusie van beweging te creëren. Het maken van animaties op deze manier werd vervolgens met pen en papier gedaan, wat leidde tot het ontstaan van tekenfilms in de filmindustrie. Eén van de meer gekende voorbeelden is Walt Disney. De animatieteams van toen bestonden uit één (of meer) hoofdanimatoren en een groep animatieassistenten. De hoofdanimatoren creëerden een aantal afbeeldingen die grof weergaven wat de film moest weergeven, deze afbeeldingen noemt men *keyframes*. Indien de grove uitlijning van de animatie werd goedgekeurd werden de keyframes doorgegeven aan de assistenten, zij moesten de animatie vervolledigen op basis van deze keyframes. Deze bijkomende afbeeldingen is men *inbetweens* gaan noemen. Dit proces noemt men *keyframe animation*. [2]

2.1.2 Animatie met computer

In 1951, 5 jaar na het ontstaan van de eerste computer "ENIAC", werd de eerste computer gebouwd die tekst en simpele tekeningen in real-time op een scherm kon weergeven, de zogenaamde "Whirlwind I". [3] Het computerscherm was een oscilloscoop en het was mogelijk om lijnen te tekenen. Sindsdien heeft de evolutie van computers een exponentiële stijging gekend, en hier lijkt geen einde aan te komen. De Wet van Moore voorspelt dat elke 2 jaar het aantal transistors op een chip, en dus indirect ook de computerkracht, verdubbelt. Tot zo ver lijkt deze stelling nog te kloppen. De evolutie van de computer heeft een nieuwe sector doen ontstaan binnen *Computer Science*, de zogenaamde *Computer Graphics*. Computer Graphics heeft een grote vooruitgang gekend sinds de WhirlWind I doordat men streefde naar fotorealisme. Animatie vond al snel genoeg zijn weg binnen Computer Graphics, een eerste voordeel van Computer Graphics over handgetekende animatie was de vereenvoudiging van keyframe animation. Nu was het slechts vereist om de keyframes te creëren en de computer rekende de inbetweens uit voor de animator. Computer Graphics werd heel populair binnen de filmindustrie allereerst voor visuele verbeteringen maar later ook om hele films mee te maken. [4] Ondertussen ontstond een sector met een bijzondere interesse in Computer Graphics: de video games industrie.

2.1.3 Video Games

De geschiedenis van video games begint al in de jaren '40 met T.T.Goldsmith Jr. en E.R.Mann die een patent aanvroegen voor een uitvinding genaamd de "cathode ray tube amusement device". [5] Toch zouden video games geen grote populariteit kennen tot de jaren 1970 - 1980. Net als de kracht van computers kende video games industrie ook een exponentiële groei. De video games industrie is uitgegroeid tot een van de grootste gebruikers van Computer Graphics.

2.1.4 Groei

Door het streven naar fotorealisme zijn de animaties kwalitatiever geworden. Deze kwaliteit veroorzaakt een hogere kost, waardoor de hardware een knelpunt wordt. Men is dus meer en meer op zoek naar betere manieren om animaties op te slaan en af te spelen. Zoals eerder vermeld levert keyframing een optimalisatie op vlak van opslag aangezien niet elke frame vooraf opgeslagen hoeft te zijn. Men begon te zoeken naar optimalisatie en nieuwe technieken om aan volgende noden te voldoen:

- Hogere variëteit aan animaties (realisme)
- Hogere kwaliteit van animaties
- Lagere opslag en verwerkingskost
- Eenvoudiger te creëren

Deze noden in combinatie met de evolutie van Computer Graphics hebben aanleiding gegeven tot een nieuw model van animatie. Waarbij men vroeger de volledige visuele representatie manipuleerde, levert het nieuwe model abstractie op vlak van manipulatie. Voor deze abstractie maakt men gebruik van een skelet, hiervan haalt het model zijn naam: skelet animatie.(zie sectie 2.2) Dit nieuwe model bracht heel wat nieuwe manieren om animatie te creëren. Er ontstonden verschillende technieken zoals procedurele en parametrische animatie. Animatie blijft een heel actief onderzoeksveld binnen Computer Graphics.

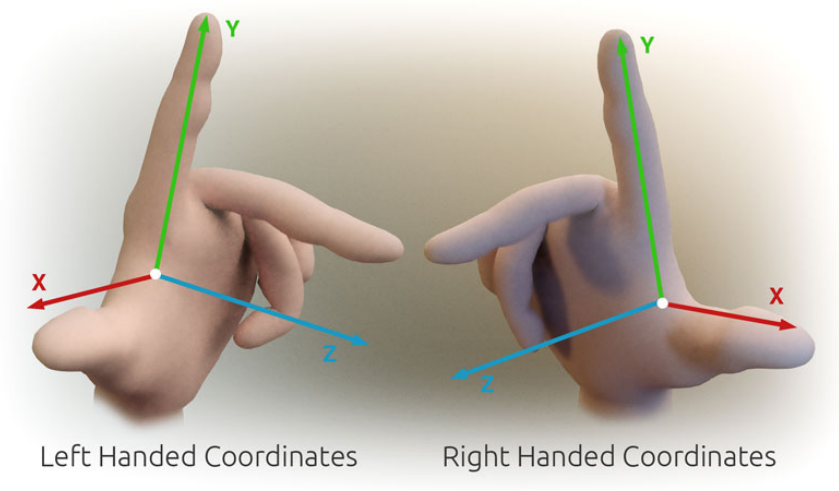
2.2 Skelet Animatie

Skelet animatie is een techniek waarbij men vertrekt van twee elementen: het skelet van een karakter en zijn visuele representatie. Het skelet wordt geanimeerd en de visuele representatie wordt vervolgens gebonden aan dit skelet via een proces genaamd *skinning.*(zie 2.2.4)

2.2.1 CG Objecten

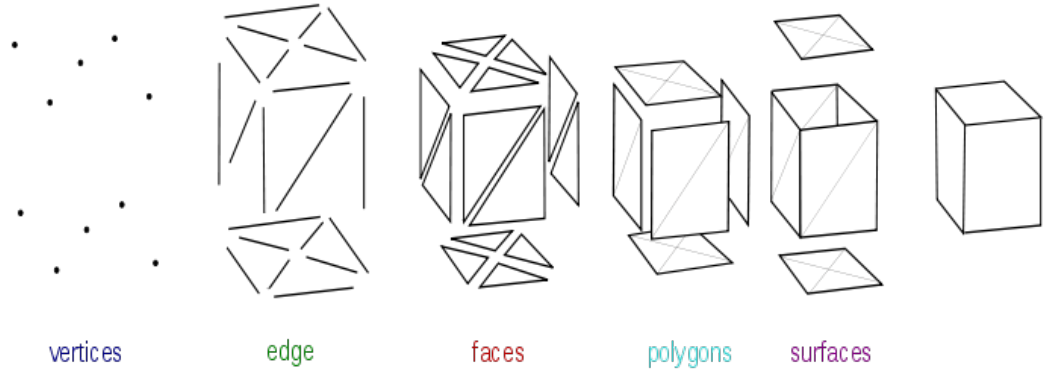
Objecten in Computer Graphics worden beschreven in 2D of 3D. 2D objecten krijgen hun vorm door zijn geometrische specificaties te beschrijven in termen van positie binnen een coördinatenstelsel. Men kan bijvoorbeeld een lijnsegment beschrijven door de posities van zijn eindpunten en een polygoon door de posities van zijn *vertices*. Een *vertex* (mv. vertices) is de beschrijving van een locatie of punt aan de hand van coördinaten binnen Computer Graphics. 3D objecten worden op een gelijkaardige manier opgebouwd, men breidt het coördinatenstelsel uit naar drie dimensies. Binnen Computer Graphics wordt er veelal gebruik gemaakt van een Cartesische coördinatenstelsel[1]. In figuur 2.1 ziet men de twee manieren om een orthogonaal stelsel te creëren op basis van drie assen: linkshandig en rechtshandig.

[1]Cartesische coördinatenstelsel is een orthogonaal coördinatenstelsel met dezelfde eenheid voor alle dimensies.



Figuur 2.1: 3D coördinatenstelsels[2]

Om de beschrijving van objecten in 3D te vereenvoudigen, worden vertices hiërarchisch gegroepeerd. De vertices van 3D objecten worden gegroepeerd tot polygonen, die op hun beurt samengevoegd worden tot een *mesh*. In figuur 2.2 ziet men hoe vertices gecombineerd worden tot een mesh.



Figuur 2.2: Van vertices naar mesh[3]

Men spreekt in beide gevallen van locale coördinaten of *local space*. Indien men objecten wil verschuiven (translatie) kan men dit op twee manieren doen, enerzijds de vertices van het object binnen het beschreven coördinatenstelsel verplaatsen of de oorsprong van het coördinatenstelsel verplaatsen. Men kiest veelal voor de tweede manier aangezien deze de originele data van het object onveranderd laat. Deze bewerking wordt veralgemeend

[2]Van "Cartesian coordinate system" beschikbaar op Wikipedia. Creative Commons Attribution-Share Alike 3.0 Unported license. http://en.wikipedia.org/wiki/Cartesian_coordinate_system

[3]Van "Polygon mesh" beschikbaar op Wikipedia. Creative Commons Attribution-Share Alike 3.0 Unported license. http://en.wikipedia.org/wiki/Polygon_mesh

met behulp van matrixberekeningen[4], voor een translatie in 3D kan dit als volgt worden voorgesteld:

$$p' = p + T = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.1}$$

Waar p een punt voorstelt, T de translatiematrix en p' het resulterende punt. Voor rotatie van objecten kunnen ook transformatiematrices beschreven worden:

$$p' = R_x.p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} . \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{2.2}$$

Waar p een punt voorsteld, R_x de rotatiematrix rond de X-as en p' het resulterende punt. Om de volledige oriëntatie van een object te specifiëren bestaan er meerdere manieren, de meest gekende zijn de hoeken van Euler waarvoor verschillende conventies bestaan. Iedere conventie beschrijft een opeenvolging van rotaties rond drie verschillende assen. De gekozen volgorde van assen bepaalt de waarde van de hoeken. De twee meest gebruikte conventies zijn z-x-z en x-y-z.[5] In matrixvorm wordt dit als volgt voorgesteld:

$$p' = A.p = (R_x.R_y.R_z).p \tag{2.3}$$

$$p' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} . \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} . \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{2.4}$$

Ten slotte is er nog de schaling van objecten, deze wordt als volgt beschreven:

$$p' = S.p = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} . \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{2.5}$$

Hier stelt p een punt voor, S de verschalingsmatrix en p' het resulterende punt. Hierbij kan er een schaalfactor toegekend worden aan iedere hoofdas, dit laat toe om het object te vervormen over de hoofdassen. Translatie transformatiematrices worden standaard verwerkt met behulp van optelling. Rotatie/schaal transformatiematrices worden verwerkt met behulp van vermenigvuldiging. Om deze transformatiematrices op een homogene manier te verwerken introduceert men één extra dimensie, de schaalparameter 'h'.

[4]De matrixberekeningen, die hier beschreven zijn, gelden voor *column-major* matrices. In een column-major systeem wordt een vector in een kolommatrix opgeslagen. Bij een *row-major* systeem wordt dit een rijmatrix, dit draait de volgorde van vermenigvuldiging die hier beschreven is om.

[5]In *video games* wordt er meestal met het *Yaw-Pitch-Roll* concept gewerkt. Waarbij *yaw* de rotatie voorstelt rond de globale verticale as (meestal y-as in games). *Pitch* en *roll* worden dan meestal respectievelijk rotatie rond z-as en x-as.

Indien men de transformatiematrices homogeen maakt, kunnen alle bewerkingen met behulp van matrixvermenigvuldiging gerealiseerd worden. De combinatie van verschillende transformaties kan dus beschreven worden door een vermenigvulding van de individuele transformatiematrices.

$$p' = T.R.S.p = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} \quad (2.6)$$

Één van de eigenschappen van matrixvermenigvuldiging is associativiteit:

$$p' = T.R.(S.p) = T.(R.S).p = (T.R.S).p \quad (2.7)$$

Dit betekent dat alle transformaties vooraf kunnen vermenigvuldigd worden tot een complete transformatie beschreven door één 4x4 matrix.

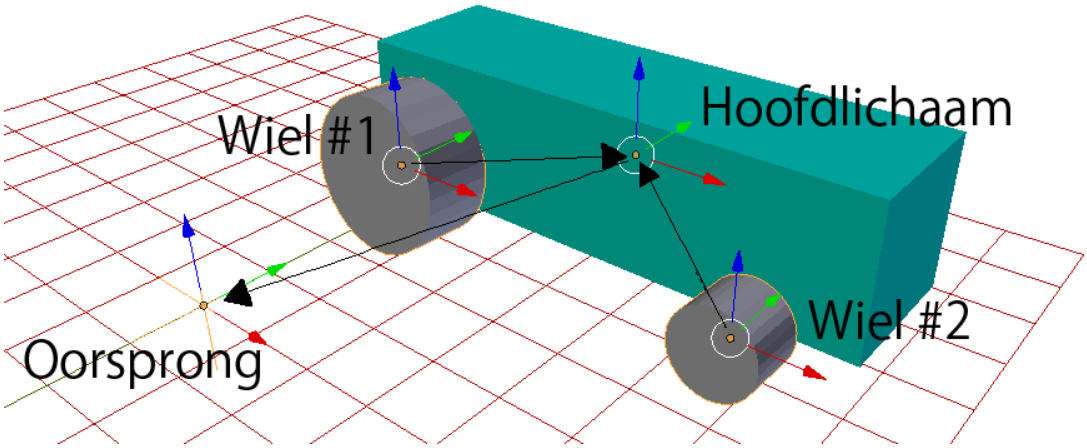
$$p' = (T.R.S).p = M_{transform}.p = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} . \begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} \quad (2.8)$$

Deze volgorde van translatie, rotatie en schaal is courant, maar uiteraard is het mogelijk om een andere volgorde van transformaties te gebruiken naargelang het beoogde effect.

2.2.2 Scène

Computer Graphics rendert[6] omgevingen met behulp van scènes, waarbij een scène is een collectie van objecten is. Hierin worden deze objecten beschreven a.d.h.v. hun plaats ten opzichte van een globaal gekozen coördinatenstelsel. Men spreekt bij het vergelijken van de vertices tot het globaal coördinatenstelsel van wereld coördinaten of *world space*. Deze wereld coördinaten worden gebruikt voor het renderproces en worden bekomen door de vertices beschreven in local space te vermenigvuldigen met de totale transformatiematrix van het object in world space. Dankzij homogene transformatiematrices kan men op een simpele manier objecten ook hiërarchisch gaan beschrijven. Stel dat men een auto wil beschrijven kan men dit doen door de auto op zijn beurt te gaan beschrijven a.d.h.v. wielen en een chassis.(zie figuur 2.3) De vertices van de wielen worden dan bekomen door de positie van de vertices in local space van het wiel te vermenigvuldigen met de transformatiematrix van het wiel in de local space van de auto en dit nogmaals te vermenigvuldigen met de transformatiematrix van het chassis in world space.

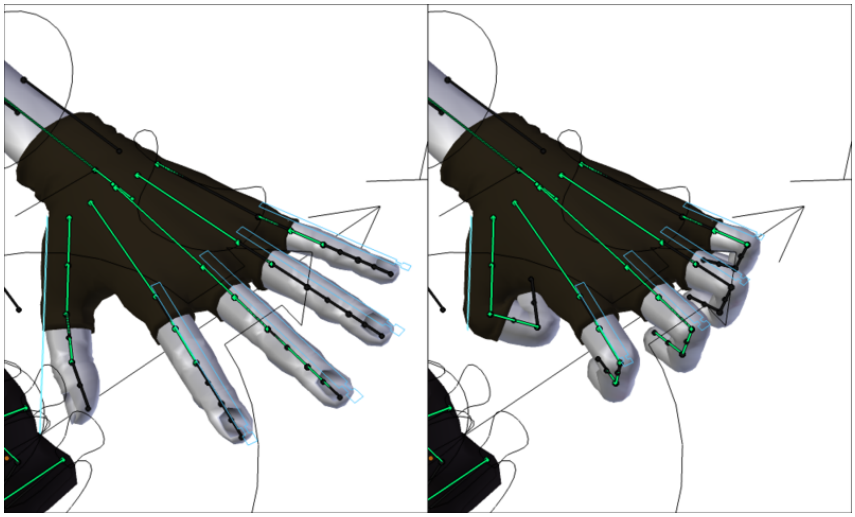
[6] *Renderen* is het genereren van een tweedimensionale weergave a.d.h.v. een driedimensionaal model om deze weer te kunnen geven op een beeldscherm.



Figuur 2.3: Hiërarchisch modelleren[7]

2.2.3 Skelet

Een skelet van een object of karakter is een hiërarchische beschrijving van gewrichten, ook wel *joints* genaamd. De verbindingen tussen twee joints worden botten of *bones* genoemd. Een startgewricht of *rootjoint* wordt geselecteerd, veelal tussen de heupen, en de naburige joints worden beschreven relatief t.o.v. van deze root. Vervolgens wordt iedere joint relatief t.o.v. zijn buur beschreven met een ouder-kind structuur, waarbij het kind in de local space van de ouder beschreven wordt. Hierbij ontstaat een boomstructuur van transformatiematrices. Door de transformatiematrix van een joint te manipuleren kan men de positie/oriëntatie van al zijn kinderen veranderen.(zie figuur 2.4)



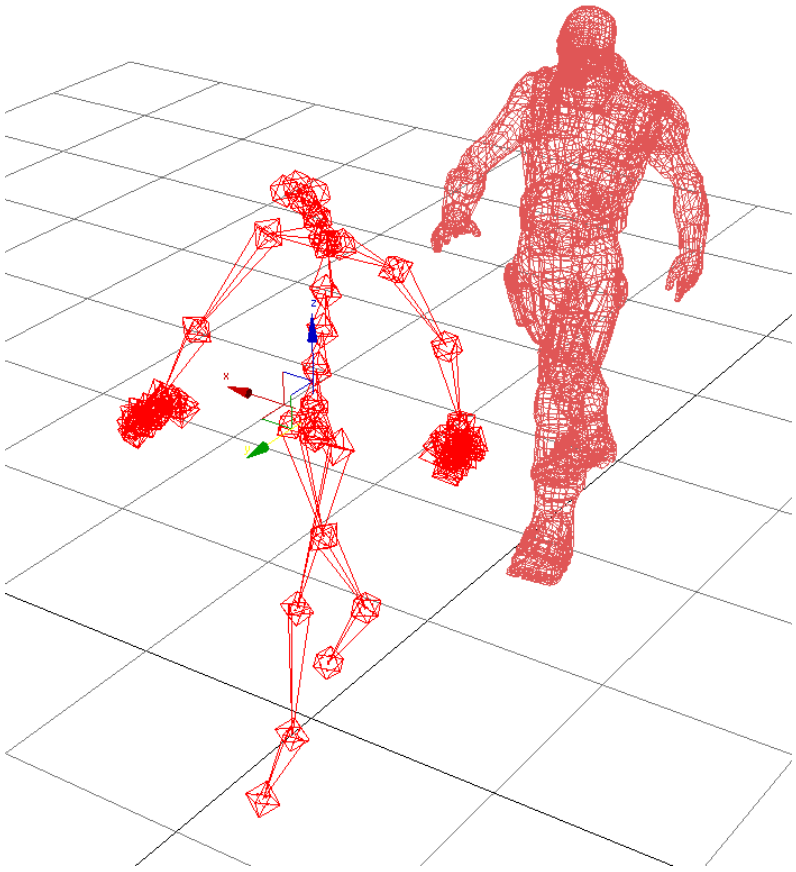
Figuur 2.4: Animeren van een hand met behulp van een skelet[8]

[7]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

De volledige transformatie van een joint is dus afhankelijk van de combinatie van transformaties van alle voorouders van de joint. Dit wordt bereikt door het opeenvolgend vermenigvuldigen van de transformatiematrices van het object met deze keten.

2.2.4 Skinning

Skinning is een techniek waarbij een visuele representatie gebonden wordt aan een skelet. Elke vertex van zo'n mesh wordt hiërarchisch verbonden aan een of meerdere joints, de sterkte van de afhankelijkheid tot die joint kan ingesteld worden met behulp van gewichten[9]. Dit bepaalt hoe sterk elke vertex meebeweegt met de joint. De volledige mesh van een karakter wordt op deze manier verbonden met een skelet, waarbij het karakter simpelweg kan geanimeerd worden door zijn skelet te manipuleren.(zie figuur 2.5)



Figuur 2.5: Een visuele representatie gebonden op een skelet[10]

[8]Van "Skeletal animation" beschikbaar op Wikipedia. Creative Commons Attribution-Share Alike 3.0 Unported license. http://en.wikipedia.org/wiki/Skeletal_animation

[9]De som van de gewichten voor een vertex moet gelijk zijn aan één.

[10]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

2.3 Animatie Algoritmen

2.3.1 Overzicht

Animatie kan in vier grote groepen onderverdeeld worden: traditionele animatie, procedurele animatie, parametrische animatie en motion capture. Bij traditionele animatie wordt de animatie volledig door de animator gecreëerd.

Bij procedurele animatie wordt de animatie gegenereerd op basis van een algoritme om een meer diverse reeks van acties te creëren. Bij parametrische animatie wordt een animatie opgebouwd uit twee of meer animaties. Bij motion capture repliceert de animatie bewegingen uit het echte leven. De sterktes en zwaktes van elke groep bepalen welke gebruikt wordt in een specifieke situatie. Zo kent traditionele animatie zijn toepassing binnen simpelere en meer algemene situaties. Procedurele animatie wordt veelal gebruikt bij simulaties waarbij fysisch realisme centraal staat, bvb. ragdoll physics waarbij een karakter op een realistische manier op de grond valt. Parametrische animatie wordt dan weer in hoofdzaak gebruikt om overgangen te creëren tussen verschillende animaties. Als laatste is motion capture een techniek die gebruikt wordt indien een hoge graad van realisme vereist is en hier dan ook budget voor is. Hieronder wordt een lijst weergegeven van de animatie algoritmen die besproken zullen worden.

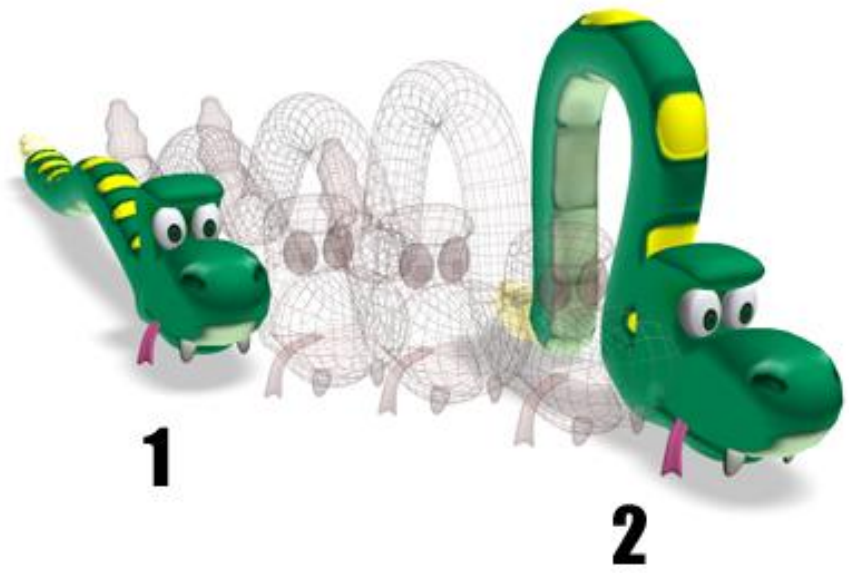
- Traditionele animatie
- Procedurele animatie
  - Inverse Kinematics (IK)
  - Physics animatie
  - Fuzzy animatie
- Parametrische animatie
  - Blending
  - Motion graphs
- Motion capture

2.3.2 Traditionele Animatie

Definitie

Traditionele animatie, ook wel keyframe animation genoemd, is een techniek waarbij een animator verschillende keyframes aanmaakt die aangevuld worden met inbetweens. Een animatie wordt weergegeven op basis van frames, in traditionele animatie wordt gesproken van twee types: keyframes en inbetweens, waarbij keyframes uiteenliggen doorheen de tijd en ertussen inbetweens geplaatst worden.

Inbetweens worden berekend op basis van interpolatie van de twee omliggende keyframes. Een animator maakt een ruwe schets van de animatie door een aantal keyframes te maken, het animatie programma creëert dan inbetweens op basis van de gecreëerde keyframes.(zie figuur 2.6)



Figuur 2.6: Keyframes bij posities 1 en 2. De computer genereert de inbetweens[11]

Evaluatie

De kwaliteit van de animatie is volledig afhankelijk van de vaardigheid van de animator. Een tekortkoming van deze techniek is de hoeveelheid werk die erin kruipt, voor iedere animatie moet een animator een set frames zien aan te maken. Hoe meer realisme gewenst hoe meer tijd er moet gespendeerd worden per keyframe. Dit schaalt relatief slecht met de vraag naar een grotere waaier aan animaties per karakter.

2.3.3 Physics animatie - Procedurele Animatie

Definitie

Physics animatie is animatie die gecreëerd wordt via een krachtenmodel bekomen door simulaties. In zijn meest generieke implementatie is physics animatie wat men *ragdoll physics* noemt.

[11]Van "Animation concepts". http://www.3dmax-tutorials.com/Animation_Concepts.html

Ragdoll physics wordt gebruikt als vervanging van traditionele val/sterf animaties. Het systeem waarbij de animatie op basis van een krachtenmodel wordt beïnvloed. Elk bot van het skelet wordt als een fysisch object gesimuleerd in een scène en de botten worden beperkt in hun vrijheidsgraden van beweging. Deze techniek zorgt ervoor dat een karakter op een realistische manier op de grond valt, rekening houdend met objecten in de omgeving (zoals muren, trappen en grondoppervlak).

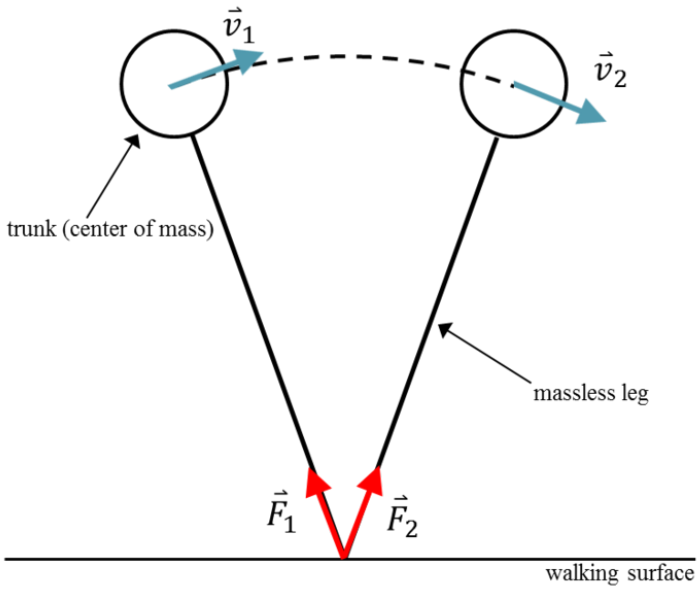
In figuur 2.7 wordt een voorbeeld getoond van een val geanimeerd voor een karakter met behulp van ragdoll physics.

Figuur 2.7: Ragdoll physics voorbeeld[12]

Evolutie

Recent onderzoek heeft een nieuwe techniek naar voor gebracht, de impuls gebaseerde animatie. [6] Het animatiemodel werkt op basis van richtposities, waarbij het model aan bepaalde voorwaarden moet voldoen. Voor een wandelmodel modeleert men het geheel naar een geïnverteerde pendel.(zie figuur 2.8)

[12]Deze afbeelding werd gemaakt door John Nagle van Animats in 1997, met behulp van Softimage. http://en.wikipedia.org/wiki/Ragdoll_physics



Figuur 2.8: Geïnverteerde pendel[13]

Men legt de restrictie op dat het bovenlichaam niet buiten bepaalde grenzen van rechtopstaand mag vallen. Het model wordt fysisch berekend en reageert dus ook op externe krachten.

Evaluatie

Dit is een experimenteel model waar nog onderzoek naar gedaan wordt, het aanleren van de restricties kost tijd (iteratief leerproces) maar levert na genoeg tijd fysisch realistische *gaits*[14]. Een groot voordeel is de reactie van het model op externe krachten.

2.3.4 Inverse Kinematics - Procedurele Animatie

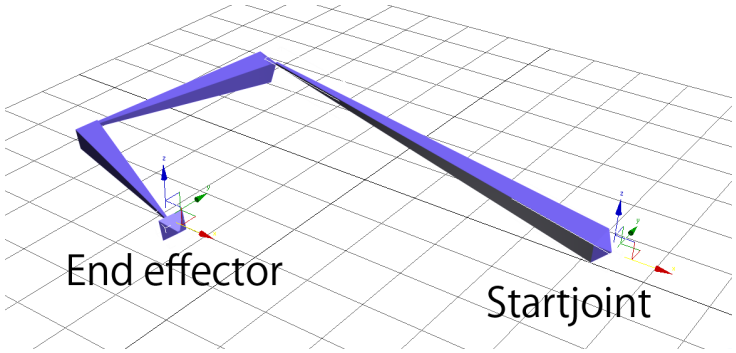
Definitie

Inverse Kinematics(IK) is een techniek die gebruikt wordt om de armen van karakters te besturen, origineel afkomstig uit de industrie waar men dit gebruikt om een robotarm aan te sturen. Waar men vertrekt van de hoeken van alle gewrichten om een eindpositie uit te komen bij *forward kinematics*, doet men het tegengestelde bij Inverse Kinematics. Men definieert een keten van gewrichten met *startjoint* en *end effector*[15] waarop het algoritme zal inwerken.(zie figuur 2.9)

[13] Van "Neuromechanics" beschikbaar op Wikipedia. Creative Commons Attribution-Share Alike 3.0 Unported license. http://en.wikipedia.org/wiki/Neuromechanics

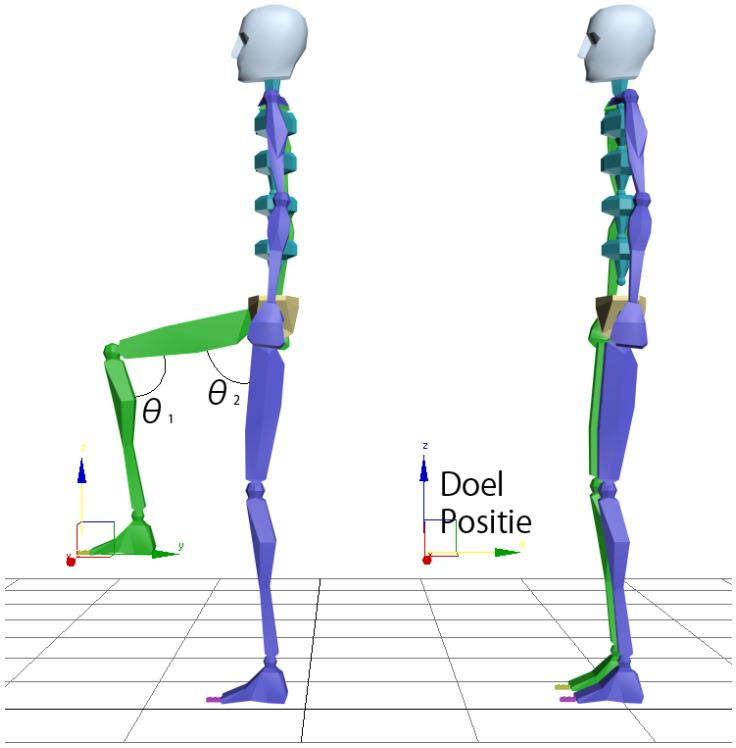
[14] *Gait* is de manier waarop een object zich voortbeweegt, voor mensen is dit de manier van lopen.

[15] *End effector*, een term uit de robotica die naar het einde van een robotarm refereert.



Figuur 2.9: Kinematische keten[16]

Vervolgens maakt men gebruik van kinematische vergelijkingen om de gewrichtparameters van deze keten te bepalen die de gewilde positie van de end effector opleveren. Bvb. de positie van de voet bepaalt de stand van het been.(zie figuur 2.10)



Figuur 2.10: Inverse Kinematics toegepast op een been van een karakter[17]

Stel de doelpositie e en θ als de set van m hoeken die deze positie opleveren.

$$e = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix} \tag{2.9}$$

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & ... & \theta_m \end{bmatrix} \tag{2.10}$$

[16]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[17]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Bij forward kinematics vertrekt men van de hoeken en bekomt men de eindpositie. Deze relatie wordt als volgt beschreven:

$$e = f(\theta) \tag{2.11}$$

De betrekking van Inverse Kinematics wordt dan zo beschreven:

$$\theta = f^{-1}(e) \tag{2.12}$$

Technieken

Er bestaan verschillende technieken om kinematische vergelijkingen uit te rekenen voor dit algoritme. Twee van de meer gekende zijn JacobianTranspose IK en Cyclic Coordinate Descent IK.

$$JacobianTranspose$$

JacobianTranspose IK [7] maakt gebruik van de Jacobiaan van een systeem om het probleem te vereenvoudigen. De Jacobiaan wordt beschreven als een matrix van de partieel afgeleiden van het systeem, en zegt dus hoe de end effector zal reageren op kleine hoekveranderingen.

$$J = \frac{de}{d\theta} = \begin{bmatrix} \frac{de_x}{d\theta_1} & \frac{de_x}{d\theta_2} & \cdots & \frac{de_x}{d\theta_m} \\ \frac{de_y}{d\theta_1} & \frac{de_y}{d\theta_2} & \cdots & \frac{de_y}{d\theta_m} \\ \frac{de_z}{d\theta_1} & \frac{de_z}{d\theta_2} & \cdots & \frac{de_z}{d\theta_m} \end{bmatrix} \tag{2.13}$$

Men kan de Jacobiaan herschrijven, zodat men de hoeken kan vinden op basis van de end effector positie.

$$J = \frac{de}{d\theta} \tag{2.14}$$

$$de = Jd\theta \tag{2.15}$$

$$d\theta = J^{-1}de \tag{2.16}$$

Indien de inverse Jacobiaan gekend is, kan men dus de hoeken bepalen die vereist zijn voor een gegeven end effector positie. Aangezien dit systeem niet-lineair is, is deze betrekking slechts geldig voor kleine variaties van positie en hoek. De inverse Jacobiaan moet dus iteratief uitgerekend en toegepast worden indien men het systeem wil oplossen. Het algoritme maakt ook gebruik van een α parameter, deze parameter werkt direct in op de snelheid waarmee geroteerd wordt. Wordt deze parameter te hoog ingesteld dan is het systeem instabiel, wordt de parameter te laag ingesteld dan convergeert het te traag naar de oplossing.

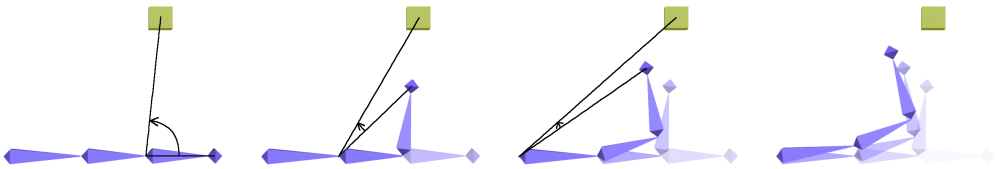
De inverse Jacobiaan bepalen is een rekensintensief proces, maar men kan aantonen dat voor kleine variaties in positie van de end effector de inverse Jacobiaan de getransponeerde Jacobiaan benadert. (Hier haalt JacobianTranspose zijn naam vandaan.)

$$J^{-1} \simeq J^T \tag{2.17}$$

CCD

Cyclical Coordinate Descent of CCD [8] is een simpele en efficiënte techniek om Inverse Kinematics oplossingen mee te genereren. Het is een iteratief proces, waarbij elke iteratie de end effector iets dichter bij de doelpositie brengt. CCD zal convergeren naar de doelpositie maar die nooit bereiken, er wordt dus een limiet geplaatst op het aantal iteraties en de afwijking van de positie.

Eén iteratie van CCD bestaat op zijn beurt ook uit iteraties, er wordt geïtereerd over alle joints vertrekkende van de joint net voor de end effector tot de startjoint. Elke joint bepaalt de afwijking in hoek tussen de end effector en de doelpositie en gebruikt dit om zijn rotatie te corrigeren. In figuur 2.11 ziet men het verloop van een buitenste iteratie.



Figuur 2.11: CCD: 1 iteratie [18]

Evolutie

Inverse Kinematics is één van de actievere regio's van onderzoek binnen computeranimatie. Een paar vooruitgangen die zijn gemaakt zijn de volgende:

- Collision Avoidance [9], waarbij botsingen tussen de delen van het lichaam vermeden worden
- Example Guidance [10], optimaliseert de keuze van eindoriëntaties door te vergelijken met een voorbeeld animatie
- Stylization [11], waarbij het systeem bepaalde standen prefereert door gebruik te maken van waarschijnlijkheidsverdelingen
- Neural Network usage [12], er worden neurale netwerken voorgesteld als oplossing voor Inverse Kinematics

[18] Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Evaluatie

IK is een techniek die gewrichtsparameters voor een doelpositie bepaalt op een bepaald ogenblik. Dit op zich volstaat niet om een animatie mee op te bouwen. Om met IK animaties te creëren moet men een doelpositie animeren over een pad. Naar deze paden wordt soms gerefereerd als *animatiepad*. Hierbij zal er voor elke frame de doelpositie opgevraagd worden en het IK algoritme bepaalt dan de gewrichtsparameters voor die doelpositie op dat specifieke moment. Het object dat gebruik maakt van een IK algoritme om de gewrichtsparameters te bepalen noemt men binnen Computer Graphics een *Inverse Kinematic solver* of *IK solver*. Door gebruik te maken van IK kan men karakters verbinden met hun fysieke omgeving. Hoewel IK de animatie in eerste opzicht er realistisch zal doen uitzien, zal men deze als (licht) robotisch ervaren. Dit is te wijten aan de sterk wiskundige aspecten van dit algoritme en de focus van het algoritme op positie i.p.v. snelheid of acceleratie. Men kan echter door middel van constraints of beperkingen te plaatsen op de joints vaak beter resultaten bekomen. Dit zal nooit de kwaliteit opleveren van bvb. traditionele animatie, maar dit wordt verwaarloosd omwille van de mogelijkheden die deze techniek toelaat. Om toch een betere graad van realisme te behalen wordt vaak gebruikt gemaakt van blending (zie sectie 2.3.6), waarbij een IK animatie wordt samengevoegd met een niet-procedurele animatie.

Fullbody IK

Aangezien IK werkt op enkelvoudige ketens is dit niet direct toepasbaar op het volledige lichaam van een karakter aangezien deze splitsingen heeft in zijn skelet. Men lost dit op door verschillende IK-solvers toe te passen op verschillende delen van het lichaam en blending toe te passen om alle afzonderlijke animaties samen te voegen.

2.3.5 Fuzzy animatie - Procedurele Animatie

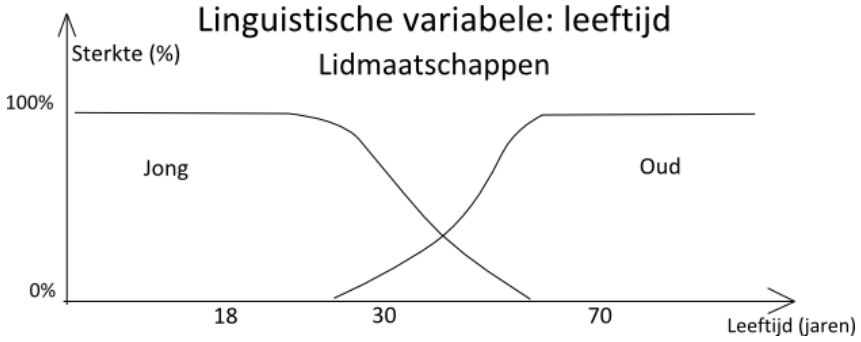
Definitie

Fuzzy animatie is het toepassen van een Fuzzy Logic controller op animatie. Fuzzy Logic is een regeltechniek die ontstond in de jaren '60. [13]

Fuzzy Logic handelt in logica die eerder benaderend dan exact is. Het maakt gebruik van partiële waarheid, wat zelfs linguistische variabelen toelaat. De resulterende regeling maakt een gewogen keuze op basis van een groep input data. Fuzzy Logic heeft minder last van instabiliteit in vergelijking met een gewone PID-controller[19], en laat ook toe om breder scala aan processen te beschrijven dan voorheen mogelijk was met PID-controllers.

[19]Een PID-controller is de meest voorkomende regelaar bij (industriële) procesregeling.

Fuzzy Logic werkt met geschreven regels die gebruik maken van linguistische variabelen. Een linguistische variabele zoals bijvoorbeeld *leeftijd* kan een waarde hebben van *jong* of *oud*. Binnen Fuzzy Logic noemt men de waarden van linguistische variabelen lidmaatschappen. Een lidmaatschap beschrijft een gradatie of sterkte van 'waar zijn' voor verschillende numerieke waarden. Voor een lidmaatschap *jong* zou bijvoorbeeld een leeftijd van 15 jaar 100% waar zijn, een leeftijd van 30 jaar 50% waar en een leeftijd van 70 jaar 0% waar. In figuur 2.12 wordt dit grafisch weergegeven.



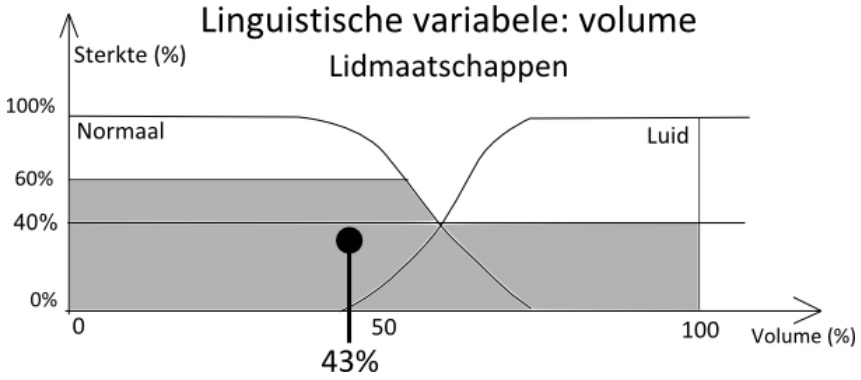
Figuur 2.12: Voorbeeld linguistische variabele[20]

Indien men nu bijvoorbeeld een Fuzzy Logic controller zou willen ontwerpen die het volume luider of stiller zet van een toestel afhankelijk van de ouderdom van de gebruiker. Men ontwerpt een linguistische variabele *volume* met lidmaatschappen *normaal* en *luid*. De Fuzzy Logic controller wordt dan ingesteld met volgende regels:

```
ALS leeftijd IS oud DAN volume IS luid
ALS leeftijd IS jong DAN volume IS normaal
```

Indien er nu een numerieke leeftijd ingegeven wordt in de controller zal deze ook een fuzzy antwoord geven, in termen van sterkte van elke lidmaatschap. Voor dit voorbeeld zou de volume 60% *normaal* en 40% *luid* kunnen zijn. Aangezien we werken met een numerieke ingangswaarde, zijn numerieke uitgangswaarden ook gewenst. Dit maakt het gemakkelijker om er praktisch mee om te gaan zoals bijvoorbeeld het effectieve geluidsniveau instellen van de luidsprekers. Deze laatste stap om van een fuzzy uitgang naar een numerieke waarde te gaan is wat men *defuzzificatie* noemt. De implementatie hiervan is afhankelijk van wat voor type gedrag gewenst is. Eén van de meer accurate methoden is wat men een *centroid defuzzifier* noemt. Hierbij worden de sterktes van de lidmaatschappen respectievelijk begrensd op het niveau van iedere uitgangssterkte. Van het overgebleven oppervlak wordt het zwaartepunt bepaald en de horizontale positie van dit punt wordt gebruikt als numerieke uitgangswaarde. Dit wordt grafisch weergegeven in figuur 2.13.

[20]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 2.13: Voorbeeld defuzzificatie[21]

Voor dit voorbeeld zou dit dus een volume opleveren van 43%.

Evaluatie

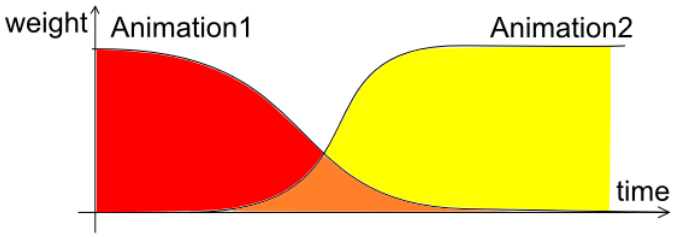
Om met Fuzzy Logic te animeren, moet men animaties proberen numeriek voor te stellen. Een animatie van een karakter bestaat uit het animeren van de verschillende gewrichten van dat karakter. De oriëntatie van ieder gewricht, die meedoet aan de animatie, is een functie van de tijd. Men kan deze oriëntatie voorstellen als een groep numerieke waarden door deze op te splitsen in hoeken van Euler. Deze waarden kunnen gebruikt worden in Fuzzy Logic controllers om nieuwe hoeken mee te bepalen volgens ingestelde regels.De implementatie van deze controllers wordt volledig overgelaten aan de programmeur, met mogelijke ondersteuning van animatoren. De implementatie wordt dus uitgewerkt per situatie, dit is tegelijkertijd zijn grootste kracht en zwakheid. In de toekomst zou er nog onderzoek gedaan kunnen worden omtrent het veralgemenen en hergebruiken van deze controllers om deze problematiek aan te pakken.

2.3.6 Blending - Parametrische Animatie

Definitie

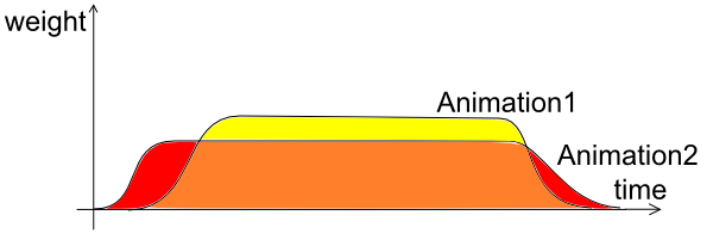
Blending produceert nieuwe animaties door meerdere animaties samen te voegen met tijd-variërende gewichten. Een toepassing van blending is het creëren van overgangen tussen verschillende animaties zodat deze natuurlijker overkomen. In figuur 2.14 ziet men het verloop van de gewichten van twee animaties bij een overgang.

[21]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 2.14: Blending: Overgang tussen animaties[22]

Een andere toepassing is interpolatie, waarbij meerdere animaties gelijktijdig worden samengevoegd. Hierbij kunnen volledige animaties gecombineerd worden om nieuwe te creëren, zoals de combinatie van een wandelanimatie op een helling van 30 graden met één van 20 graden om een wandelanimatie op een helling van 25 graden te verkrijgen. (zie figuur 2.15).



Figuur 2.15: Blending: Interpolatie van animaties[23]

Alternatief kan ook elke animatie slechts een specifiek deel van het skelet beïnvloeden. Men kan bijvoorbeeld een schietanimatie met een wandelanimatie blenden, waarbij de schietanimatie het bovenlichaam beïnvloedt en de wandelanimatie het onderlichaam.

Evolutie

Onderzoek [14] naar optimalisaties bij *automatic blending* heeft *registration curves* opgeleverd. Een registration curve bestaat uit een *timewarp curve*,*coordinate alignment curve* en een set van *constraint matches*. De timewarp curve probeert de snelheid van gelijkaardige animaties te synchroniseren. De coordinate alignment curve probeert richtingen gelijk te stellen. Indien animaties gelijkaardig zijn maar verschillende richtingen uitgaan kan de resulterende blending ongewenste effecten vertonen. De constraint matches zorgen ervoor dat bij blending de gegeven *constraints* of beperkingen voor iedere animatie

[22]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[23]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

voldaan blijven. Deze curves kunnen gebruikt worden om automatic blending te verbeteren waardoor blending kan toegepast worden op een groter gamma aan animaties zonder interventie van de animator.

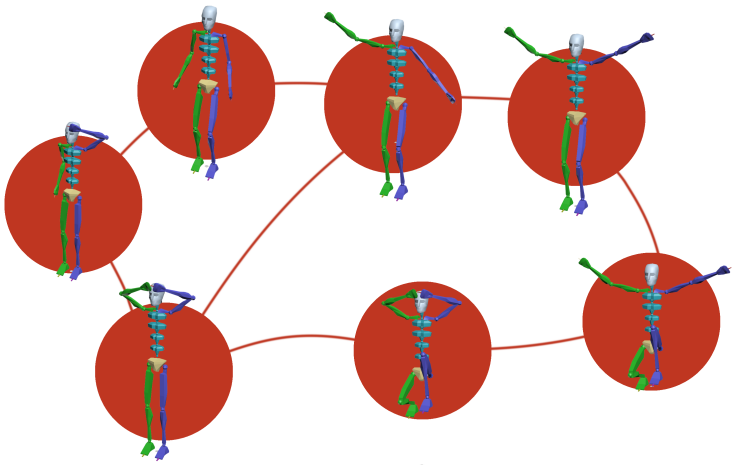
Evaluatie

Blending kan men niet alleen gebruiken bij animatietechnieken van eenzelfde type maar ook bij animatietechnieken van verschillende types. Eén zo'n voorbeeld is het blenden van traditionele animatie met procedurele animatie. Zo kan men het hoge realisme van traditionele animatie combineren met de dynamische eigenschappen van animatiecreatie bij procedurele technieken.

2.3.7 Motion graphs - Parametrische Animatie

Definitie

Motion graphs is een speciale techniek waarbij animaties dynamisch in elkaar overvloeien. [15] De techniek laat toe om in real-time de animatie te laten beslissen wat de volgende animatie wordt en op welk moment er kan overgevloeid worden. Men vertrekt van een lijst met animaties die men opsplitst a.d.h.v. (key)frames. Vervolgens creëert men een graaf[24] om deze frames in op te slaan. Frames die op elkaar lijken binnen een tolerantie worden als gelijk geacht. Zo creëert men een ongerichte graaf van frames, een animatie wordt vervolgens beschreven door een lijst van knooppunten, elke animatie is dus een wandeling over de graaf. In figuur 2.16 wordt een motion graph conceptueel voorgesteld.



Figuur 2.16: Voorbeeld van een motion graph[25]

[24]Een *graaf* is een verzameling van punten, knopen genoemd, waarvan sommige onderling verbonden zijn door lijnen.

[25]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

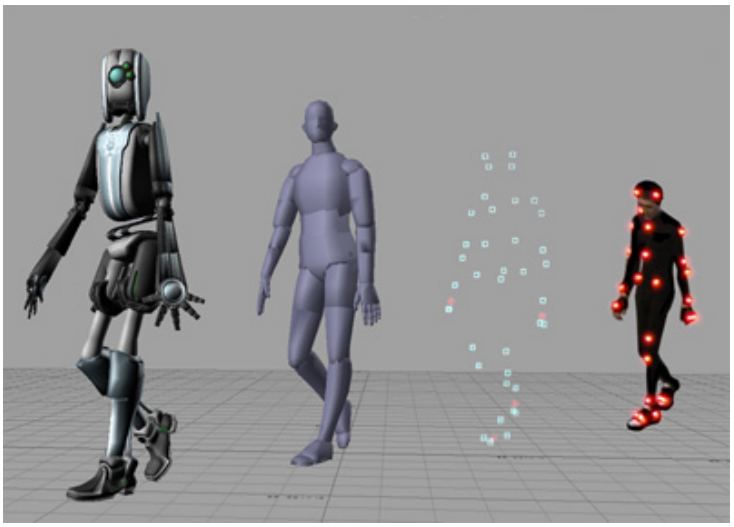
Evaluatie

Met deze methode kan men een groot aantal animaties opslaan met heel weinig geheugen. De frames worden gedeeld door alle animaties dus moeten ze slechts één maal opgeslagen worden, vervolgens wordt elke animatie als een lijst van knooppunt-indexen opgeslagen. Dit betekent dat voor een kleine set animaties niet veel winst wordt gemaakt, maar naarmate men meer animaties heeft kan dit betekenisvolle winsten opleveren.

2.3.8 Motion capture

Definitie

Motion capture is een techniek die gebruikt wordt waar hoog realisme gewenst is. Acteurs trekken een pak aan die bedekt is met markers, deze markers worden gedetecteerd door een groep camera's die in de kamer zijn aangebracht. De markers kunnen actief of passief zijn, actieve markers geven hun eigen licht af en passieve markers zijn met een retroreflectieve[26] laag bedekt. Vervolgens speelt de acteur een scene, en de camera's detecteren de posities van alle markers, deze data wordt vervolgens gebruikt om het karakter te animeren.(zie figuur 2.17) Dit is gelijkaardig aan traditionele animatie, maar bij motion capture zijn de transformaties voor alle joints op elk tijdstip gekend.



Figuur 2.17: Motion capture[27]

[26]Een retroreflector is een oppervlak dat licht terugkaatst naar de bron met een minimum aan verstrooiing.

[27]Van "Motion capture" beschikbaar op Wikipedia. (Public domain) http://en.wikipedia.org/wiki/Motion_capture

Evaluatie

Deze animatietechniek levert realistische animatie maar komt met een hoge kost. Men moet een studio aankopen of huren om de data op te nemen en ook de acteurs inhuren. Anderzijds moet er nog steeds een animator aanwezig zijn om de animatie wat bij te sturen, want de bekomen data zal nooit perfect zijn. Een groot nadeel is het statische karakter, men kan een opgenomen animatie maar voor een klein deel bijsturen. Meestal moet men de volledige animatie heropnemen bij fouten.

2.4 Multi-karakter animatie

Multi-karakter animatie is de animatie van interactie tussen verscheidene karakters. Om een accuratere weergave van de realiteit te bereiken, moet het karakter zo sterk mogelijk de verschillende aspecten van een echte mens representeren, we spreken van *virtual humans*. De verschillende aspecten die gepaard gaan met interactie moeten hierbij ook vastgelegd en besproken worden.

2.4.1 Virtual Humans

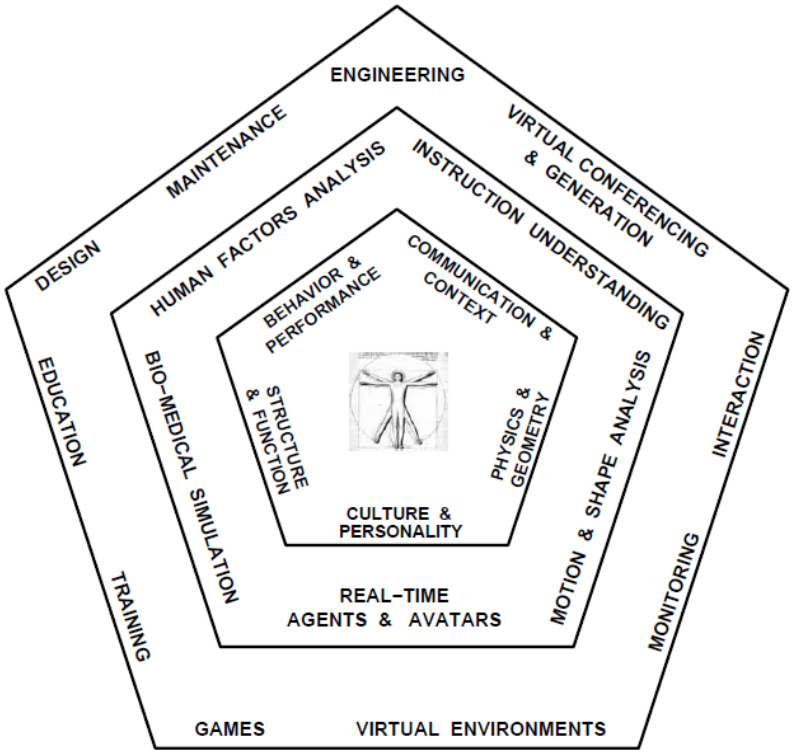
Virtual humans is een ruim onderzoeksdomein, de definitie die hier beschreven wordt is afgeleid uit onderzoek door Norman Badler. [16]

Definitie

Binnen Computer Graphics worden virtual humans beschreven als computermodellen van mensen. Deze kunnen gebruikt worden voor ergonomische testen bij computer-gebaseerde ontwerpen van bvb. auto's, werkplaatsen, machines, etc. Een ander toepassingsveld is het plaatsen van real-time representaties van onszelf of anderen in virtuele omgevingen. Bij het definiëren van virtual humans doorkruist men verscheidene domeinen zoals:

- Human Factors analysis: menselijke mogelijkheden, gedrag,...
- Real-time agents and avatars: diversiteit in persoonlijkheid en cultuur
- Biomedical simulation: fysiologische reactie, verwondingen, ...
- etc.

In figuur 2.18 ziet men een overzicht van de verschillende domeinen en aspecten van virtual humans.



Figuur 2.18: Virtual human domeinen/aspecten[28]

Bij het bouwen van virtual human modellen zijn er verschillende noties van getrouwheid, deze zijn uiteraard applicatie afhankelijk. Men kan getrouwheid wensen op vlak van tijd, grootte, fysieke mogelijkheden, etc. Deze aspecten zijn nooit zwart-wit, dit heeft Norman Badler geleid tot de beschrijving van 5 dimensies om een virtual human in te beschrijven.

1. Uiterlijk: Graad van realistisch uiterlijk (foto-realisme)
2. Functie: Menselijke mogelijkheden/limitaties nabootsen
3. Tijd: Real-time genereren van acties
4. Autonomie: Intelligentie, virtual human selecteert zelf animaties
5. Individualiteit: Personificatie van de animaties

Men kan verschillende karakters nu gaan definiëren/onderscheiden aan de hand van deze dimensies.

[28] Afbeelding uit "Virtual Humans Broad Perspective" door Norman Badler. [16]

2.4.2 Interactie

In deze sectie wordt een classificatie voorgesteld voor menselijke interactie.

Een eerste indeling kan gemaakt worden op basis van aantallen, meer specifieke animaties zoals handen schudden en schouderklop worden toegepast op 2 karakters. Animaties zoals tegen elkaar lopen en reageren kunnen op een veel breder aantal worden toegepast, twee of meer, dit wordt ook vaak "crowd animation"genoemd. Aangezien het grootste deel aan animatie kan gesimuleerd worden met twee karakters zal de focus van dit onderzoek hierop vallen. Een tweede indeling kan worden gemaakt op basis van de intentie van het karakter om mee te doen aan de animatie. We stellen dat een karakter volledige intentie heeft (initiator) of geen intentie (acceptor). Dit levert drie modellen van animatie voor twee karakters zoals te zien is in tabel 2.1.

Type	Voorbeelden
Initiator - Initiator	Handen schudden, High five
Initiator - Acceptor	Schouderklop
Acceptor - Acceptor	Botsen (Crowd animation)

Tabel 2.1: Drie modellen voor animatie tussen twee of meer karakters

Een bijkomende eigenschap is de personificatie van de animaties, dit kan slechts bij initiators voorkomen. De persoonlijkheid van het karakter beïnvloedt de animatie, als men het schudden van handen bekijkt speelt assertiviteit een grote rol in de houding van de hand. [17] Een persoon die heel assertief is zal doorgaans zijn handpalm naar de grond doen wijzen terwijl een verlegen persoon zijn handpalm eerder omhoog doet wijzen. De implementatie van personificatie zal voor de gekozen techniek(en) onderzocht(/vergeleken) worden.

Hoofdstuk 3

Probleem definitie

Het animeren van interacties is altijd problematisch geweest en wordt nog vaak vermeden tenzij er budget voor aanwezig is. Werk naar procedurele animaties zou hiervoor een antwoord kunnen bieden. Procedurele animaties worden veelal gebruikt om animaties te genereren die aangepast zijn aan hun omgeving, zoals een simpele wandelanimatie die dynamisch aangepast is aan de vloer om zo een realistischer effect te creëren. Dit dynamisch aspect van procedurele animatie kan gebruikt worden om *interactieanimaties* op een realistische manier uit te werken, waarbij elk karakter zijn handelingen aanpast aan de eigenschappen en dimensies van het andere karakter.

3.1 Het concept

Elke interactieanimatie heeft zo zijn eigenschappen en vereisten of het nu een handdruk, een knuffel of simpelweg zwaaien naar elkaar betreft. Interactieanimaties bestaan in grote variëteit en aantal. Om dit brede probleem aan te pakken wordt ervoor gekozen om één specifieke case uit te werken en hieruit te onderzoeken hoe een framework voor interactieanimaties gecreëerd kan worden. Bij het onderzoeken van een animatie zijn volgende criteria belangrijk:

1. Menselijkheid
2. Fysieke accuraatheid
3. Personificatie aspecten

In dit onderzoek wordt er vertrokken van een casestudy over de handdruk, in het volgende hoofdstuk wordt dit naar voor gebracht. Vervolgens wordt een virtueel model van een handdruk geïmplementeerd op twee manieren: fuzzy animatie en Inverse Kinematics. Er wordt een vergelijkende studie gemaakt van beide technieken.

Hoofdstuk 4

Casestudy

Om een virtueel model van een handdruk te creëren werd er eerst een casestudy gemaakt van een handdruk. In deze studie werd er vertrokken van gefilmd beeldmateriaal van een aantal handdrukken. De steekproef betreft drie verschillende handdrukken waarvan elk vijf opnames gemaakt werden. Het doel van dit onderzoek is niet het creëren van de perfecte handdruk dus wordt dit als voldoende aanzien om een simpel virtueel model te maken. De vijf opnames zijn gemaakt om de fouten ten gevolge van uitschieters te vermijden, er moet ook bemerkt worden dat alle proefpersonen zich bewust waren van de opname en dit allicht invloed zal hebben op de uitvoering van de animatie. Ten slotte werd er nog een specifieke analyse gemaakt van de handgreep tijdens het schudden zelf.

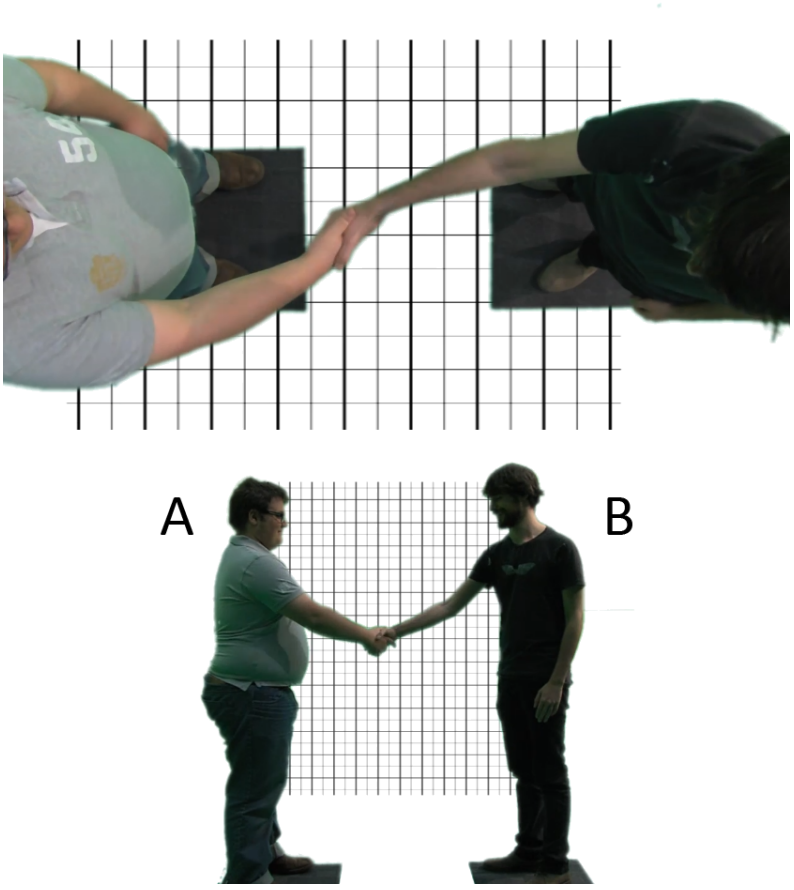
4.1 Vooraf

Deze sectie licht kort de structuur voor elk van de drie cases toe. Zoals in de afbeeldingen van de proefopstellingen duidelijk zal worden werd er telkens een zij- en bovenaanzicht genomen van de handdruk. Voor de analyse zal specifiek het schudden zelf besproken worden, m.a.w. de tijd waarin beide handen in contact zijn met elkaar. Dit wordt de schud- of *shakefase* genoemd. De analyse gebeurt in drie stappen: analyse van de volledige shakefase, analyse van het bovenaanzicht (voor o.a. het initieel contactpunt te bekijken) en ten slotte analyse van het zijaanzicht om de schudrichting te bepalen.

4.2 Case 1: Twee mannelijke proefpersonen

4.2.1 Opstelling

De eerste case is een opname van twee mannelijke proefpersonen. Proefpersoon A (links) is iets kleiner dan proefpersoon B (rechts), een klein verschil maar desalniettemin aanwezig. In figuur 4.1 ziet men de proefopstelling.



Figuur 4.1: Proefopstelling Case 1[1]

4.2.2 Analyse

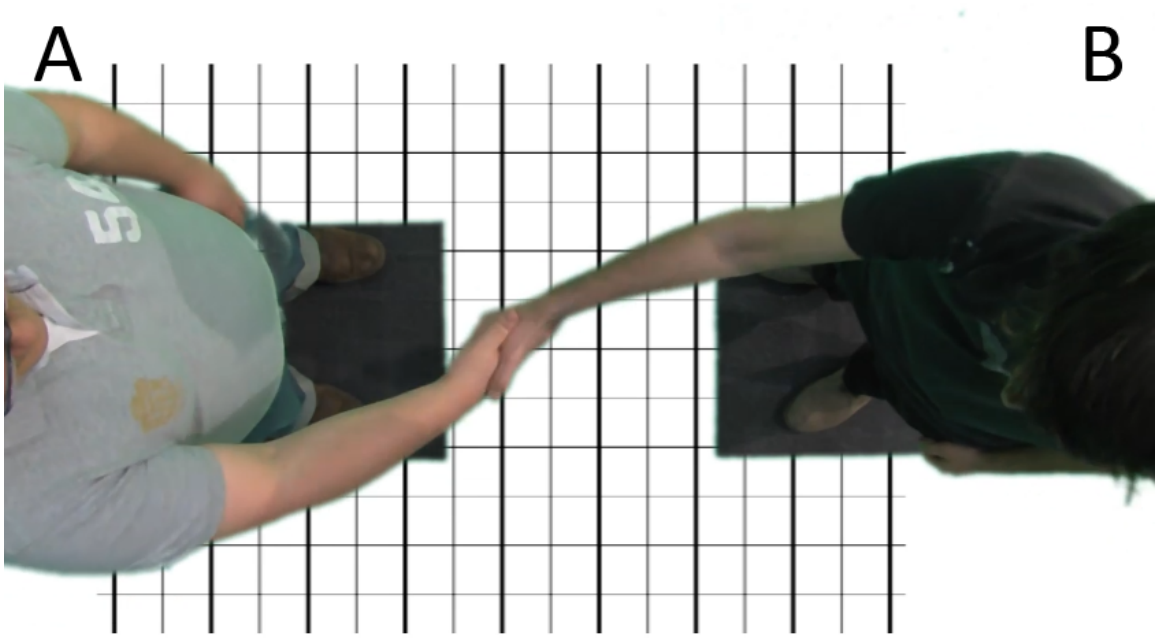
In deze case gaan de handen tijdens het schudden slechts éénmaal omhoog en naar beneden waarna beide handen terugkeren naar hun startpositie. Deze uitwijking is beperkt in hoogte en gebeurt aan een constante snelheid.

Bovenaanzicht

De locatie van de handen blijft in lijn met de twee ellebogen, of m.a.w. de locatie van de handen blijft binnen het vlak gecreëerd door de ellebogen en de startpositie van de

[1]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

handdruk. De startpositie bevindt zich dichter bij proefpersoon A. (Zie figuur 4.2)



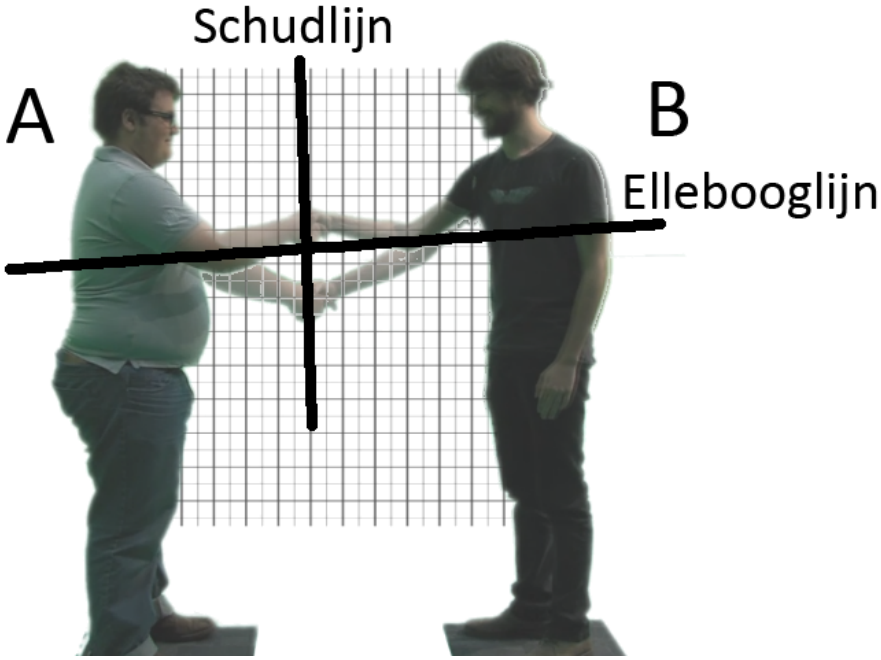
Figuur 4.2: Bovenaanzicht Case 1[2]

Aangezien beide proefpersonen ongeveer even lange armen hebben, steekt proefpersoon B zijn arm dus verder uit. Dit zou kunnen zijn omdat proefpersoon A de meer dominante persoon is in deze situatie.

Zijaanzicht

Er wordt verwacht dat de richting van de volledige shakefase loodrecht op de vloer staat, onder andere door de symmetrische aspecten van een handdruk. Maar er is een vermoeden dat lengteverschil tussen proefpersonen een hoekverschuiving zal introduceren. De hoek tussen de schudlijn en de vloer wordt bepaald en ook de hoek t.o.v. de ellebooglijn van de proefpersonen. In figuur 4.3 ziet men de meetopstelling.

[2]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



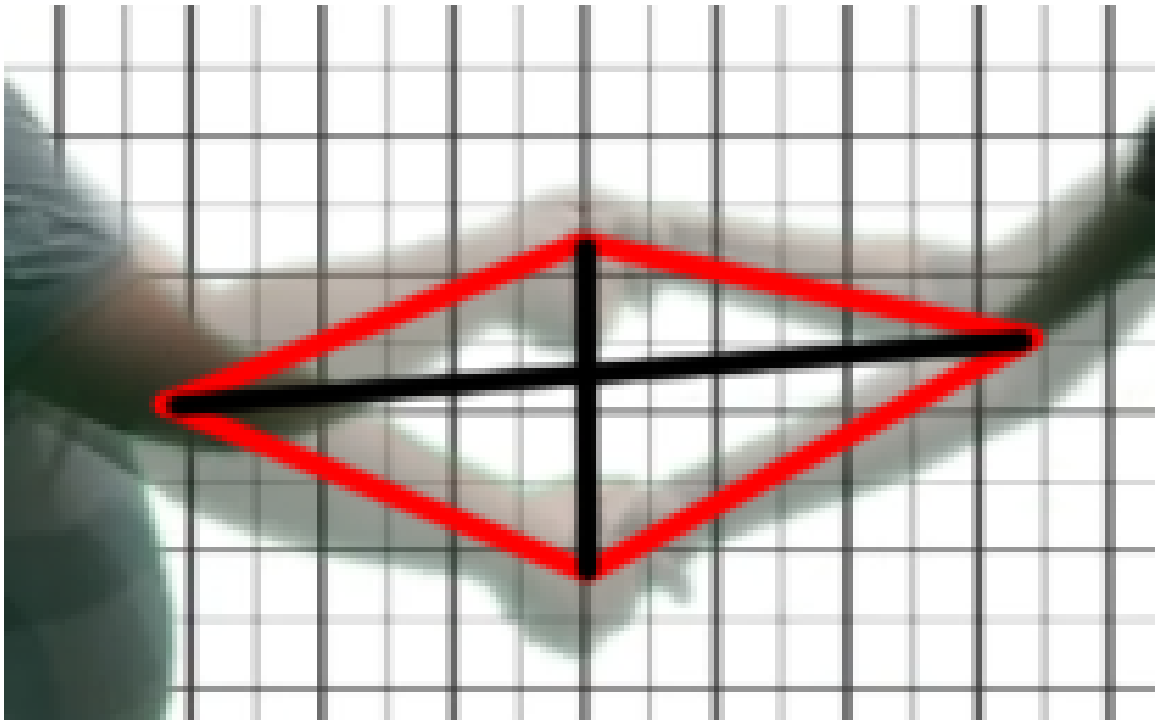
Figuur 4.3: Hoekmeting Case 1[3]

	Hoek ellebooglijn-vloer	Hoek schudlijn-vloer	Hoek ellebooglijn-schud
HS_MM_01	4,4	97,1	92,7
HS_MM_02	2,4	90,0	87,6
HS_MM_03	4,6	92,8	88,2
HS_MM_04	2,3	90,0	87,7
HS_MM_05	2,2	92,8	90,6
Case 1 gemiddelde	3,2	92,5	89,4

Tabel 4.1: Hoekanalyse Case 1

Uit tabel 4.1 kan men enerzijds de min of meer loodrechte stand tussen de ellebooglijn en de schudlijn waarnemen. Anderzijds is er de lichte hoekverdraaiing van dit 'assenkruis' t.o.v. van de vloer. Indien men de hoekverdraaiing van dit geheel volledig toekent aan het lengteverschil tussen beide proefpersonen dan kan de loodrechte hoek tussen beide lijnen aanzien worden als een standaardtoestand. Verder onderzoek zal uitwijzen of dit correct is. Vervolgens wordt er gekeken naar de afstand die tijdens de shakefase wordt afgelegd.

[3]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 4.4: Shakefase Case 1[4]

Uit figuur 4.4 wordt de hoogte van de shakefase bepaald. Bekeken van opzij is deze ongeveer de helft van de afstand tussen beide ellebogen. Om de werkelijke verhouding te bekomen moet er nog compenserende factor toegevoegd worden die de rotatie in het bovenaanzicht in rekening brengt. Uit figuur 4.2 haalt men een hoek van ongeveer 30 graden voor de lijn tussen de ellebogen en het projectievlak. Dit betekent dat de effectieve hoogte van de shakefase zich verhoudt tot de afstand tussen beide ellebogen volgens volgende betrekking:

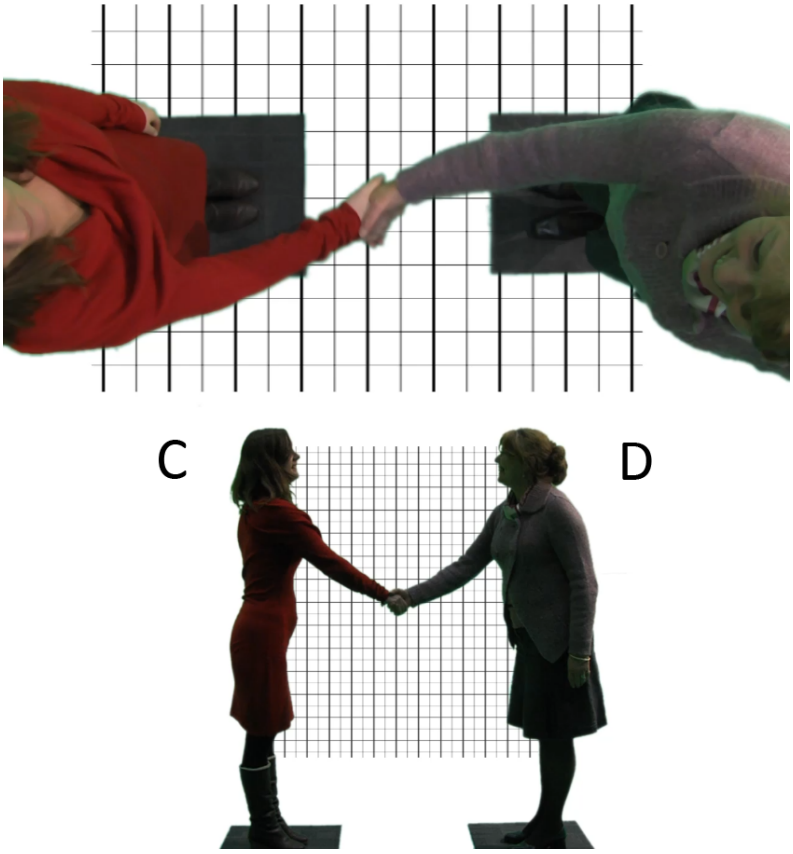
$$hoogte\ shakefase = \frac{\sqrt{2} * afstand\ tussen\ ellebogen}{4} \tag{4.1}$$

[4]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

4.3 Case 2: Twee vrouwelijke proefpersonen

4.3.1 Opstelling

De tweede case is een opname met twee vrouwelijke proefpersonen. Proefpersonen C en D hebben ongeveer dezelfde lichaamslengte. In figuur 4.5 ziet men de proefopstelling.



Figuur 4.5: Proefopstelling Case 2[5]

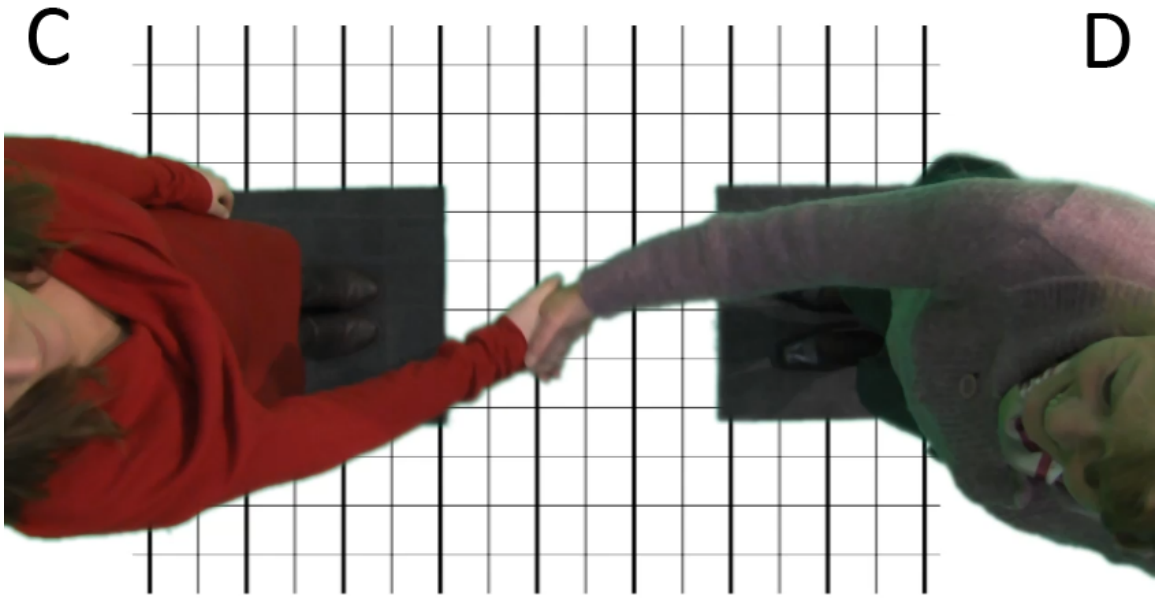
4.3.2 Analyse

Deze handdruk start met één grote uitwijking opwaarts. Nadat de handdruk weer neerwaarts is gegaan, volgen een aantal kleinere oscillaties in variërende richting. De opwaartse beweging is de meest significante en komt vrij goed overeen met de beweging van de vorige case, dus deze wordt gebruikt in de verdere analyse. De kleinere bewegingen worden hierbij verwaarloosd voor het verdere ontwerp van het raamwerk, maar worden geplaatst als een aspect van toekomstig werk. Bij verdere analyse wordt enkel naar de eerste uitwijking gekeken.

[5]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Bovenaanzicht

De locatie van de handen blijft opnieuw in lijn met de twee ellebogen. De startpositie bevindt zich bijna in het midden, net iets dichter bij proefpersoon C. (Zie figuur 4.6)

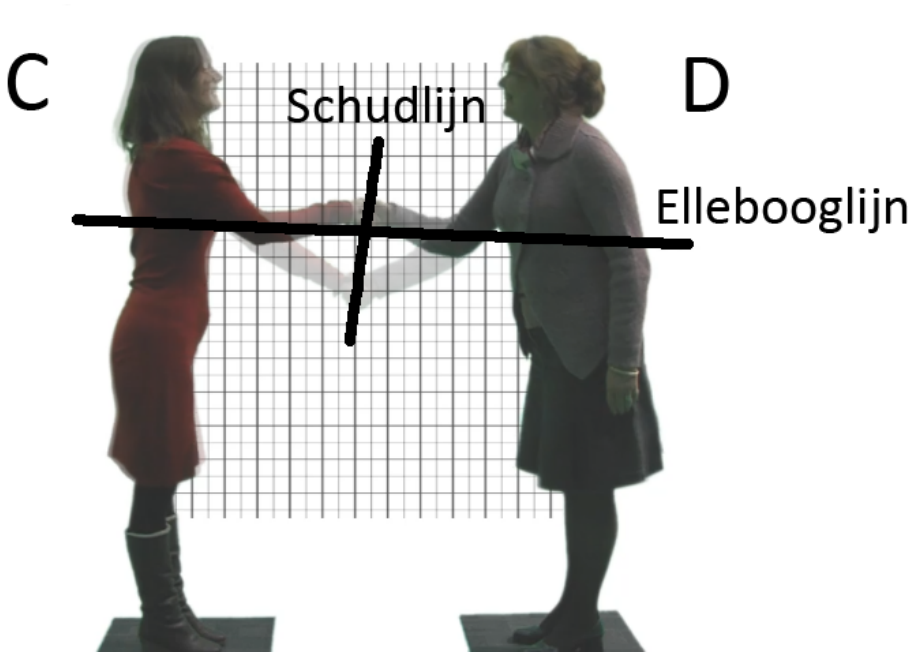


Figuur 4.6: Bovenaanzicht Case 2[6]

Aangezien beide proefpersonen ongeveer even lange armen hebben, steekt proefpersoon D zijn arm dus verder uit. Dit zou kunnen impliceren dat proefpersoon C de meer dominante persoon is, alhoewel de afwijking tamelijk klein is.

[6]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Zijaanzicht



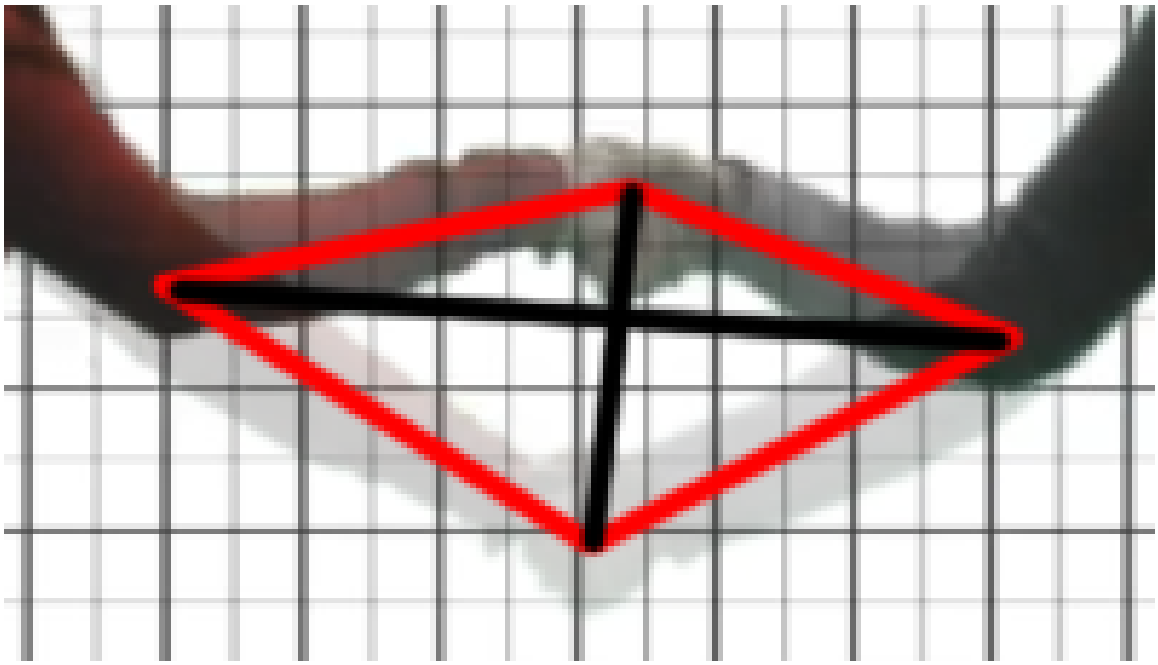
Figuur 4.7: Hoekmeting Case 2[7]

	Hoek ellebooglijn-vloer	Hoek schudlijn-vloer	Hoek ellebooglijn-schud
HS_FF_01	0,0	82,2	82,2
HS_FF_02	2,3	63,4	61,1
HS_FF_03	0,0	83,7	83,7
HS_FF_04	4,1	66,8	62,7
HS_FF_05	-3,8	90,0	93,8
Case 2 gemiddelde	0,5	77,2	76,7

Tabel 4.2: Hoekanalyse Case 2

Uit tabel 4.2 bemerkt men allereerst dat de hoek tussen ellebooglijn en schudlijn ditmaal geen 90 graden bedraagt. Dit is in tegenspraak met eerdere assumpties, nadere analyse van het beeldmateriaal blijkt aan te tonen dat proefpersoon D een dominanter gedrag vertoont. Dit staat in contrast met de assumptie uit de analyse van het bovenaanzicht, maar zoals eerder gezegd kon die afwijking verwaarloosd worden. Proefpersoon D lijkt meer de handdruk haar richting uit te trekken. Anderzijds is de hoek van de ellebooglijn met de vloer dicht bij nul graden, wat aangezien beide proefpersonen even groot zijn indicatie is dat deze hoek afhankelijk is van hoogteverschil tussen proefpersonen. Ten slotte wordt er nogmaals gekeken naar de afstand van de shakefase.

[7]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 4.8: Shakefase Case 2[8]

Uit figuur 4.8 wordt de afstand van de shakefase bepaald. Bekeken van opzij is deze wederom ongeveer de helft van de afstand tussen beide ellebogen. Om de werkelijke verhouding te bekomen moet er nog een compenserende factor toegevoegd worden die de rotatie in het bovenaanzicht in rekening brengt. Uit figuur 4.6 haalt men een hoek van ongeveer 25 graden voor de lijn tussen de ellebogen en het projectievlak. Dit betekent dat de effectieve hoogte van de shakefase zich verhoudt tot de afstand tussen beide ellebogen volgens volgende betrekking:

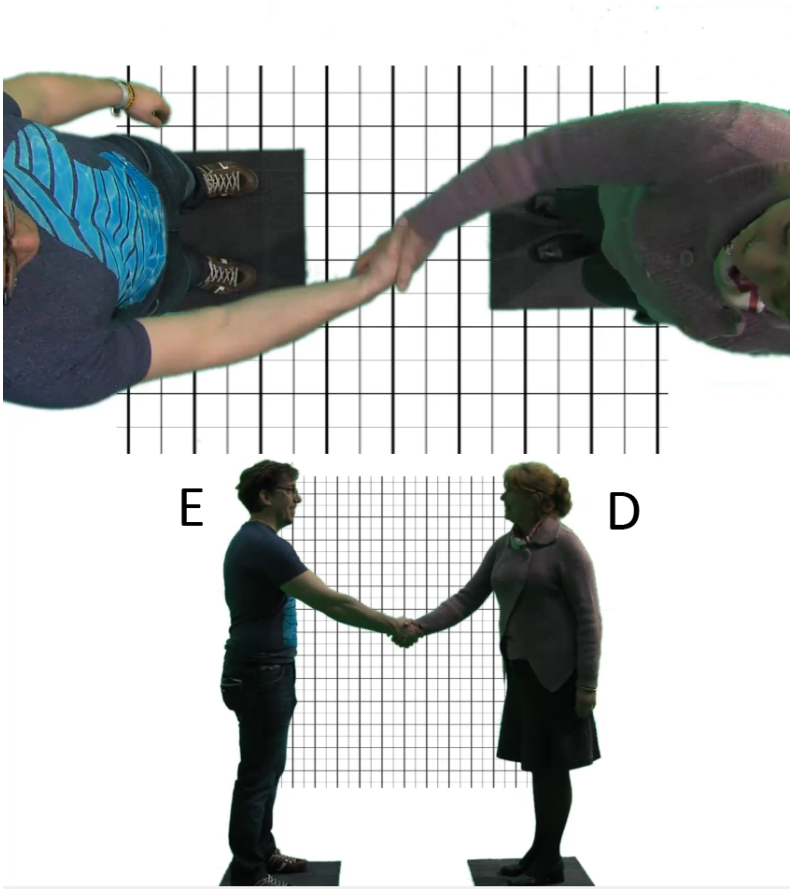
$$hoogte\ shakefase = \frac{cos(25°) * afstand\ tussen\ ellebogen}{2} \tag{4.2}$$

[8]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

4.4 Case 3: Één mannelijke en één vrouwelijke proefpersoon

4.4.1 Opstelling

De laatste case is een opname met één mannelijke en één vrouwelijke proefpersoon. Proefpersonen D en E hebben ongeveer dezelfde lichaamslengte. In figuur 4.9 ziet men de proefopstelling.



Figuur 4.9: Proefopstelling Case 3[9]

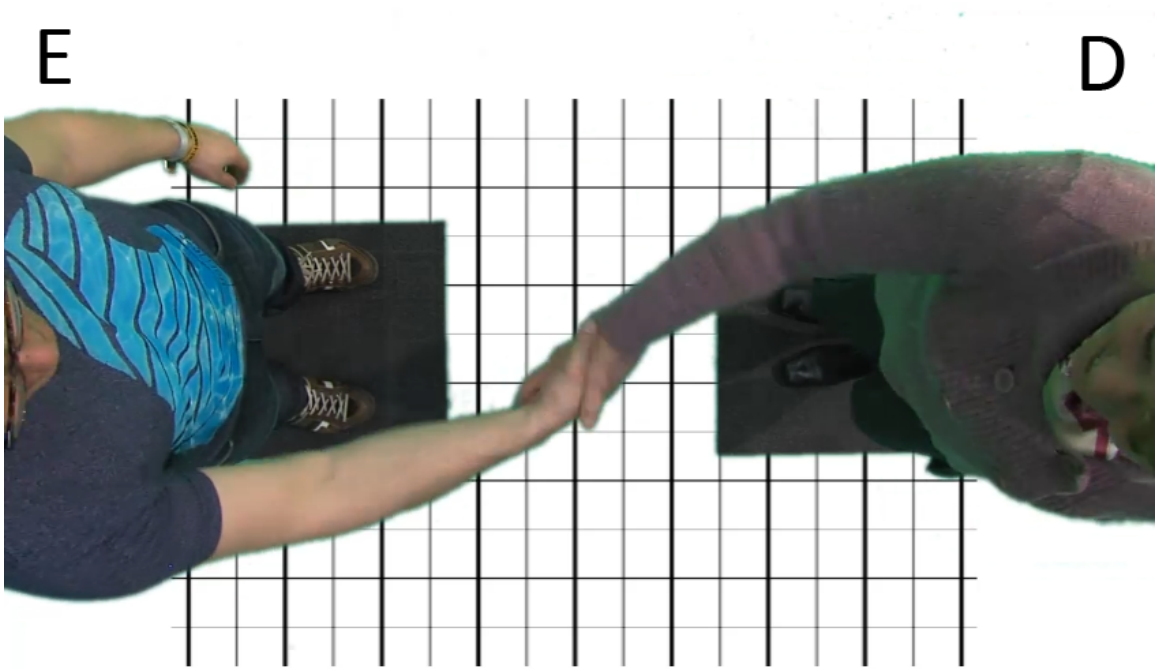
4.4.2 Analyse

Gelijkaardig aan Case 2 bestaat deze animatie uit meerdere oscillaties, ditmaal lijkt proefpersoon E te proberen de uitwijking groot te houden voor alle oscillaties. Dit in combinatie met het dominantere gedrag[10] van proefpersoon D, creëert oscillaties die langzamer gedempt worden dan in Case 2. Aangezien de demping blijft en de eerste uitwijking nog steeds aanwezig is, wordt de focus nog steeds geplaatst op de eerste of primaire uitwijking.

[9] Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[10] Zoals besproken in sectie 4.3.

Bovenaanzicht

De locatie van de handen blijft wederom in lijn met de twee ellebogen. De startpositie bevindt zich in het midden. (Zie figuur 4.10)

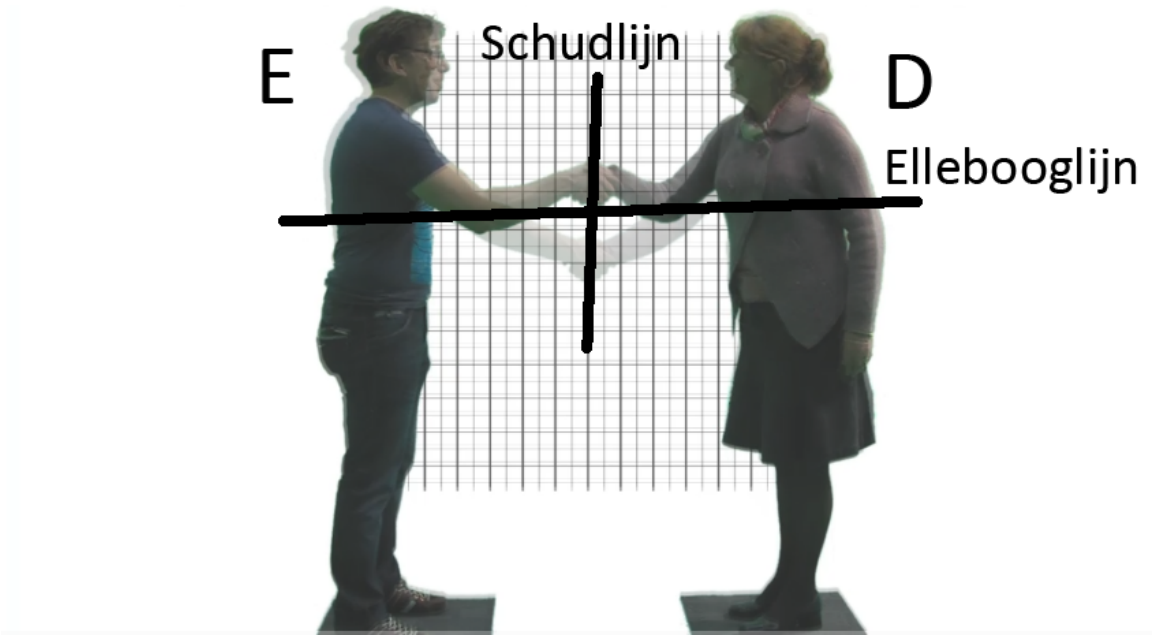


Figuur 4.10: Bovenaanzicht Case 3[11]

Er is geen afwijking aanwezig en de proefpersonen zijn even groot, er is hier geen significant dominantieverschil aanwezig.

[11]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Zijaanzicht



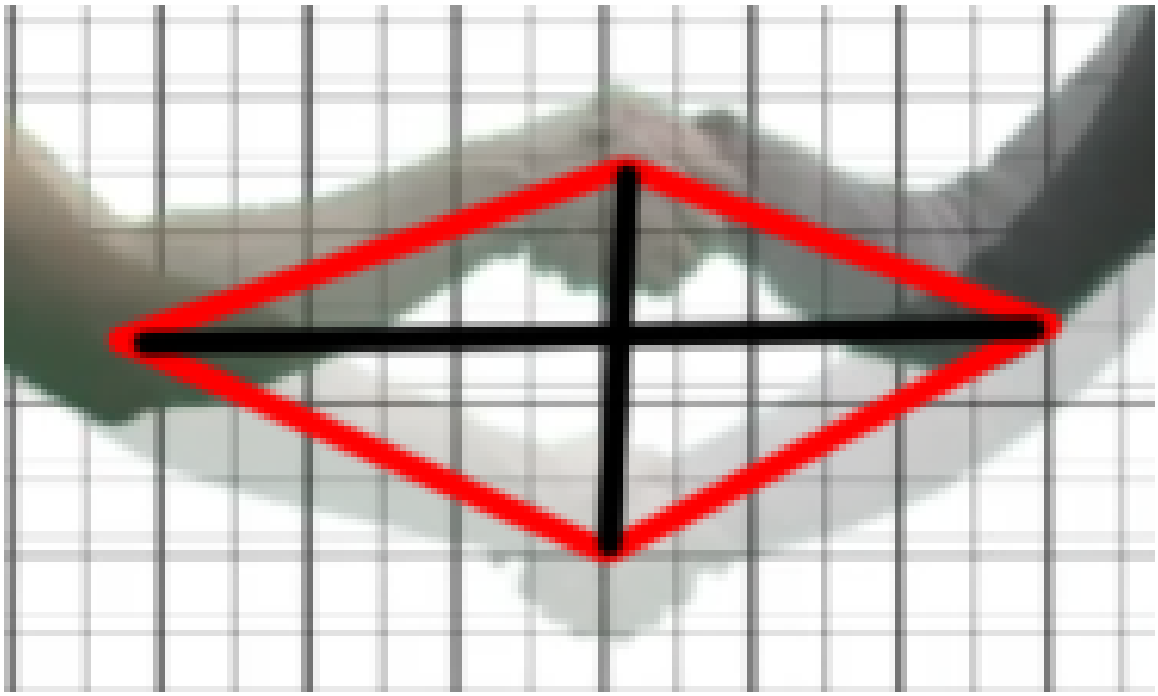
Figuur 4.11: Hoekmeting Case 3[12]

	Hoek ellebooglijn-vloer	Hoek schudlijn-vloer	Hoek ellebooglijn-schud
HS_MF_01	4,8	86,3	81,5
HS_MF_02	5,2	90,0	84,8
HS_MF_03	5,2	87,7	82,5
HS_MF_04	0,0	86,6	86,6
HS_MF_05	4,2	80,5	76,3
Case 3 gemiddelde	3,9	86,2	82,3

Tabel 4.3: Hoekanalyse Case 3

Uit tabel 4.3 bedraagt de hoek tussen ellebooglijn en schudlijn ook geen 90 graden, maar het ligt er wel dichterbij dan in Case 2. Uit nadere analyse blijkt dat het dominantere karakter van proefpersoon D meer wordt tegenwerkt door proefpersoon E in vergelijking met proefpersoon C. Hoewel beide proefpersonen weer ongeveer even groot zijn, is de hoek tussen ellebooglijn en vloer niet dicht bij nul. Dit impliceert dat hier wellicht ook een deel personificatie aan te pas komt. Dit wordt overgelaten als toekomstig werk. Tot slot wordt de afstand van de shakefase geanalyseerd.

[12]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 4.12: Shakefase Case 3[13]

Uit figuur 4.12 wordt de hoogte van de shakefase bepaald. Bekeken van opzij is deze wederom ongeveer de helft van de afstand tussen beide ellebogen. Om de werkelijke verhouding te bekomen moet er nog een compenserende factor toegevoegd worden die de rotatie in het bovenaanzicht in rekening brengt. Uit figuur 4.10 haalt men een hoek van ongeveer 30 graden voor de lijn tussen de ellebogen en het projectievlak. Dit betekent dat de effectieve hoogte van de shakefase zich verhoudt tot de afstand tussen beide ellebogen volgens volgende betrekking:

$$hoogte\ shakefase = \frac{\sqrt{2} * afstand\ tussen\ ellebogen}{4} \tag{4.3}$$

4.5 Handgreep

In elke case behoudt de handgreep zijn oriëntatie doorheen de shakefase. Om het model te vervolledigen wordt er nog even dieper ingegaan op de handgreep en hoe die wordt gevormd op basis van de dimensies van handen. Er wordt een analyse gemaakt van één enkele handgreep tussen twee proefpersonen. Dit maakt het mogelijk om een rudimentair handgreepmodel te maken. De limitatie hierbij is natuurlijk dat indien er een verschil is in grootte tussen beide handen dat het model mogelijk niet meer correct reageert. Het onderzoek naar invloed van handgrootte wordt voor de toekomst bewaard.

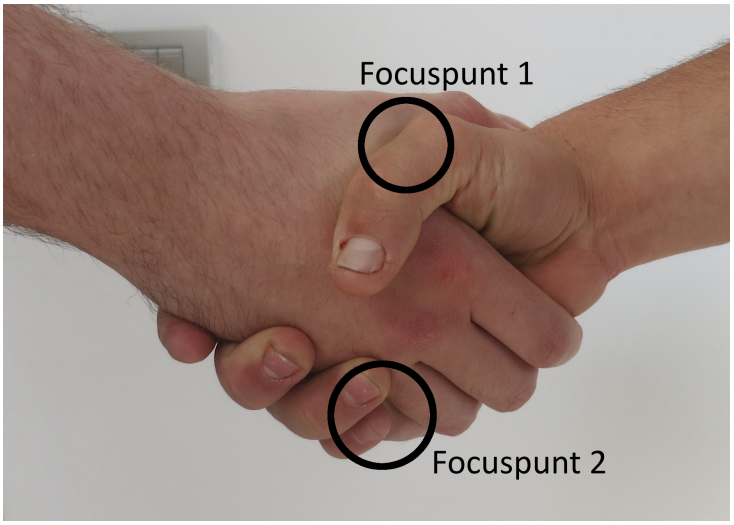
[13]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

4.5.1 Proefopstelling

Figuur 4.13: Proefopstelling Handgreep[14]

In figuur 4.13 ziet men de proefopstelling.

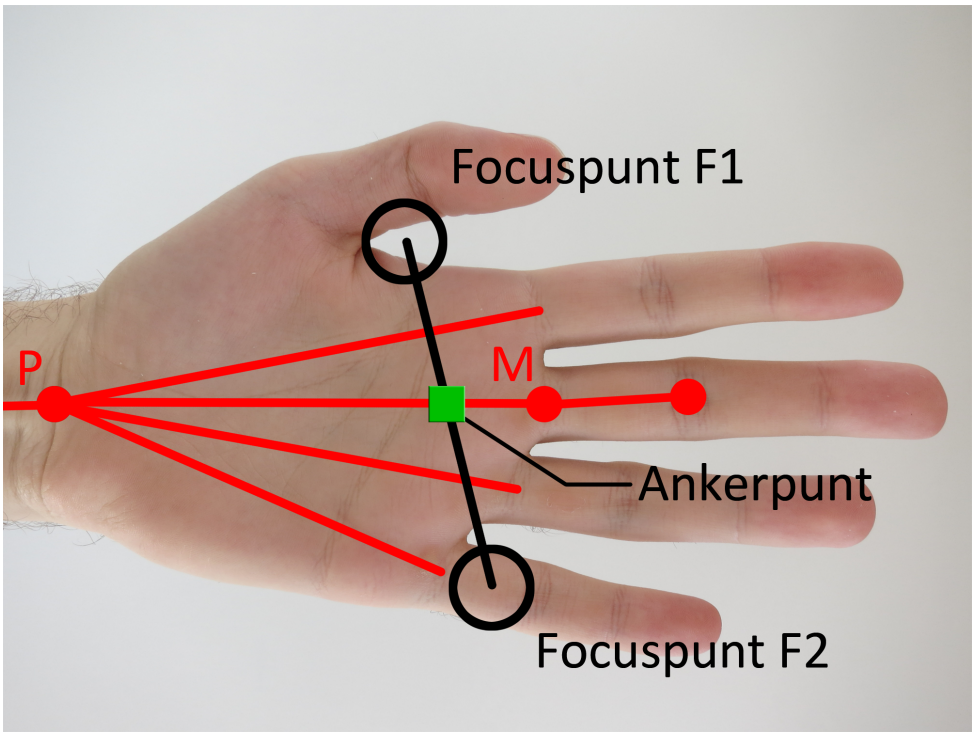
4.5.2 Analyse



Figuur 4.14: Analyse Handgreep: Focuspunten[15]

In figuur 4.14 worden twee focuspunten vastgelegd. Focuspunt F1 duidt het contact aan van het diepste punt tussen duim en wijsvinger van beide handen. En focuspunt F2 duidt het contact aan van het onderste vingerkootje van de pink van beide handen.

[14]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[15]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 4.15: Analyse Hand: Focuspunten[16]

In figuur 4.15 ziet men de focuspunten op één hand. Om een hand te oriënteren in Computer Graphics, wordt gebruik gemaakt van transformatiematrices. Dit betekent dat er een translatie en rotatie nodig is voor de volledige hand. De translatie wordt aangegeven als een *offset* t.o.v. de oorsprong van het object wat in dit geval de pols is. We zoeken dus één ankerpunt die deze offset representeert. De twee focuspunten geven positionele informatie en worden dus gebruikt om een offset mee te bepalen. We nemen een punt op de lijn tussen de twee focuspunten dewelke kruist met één van de botten, dit maakt het gemakkelijker om de vectoren uit te rekenen. We nemen meer specifiek de kruising met het bot die loopt richting de middelvinger. Er moet ook nog rekening gehouden worden met de afwijking in de richting van de normaal op de handpalm, dit zodat de twee handen niet door elkaar gaan en er dus enig fysisch realisme aanwezig is. Nu ontbreekt slechts nog de rotatie. Aangezien de twee focuspunten samenvallen op beide handen(Zie figuur 4.14), wordt de lijn die hier gecreëerd wordt gebruikt om de hand mee te oriënteren. De hoek tussen de lijn F1-F2 en de as die loodrecht staat op het middenhandsbeentje[17] richting de middelvinger wordt hiervoor gebruikt. (Zie figuur 4.18) Hierbij wordt er relatief t.o.v. het middenhandsbeentje richting de middelvinger gekeken omdat deze in neutrale staat vrijwel uitgelijnd staat met de voorarm. Deze hoek wordt gebruikt om de hand te roteren rond het punt P in het vlak beschreven door de punten P,M en F2. A.d.h.v. de translatie en rotatie kan men nu de positie van de hand in een handdruk vastleggen.

[16]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

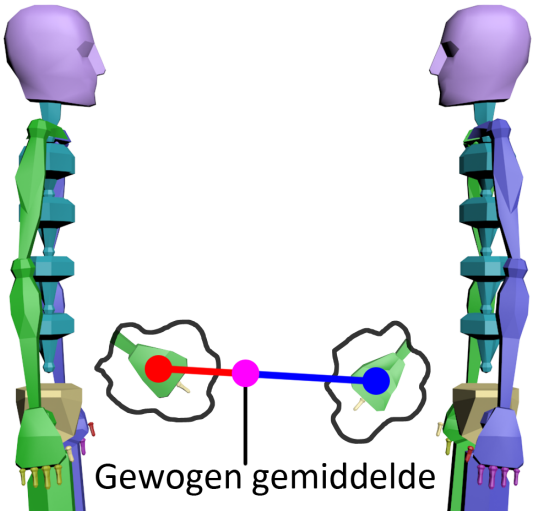
[17]Een middenhandsbeentje is een been in een hand die van de pols naar de voet van één van de vingers loopt.

4.6 Conclusie

Aan de hand van alle analyses kan nu een model gebouwd van een basis handdruk. Het handdruk model wordt hoofdzakelijk gedefinieerd met drie componenten: de startpositie van de handdruk, de vector van beweging voor de shakefase en de handgreep.

4.6.1 Startpositie van de handdruk

De startpositie van de handdruk ligt gemiddeld gezien exact in het midden tussen beide personen. Deze positie kan licht verschoven worden door een verschil in dominantie tussen beide personen. Hierdoor zal in het model deze positie bepaald worden door een gewogen gemiddelde te nemen op basis van het karakter van de persoon. Zo wordt op basis van de posities, waar beide karakters voor zichzelf denken dat de startpositie moet zijn, het gewogen gemiddelde bepaald. (Zie figuur 4.16)



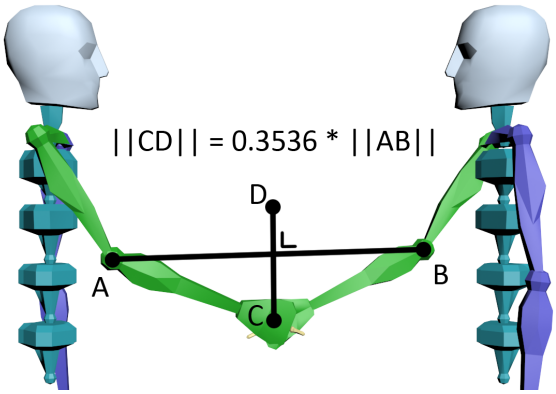
Figuur 4.16: Startpositie handdruk[18]

Hier moet rekening gehouden worden met de fysieke limitaties, meer specifiek of beide armen deze gemiddelde positie al of niet kunnen bereiken. Om dit toch deels te compenseren wordt er nog een compenserende factor toegevoegd op basis van de lengte van de armen en de afstand tussen beide karakters.

[18]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

4.6.2 Vector van beweging voor de shakefase

De vector van beweging wordt bepaald door richting en grootte. De richting wordt genomen in het vlak gemaakt door de twee ellebogen en de handdruk zelf. De richting staat loodrecht op de as die door beide ellebogen gaat, maar kan licht gekanteld worden afhankelijk van de dominantie van de personen die in interactie gaan. De richting is altijd omhoog ten opzichte van de vloer. Hoewel de grootte niet altijd gelijk was bij alle cases, wordt de meest voorkomende genomen. (Deze met een compensatie factor van 30 graden.) De grootte wordt dan gedefinieerd als ongeveer 35% van de afstand tussen beide ellebogen bij de start van de shakefase. (Zie figuur 4.17)



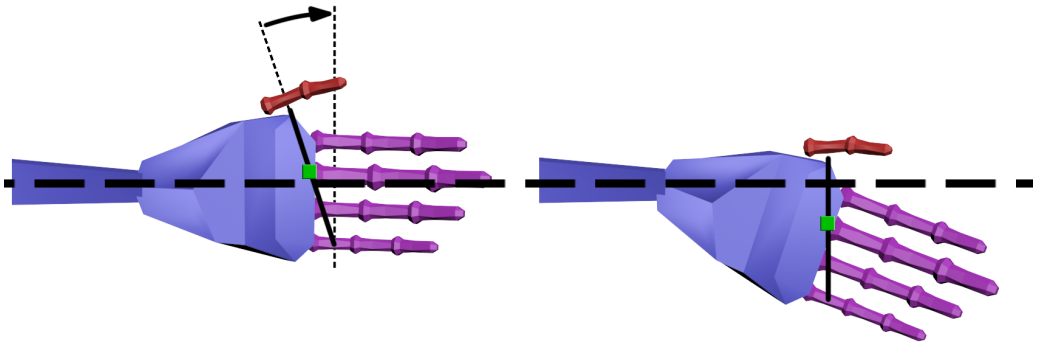
Figuur 4.17: Vector van beweging voor de shakefase[19]

4.6.3 Handgreep

Voor de handgreep zelf wordt de stand individueel bepaald door elke persoon. De handgreep bevat twee factoren: positie en oriëntatie. De positie wordt gedefinieerd als de afstand van de pols tot het snijpunt van het middenhandsbeentje, dat verbonden is met de middelvinger, en de as gecreëerd door de verbinding van de twee focuspunten. De exacte beschrijving van de focuspunten vindt men in sectie 4.5. Er moet ook een verschuiving gebeuren in de richting van de normaal op de palm, dit zodat de handen niet door elkaar schuiven en het fysisch realistisch blijft.

[19]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

De oriëntatie van de hand wordt als volgt beschreven. De handpalm bevindt zich in het verticale vlak dat door de twee ellebogen loopt. (Let wel dat dit vlak niet perfect verticaal hoeft te zijn, in de literatuurstudie werd al aangehaald dat dominantie een invloed heeft op deze hoek.) In dit vlak is er ook nog een kleine rotatie aanwezig die het mogelijk maakt dat de twee focuspunten van beide handen mooi uitlijnen.



Figuur 4.18: Handgreep: Rotatie[20]

[20]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Hoofdstuk 5

Raamwerk

5.1 Inleiding

Om animaties te kunnen simuleren, werd er eerst een raamwerk gerealiseerd. Dit raamwerk moest geanimeerde karakters in een *3D-viewport*[1] kunnen weergeven en simuleren. Het basisraamwerk voor het renderen van 3D objecten werd gerealiseerd in het XNA 4.0 Framework van Microsoft. Voor Inverse Kinematics werd er een externe bibliotheek gebruikt, DigitalRune genaamd. DigitalRune biedt tal van functionaliteiten aan voor verscheidene soorten animaties, daarom werd deze bibliotheek als basis gebruikt voor zowel animatie sturing als skelet structuur. XNA en DigitalRune hebben elk hun eigen implementatie voor vectoren en matrices[2]. XNA maakt gebruik van volgende conventies [18]:

- Rechtshandig coördinatensysteem
- Row-major matrices, rijen voor kolommen

DigitalRune heeft volgende conventies [19]:

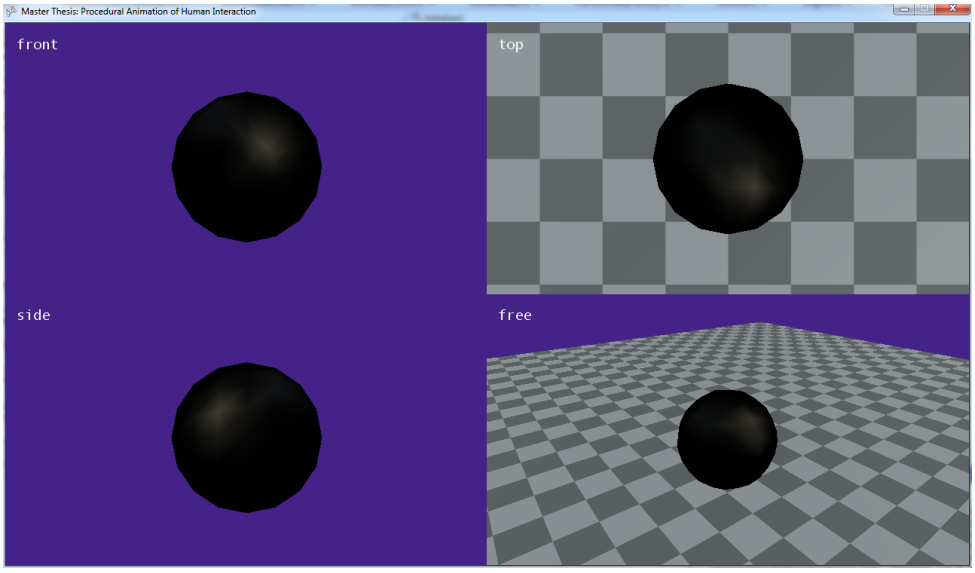
- Rechtshandig coördinatensysteem
- Column-major matrices, kolommen voor rijen

DigitalRune levert echter geen implementatie van Fuzzy Logic. Hiervoor werd de open-source bibliotheek *AForge.NET* gebruikt. Om deze controllers extern te kunnen calibreren via bestanden werd een wrapper geschreven. Deze wrapper laat toe om FCL bestanden binnen te laden en Fuzzy Logic controllers mee te bouwen. De FCL bestanden zijn gebaseerd op de standaard die beschreven staat bij de Free Fuzzy Logic Library [20].

[1] Een *3D-viewport* is een venster waarin een 3D omgeving getekend kan worden.

[2] Matrixdefinitie bepaalt de manier waarop matrixen moeten vermenigvuldigt worden: links naar rechts of rechts naar links.

5.1.1 GUI



Figuur 5.1: Graphical User Interface[3]

In figuur 5.1 ziet men de GUI. Er werd gekozen voor een 2 op 2 raster van viewports, dit wordt ook gebruikt in meeste 3D tekenprogramma's.

Figuur 5.2: Viewportverdeling. [4]

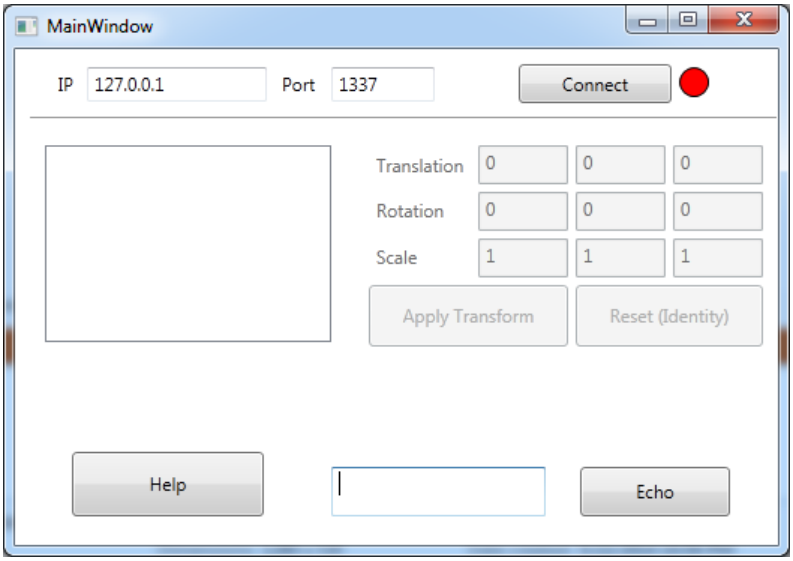
In figuur 5.2 ziet men de verdeling van aanzichten voor elk van de vier viewports. Waarbij vooraanzicht, zijaanzicht en bovenaanzicht orthografische projecties zijn en de 3D-viewport een perspectief projectie van de scène weergeeft.

[3]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[4]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Voor de GUI werd gekozen voor een minimum aan GUI-elementen, dit komt door het gebrek aan standaard componenten binnen XNA. Gebruikersinput wordt hoofdzakelijk gegeven door sneltoetsen te gebruiken, deze worden verbonden aan specifieke acties zoals handen schudden, objecten roteren en verplaatsen.

5.1.2 Externe GUI

Om toch een zekere vergemakkelijking van gebruikersinput te hebben, werd er een externe applicatie gemaakt in Microsoft WPF. (Zie figuur 5.3)



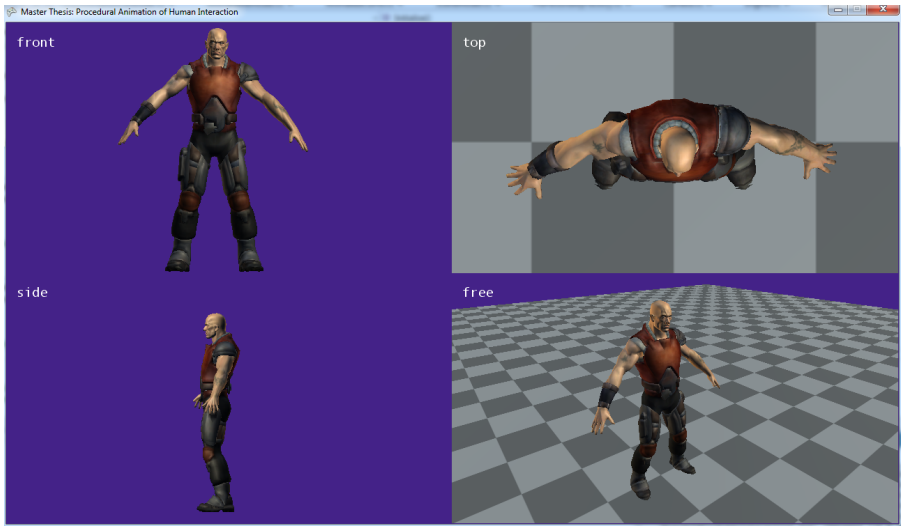
Figuur 5.3: Extern besturingsvenster[5]

Deze applicatie wordt simpelweg uitgerust met verscheidene GUI-elementen die WPF aanbiedt. Om dit venster te koppelen aan het *3D rendering* raamwerk werd er gebruikt gemaakt van netwerksockets meer specifiek een TCP-verbinding tussen beide applicaties. Dit maakt het theoretisch mogelijk om het raamwerk te bedienen vanop een externe computer.

5.2 Virtuele karakter model

In dit stuk worden kort de mogelijkheden en aspecten van de virtual humans binnen dit raamwerk besproken. In figuur 5.4 ziet men de virtual human zoals hij wordt weergegeven binnen het raamwerk. Het gebruikte 3D model werd meegeleverd met de gratis XNA bibliotheek.

[5]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 5.4: Virtual human in raamwerk[6]

Eerst wordt de virtual human kort besproken volgens het model van Norman Badler, en vervolgens worden bepaalde aspecten meer in detail besproken.

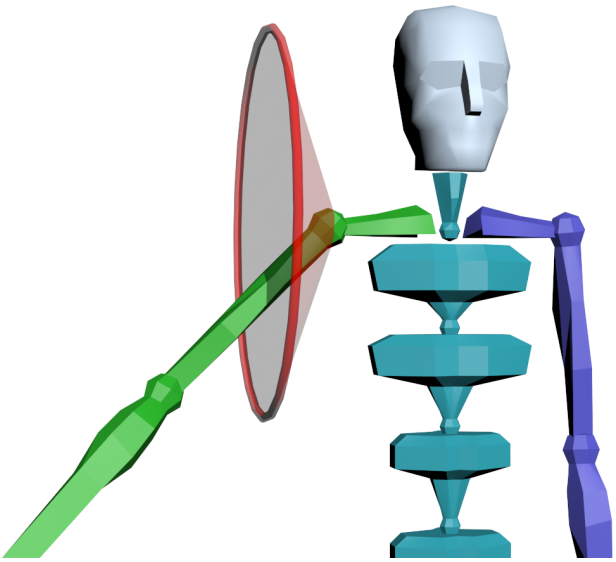
Dimensie	Beschrijving
Uiterlijk	Het uiterlijke realisme is niet cruciaal voor deze thesis. Er werd ervoor gekozen om een simpel model te gebruiken dat ervaren wordt als zijnde menselijk, maar niet foto-realistisch.
Functie	Aangezien de focus van dit werk gaat over handdrukken wordt gekeken om aan de menselijke limitaties in zake gewrichten voor de rechtarm en een stuk van het torso te voldoen.
Tijd	De handdruk wordt in real-time opgebouwd bij het starten van de animatie, maar zal zich niet aanpassen aan externe factoren gedurende de animatie.
Autonomie	De karakters bezitten geen autonomie en moeten aangestuurd worden door de onderzoeker.
Individualiteit	Voor elk karakter kan een persoonlijkheid gecreëerd door een reeks van karaktertrekken te specifiëren.

Tabel 5.1: Virtual Human Specificatie volgens Norman Badler

[6]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

5.2.1 Functie

Er bestaan verschillende methoden om gewrichtsrotatie te begrenzen bij virtual humans. De simpelste manier is om de oriëntatie van een gewricht op te splitsen in hoeken van Euler en elke hoek apart te begrenzen. Een andere methode is door gebruik te maken van kegels. Dit begrenst rotatie over twee vrijheidsgraden maar begrenst rotatie rond het bot zelf niet. In figuur 5.5 wordt een kegelbegrenzing weergegeven.



Figuur 5.5: Voorbeeld van kegelbegrenzing[7]

Verder onderzoek naar betere manieren van begrenzing heeft onder andere geleid tot de bereikskegel of *reach cone*. [21] Hierbij wordt de begrenzing voorgesteld als een een veelhoek die zich op een boloppervlak bevindt. Deze veelhoek kan nog uitgebreid worden naar een parametrische curve.

5.2.2 Tijd

De virtual humans worden best niet bewogen tijdens de animatie, de animatie zal zich gedeeltelijk aanpassen aan de nieuwe situatie. De locatie van de handdruk wordt echter vastgelegd in de wereld, wat het bewegen van beide virtual humans tegelijk verbiedt.

[7] Gemaakt door Gaétan Deglorie voor gebruik in dit document.

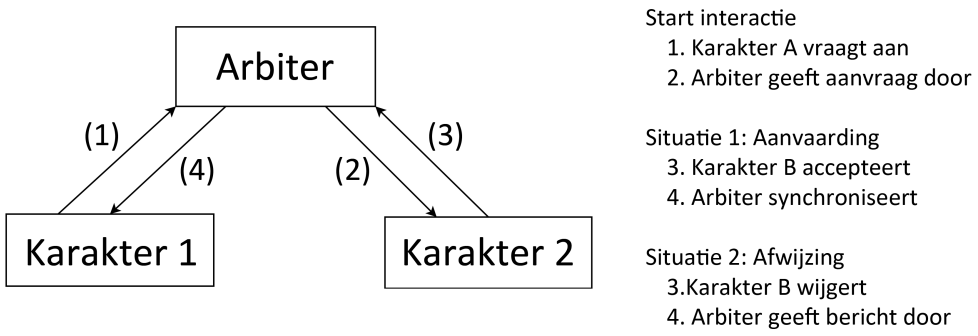
5.2.3 Individualiteit

De persoonlijkheid van de virtual human bestaat uit een reeks karaktertrekken. Voor de eenvoud worden de karaktertrekken binnen het raamwerk voorgesteld door numerieke waarden. Hierbij worden fysieke karaktertrekken omgezet in getallen, zoals bvb. dominantie. Dit maakt het gemakkelijker om persoonlijkheden te vergelijken in code. Het raamwerk laat toe dat in de toekomst andere types van karaktertrekken kunnen gebruikt worden zoals tekstuele variabelen of groeperingen van meerdere variabelen. Hieronder ziet men een voorbeeld van de implementatie van een persoonlijkheid.

```
class CustomPersonality : PersonalityF
{
        void Initialize()
        {
                AddTrait("Dominantie",0.7f);
                AddTrait("Verlegenheid",0.2f);
                ...
        }
}
```

5.3 Interacties

Zoals besproken in sectie 2.4.2 wordt handen schudden geclassificeerd als een Initiator - Initiator interactie. Dit betekent dus een interactie model waarbij beide karakters actief input moeten geven om een interactie te ondergaan. Let wel dat deze actie niet gelijktijdig moet gebeuren, zoals bij handdrukken en "high fives" waarbij één van beide de interactie start waarna de andere de interactie accepteert door mee te doen aan de interactie. Dit vereist een model voor Initiator - Initiator interactie die werkt met asynchrone start. Om eigenschappen van beide karakters door te geven aan elkaar werd een arbiter structuur opgesteld. (Zie figuur 5.6)



Figuur 5.6: Simpele arbiter structuur[8]

Dit betekent dat de tweede partij op een asynchrone manier kan ingaan op de interactie. Aangezien de mogelijkheid bestaat dat er niet gereageerd wordt op de aanvraag, moet er een vervaltijd voorzien worden om de interactie af te breken. Anders zou de startende partij voor eeuwig blijven wachten op een reactie wanneer de tweede partij niet expliciet zou reageren met ja of nee. Let wel dat deze logica volledig onafhankelijk is van animatie en slechts gebruikt wordt om de staat van beide karakters tijdens de interactie bij te houden. Dit betekent dat de animatie volledig los staat van de interactie en dat deze interactie kan hergebruikt worden voor alle Initiator - Initiator interacties (voor exact 2 karakters).

[8]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

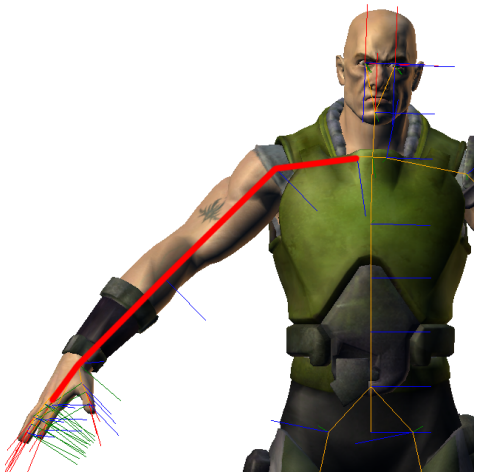
Hoofdstuk 6

De handdruk

In dit hoofdstuk wordt dieper ingegaan op het animeren van een handdruk. Eerst wordt dieper ingegaan hoe animatiecontrollers werken voor Inverse Kinematics en Fuzzy Logic, gevolgd door uitleg over de animaties specifiek en hoe die interageren met de animatie-controllers.

6.1 Animatie

De meeste animaties beschrijven posities en oriëntaties voor alle gewrichten in een skelet. Voor dit onderzoek zal de focus liggen op het animeren van de arm en het sleutelbeen. (Zie figuur 6.1)



Figuur 6.1: Focus van animatie[1]

Indien er een meer natuurlijke beweging gewenst is van het volledige lichaam, moet men zoeken naar manieren om blending te gebruiken in combinatie met de hier besproken technieken. Dit wordt overgelaten als toekomstig werk.

[1]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

6.1.1 Opbouw

Beide animatietechnieken maken gebruik van doelposities om de gewrichtsparameters te bepalen. Dus voor dit onderzoek wordt een basisanimatie aanzien als een animatie die gebruik maakt van een doelpositie zoals hieronder te zien is:

```
abstract class BaseAnimation
{
        AnimationTarget m_animationTarget;
        Subject m_subject;

        abstract void Play();
        virtual void Update(GameTime gameTime) { }
}
```

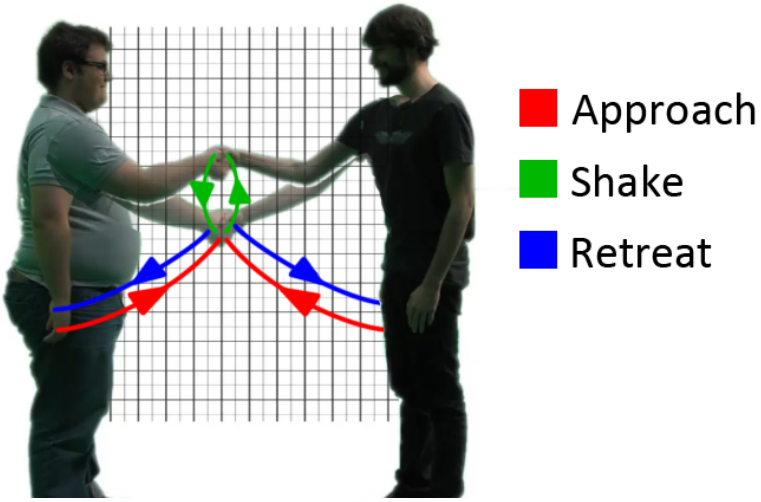
Het object dat de doelpositie beschrijft, *m_animationTarget*, bevat niet enkel positie, maar ook extra informatie om de animatie beter aan te sturen. Een offset parameter beschrijft een eventuele extra translatie t.o.v. de doelpositie en een oriëntatie parameter die de stand van het gewricht vastlegt. Deze oriëntatie parameter kan selectief in- of uitgeschakeld worden tijdens de animatie. De klasse structuur voor dit object ziet er als volgt uit:

```
class AnimationTarget
{
        Vector3F m_position;
        Vector3F m_offset;

        bool m_enforceRotation;
        Matrix33F m_handleRotation;
}
```

Een interactieanimatie heeft kennis nodig over de andere animatie die meedoet aan de interactie, dit zodat het geheel er fysisch correct uitziet. Om tijdens de animatie bepaalde informatie door te geven aan elkaar wordt er nog een synchronisatiemechanisme geïmplementeerd. (Zie codevoorbeeld hieronder)

```
abstract class InteractionAnimation : BaseAnimation
{
        InteractionAnimation m_matchingAnimation;
        event SynchronizeEventHandler Synchronize;

        abstract void SynchAll(InteractionAnimation other);
}
```

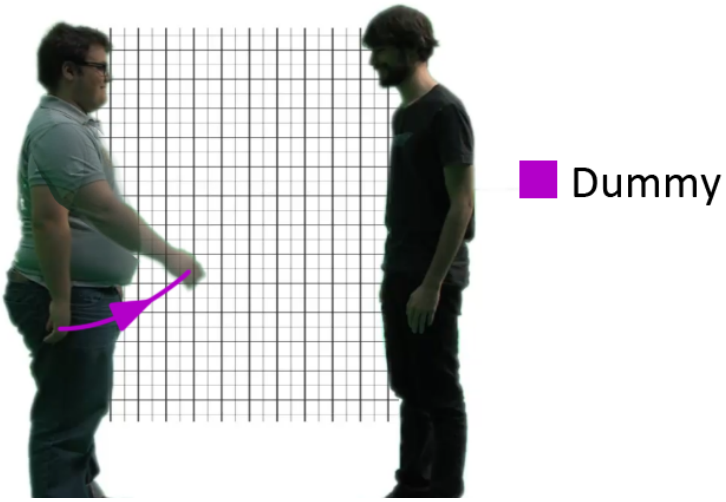
Door deze implementatie abstract te houden wordt interactieanimatie als een soort laag geïmplementeerd, volledig onafhankelijk van gebruikte animatietechnieken. De handdruk animatie moet aan een aantal regels voldoen. Tijdens het schudden mogen de beide handen niet loskomen van elkaar, en hierbij moet de oriëntatie van beide handen correct blijven. De positie/oriëntatie van de handen zal voor bepaalde stukken van de animatie dus moeten gedeeld/vastgelegd worden. Om dit op een uniforme manier aan te pakken, wordt de animatie opgesplitst in een aantal fases: *approach*, *shake* en *retreat*. (Zie figuur 6.2)



Figuur 6.2: Fases: approach, shake en retreat[2]

Om volledig te zijn wordt nog een vierde fase gedefinieerd, die alleen relevant is voor de persoon die de handdruk start. Voordat beide personen elkaars hand proberen vast te nemen, zal één van beide personen zijn hand uitreiken naar een positie die hij/zij denkt die zal overeenkomen met de uiteindelijke initieel contactpunt van beide handen. Deze fase wordt de *dummyfase* genoemd. (Zie figuur 6.3)

[2]Gemaakt door Gaétan Deglorie voor gebruik in dit document.



Figuur 6.3: Dummyfase[3]

Om de handposities te doen samenvallen op het eind van de approachfase en tijdens de shakefase wordt gebruikt gemaakt van een synchronisatie mechanisme aan de start van deze fases. Dit impliceert echter dat beide animaties niet zullen correct blijven indien er tijdens een gesynchroniseerde fase één van beide karakters van positie of oriëntatie verandert wordt. Voor de approachfase wordt de startpositie van de shakefase gesynchroniseerd, aangezien deze als einde gebruikt wordt voor deze fase. Voor de shakefase wordt het volledige pad (van startpositie naar piek en terug) gesynchroniseerd.

6.1.2 Inverse Kinematics aanpak

Zoals eerder aangehaald, werkt IK animatie op basis van paden. Voor de handdruk wordt een pad gecreëerd voor iedere fase. Om de paden te definiëren wordt er gebruikt gemaakt van parametrische curven. Deze paden worden beschreven door een start- en eindpunt, duurtijd. Verder kan er gebruik gemaakt worden van controlepunten om het pad van een rechte lijn naar een kromme te vervormen, zoals gedaan wordt bij onder meer Bézier curven. Voor de duurtijd en controlepunten werden er voor statische waarde gekozen, dit betekent dat de animaties er niet volledig natuurlijk uitzien. In de toekomst kunnen deze aspecten opgenomen worden als karaktertrekken in de persoonlijkheid van iedere karakter. Om fouten te vermijden kunnen deze geanimeerde paden gedeeld worden tussen meerdere karakters. Hier is dit nodig bij de shakefase, zodat beide handen elkaar blijven aanraken. In tabel 6.1 ziet men voor iedere fase de start- en eindpunten met een aanduiding of het een gedeeld pad betreft.

[3]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Fase	Startpunt	Eindpunt	Gedeeld
Dummy	Huidige handpositie	Gewenste handdrukpositie	Nee
Approach	Huidige handpositie	Startpositie van de handdruk	Nee
Shake	Startpositie van de handdruk	Startpositie van de handdruk	Ja
Retreat	Huidige handpositie	Neutrale handpositie	Nee

Tabel 6.1: IK fasebeschrijving

Bij de shakefase wordt er nog gebruik gemaakt van een derde punt die tussen het start- en eindpunt ligt. De locatie van dit punt wordt bepaald d.m.v. de regels die bepaald werden in de casestudy.

6.1.3 Fuzzy Logic aanpak

Fuzzy Logic werkt op basis van doelposities. Hierbij is de manier waarop tot een doel bewogen wordt niet bepaald door de animatie zelf, maar eerder door de animatiecontroller[4]. Positie en snelheid zijn dus niet gegarandeerd onderweg naar een doelpositie. Dit is problematisch voor de shakefase. Een oplossing hiervoor is de snelheid van de animatiecontroller te verhogen en meerdere tussenposities te plaatsen onderweg naar de doelpositie. Hoe kleiner de afstand tussen deze nieuwe doelposities genomen wordt, hoe nauwkeuriger het pad gevolgd zal worden. Dit wordt bereikt door opnieuw gebruik te maken van paden zoals bij IK. Dit maakt het ook mogelijk om bij deze fase één karakter IK te laten gebruiken en een ander Fuzzy Logic. Hierbij wordt deze fase ook opnieuw gedeeld. In tabel 6.2 ziet men de verschillende doelposities per fase.

Fase	Doelpositie
Dummy	Gewenste handdrukpositie
Approach	Startpositie van de handdruk
Shake	Geanimeerd over pad
Retreat	Neutrale handpositie

Tabel 6.2: Fuzzy Logic fasebeschrijving

Het pad van de shakefase is exact dezelfde als die bij IK, de beschrijving over paden is te vinden in sectie 6.1.2.

[4]Animatiecontrollers worden besproken in sectie 6.2

6.2 Animatiecontroller

Een animatiecontroller is het mechanisme, dat ervoor zorgt dat de animatie toegepast wordt op het skelet van het karakter. Animatiecontrollers zijn simpel voor standaard animaties toegepast op één karakter, maar vereisen extra logica voor het animeren van interactieanimaties.

6.2.1 Opbouw

Een animatiecontroller heeft in zijn simpelste vorm het volgende: een animatie om uit te voeren en een karakter om de animatie op toe te passen. In code ziet dit er als volgt uit:

```
abstract class BaseAnimationController
{
        BaseAnimation m_baseAnimation;
        Subject m_owner;

        abstract void PlayAnimation(string name);
        virtual void Update(GameTime gameTime) { }
}
```

Deze code specifieert niet hoe of welke animatie er wordt uitgevoerd. Dit laat toe om de specifieke functionaliteit te implementeren via overerving, waarbij beide animatietechnieken nog steeds gebruik maken van éénzelfde basisstructuur. Om de interactieanimaties correct te doen verlopen, moeten beide animatiecontrollers kennis hebben over elkaar. Dit maakt dat beide controllers data kunnen opvragen over elkaar op elk tijdstip om de synchronisatiestappen[5] correct te doen verlopen. Zoals hieronder te zien is, wordt dit via een abstracte overerving bereikt.

```
abstract class BaseInteractionAC : BaseAnimationController
{
        BaseInteractionAC m_linkedAC = null;
        void Link(BaseInteractionAC biac) { ... }
}
```

Door deze implementatie abstract te houden wordt interactie als een soort laag geïmplementeerd volledig onafhankelijk van gebruikte animaties of animatietechnieken. Voor beide technieken wordt een aparte implementatie voorzien.

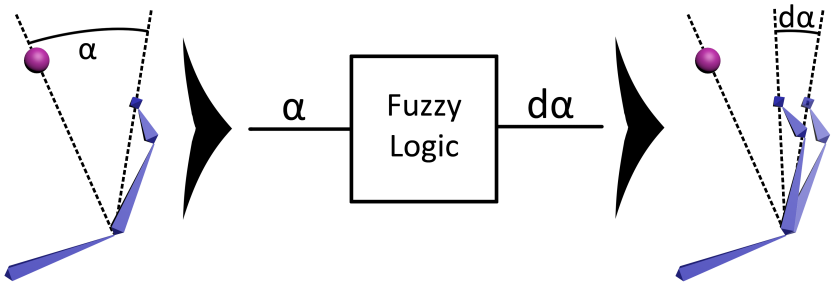
[5] De synchronisatiestappen werden besproken in sectie 6.1.

6.2.2 Inverse Kinematics aanpak

Een Inverse Kinematic animatiecontroller maakt gebruik van IK solvers. Binnen het raamwerk is er keuze tussen twee algoritmen: JacobianTranspose en CCD. Wanneer er een animatie afspeelt, wordt de doelpositie van het pad meegegeven aan de solver. Hierbij worden de berekende gewrichtsparameters ingesteld op het skelet van het karakter. De solver houdt geen rekening met gewrichtslimieten, want hij interpreteert alles als arbitraire ketens van gewrichten. Om hieruit een realistische animatie te krijgen moet dit rechtgezet worden. De huidige rotatie wordt opgesplitst in hoeken van Euler. Vervolgens wordt voor elke hoek gekeken hoeveel er teruggeroteerd moet worden indien men buiten de limiet is voor die rotatie. Dit vormt een variant op kegelbegrenzing, men kan de nieuwe vorm vergelijken met een piramide. Het is echter mogelijk dat de gewrichtslimieten de arm zover begrenzen dat de doelpositie niet meer bereikt kan worden. Dit vormt niet altijd een probleem, bijvoorbeeld tijdens de approachfase mag de hand afwijken van het ingestelde pad zolang hij maar eindigt bij de handgreep. De kans bestaat echter dat stabiliteitsproblemen naar boven komen door de IK solver die probeert de ene richting uit te gaan en de gewrichtslimieten die tegenwerken. Bij definitie van de paden moet er dus rekening gehouden worden met de ingestelde gewrichtslimieten.

6.2.3 Fuzzy Logic aanpak

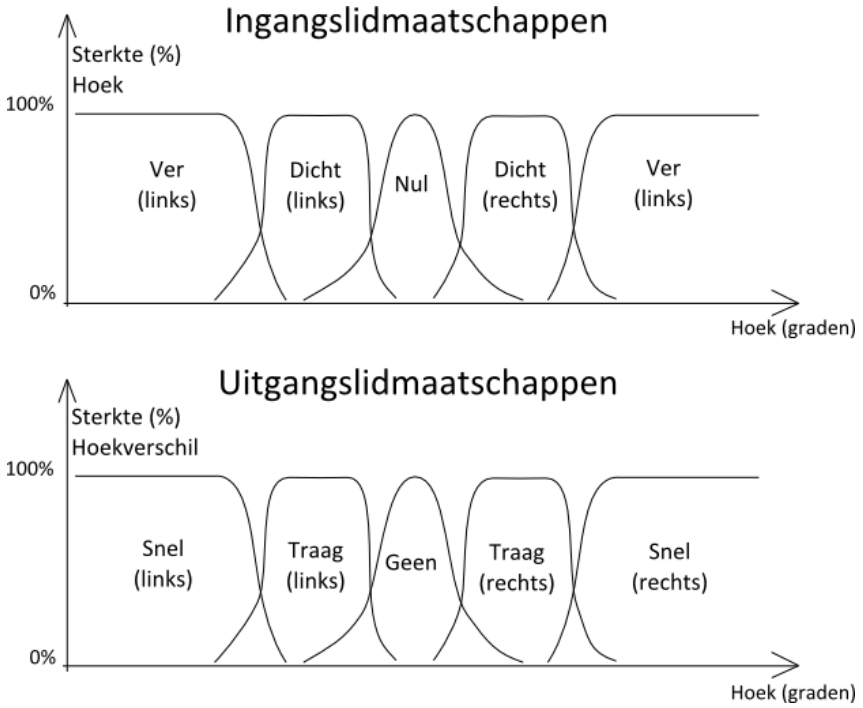
Het doel van de Fuzzy Logic animatiecontroller is net als bij IK de end effector(de hand in dit geval) te doen samenvallen met de doelpositie. Het verschil zit hem echter in het feit dat een Fuzzy Logic controller dit niet in één stap doet. De controller zal de end effector begeleiden richting de doelpositie tot deze bereikt wordt. Om dit voor een keten met meerdere joints te bereiken wordt het probleem opgesplitst. Men kijkt voor iedere joint hoever deze nog moet roteren tot de doelpositie, deze waarde wordt door een Fuzzy Logic controller gestuurd, die specifiek ingesteld is voor die joint. Het bekomen resultaat wordt gebruikt om de joint te doen roteren. In figuur 6.4 ziet men een versimpelde situatie van dit proces.



Figuur 6.4: Fuzzy Logic controller: manier van werken[6]

[6]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

De ingangswaarde wordt beschreven als de hoek, rond deze joint, die nog af te leggen is door de end effector tot de doelpositie bereikt wordt. De uitgangswaarde is de hoek die voor dit ogenblik moet geroteerd worden. Deze berekening en rotatie wordt een aantal keer per seconde herhaald om beweging te creëren, hierdoor kan de uitgang van de Fuzzy Logic controller aanzien worden als een rotatiesnelheid. Dit maakt van de fuzzy animatiecontroller een snelheidsgebaseerde controller. De gewrichten in het menselijk lichaam roteren niet aan een constante snelheid rond elke richting. Door de rotatie te ontbinden in hoeken van Euler en een aparte Fuzzy Logic controller toe te kennen aan ieder van deze hoeken kan een realistischere rotatie bekomen worden. Elk van deze Fuzzy Logic controllers werkt volgens hetzelfde principe, een grafische voorstelling van de lidmaatschappen van ingang en uitgang wordt weergegeven in figuur 6.5.



Figuur 6.5: Ingangs- en uitgangslidmaatschappen Fuzzy Logic controller bij gewrichten[7]

De regels voor dit type controller worden dan als volgt gemodelleerd:

```
ALS Hoek IS Ver(links) DAN Hoekverschil IS Snel(rechts)
ALS Hoek IS Dicht(links) DAN Hoekverschil IS Traag(rechts)
ALS Hoek IS Nul DAN Hoekverschil IS Geen
ALS Hoek IS Dicht(rechts) DAN Hoekverschil IS Traag(links)
ALS Hoek IS Ver(rechts) DAN Hoekverschil IS Snel(links)
```

[7]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Indien een rotatie-as van een gewricht niet aangestuurd moet worden, wordt er gewoon geen Fuzzy Logic controller aan verbonden. Bijvoorbeeld bij een scharniergewricht moet slechts één van de drie hoofdassen aangestuurd worden en is er dus ook maar één Fuzzy Logic controller nodig. Dit is een voordeel ten opzichte van de IK manier, aangezien waar geen rotatie moet optreden geen limitaties moeten uitgerekend worden. Voor de andere limitaties wordt er dezelfde techniek gebruikt als bij IK. Hetzelfde probleem als bij IK doet zich hier ook voor, de paden/doelposities worden best altijd zo gekozen dat de end effector ze kan bereiken.

6.3 Resultaten

In tabel 6.3 ziet men de animatie uiteengezet voor twee Inverse Kinematic animatiecontrollers, die beiden gebruik maken van een CCD IK solver. In de verticale richting staan de verschillende fases van de handdruk, in de horizontale richting het verloop van iedere fase.

	Begin	Midden	Eind
Dummy			
Approach			
Shake			
Retreat			

Tabel 6.3: Resultaat CCD-CCD

In dit stuk worden verschillende configuraties van de handdruk uiteengezet. Allereerst worden verschillende configuraties van animatiecontrollers bekeken, vervolgens wordt getoond wat de invloed is van positie en grootte van de karakters op de animatie. Om dan te eindigen met een meer algemene beoordeling op basis van een aantal criteria.

6.3.1 Configuraties van animatiecontrollers

Aangezien de invloed van animatiecontroller minder goed merkbaar is op globaal vlak, worden bepaalde aspecten van de animatie nader bekeken. Deze worden telkens vergeleken voor de verschillende animatiecontrollers. In tabel 6.4 ziet men de vier configuraties die besproken worden.

	Karakter links	Karakter rechts
Configuratie 1	IK (JacobianTranspose)	IK (JacobianTranspose)
Configuratie 2	IK (CCD)	IK (CCD)
Configuratie 3	Fuzzy Logic	Fuzzy Logic
Configuratie 4	Fuzzy Logic	IK (CCD)[8]

Tabel 6.4: Configuraties van animatiecontrollers

Er zal voornamelijk geconcentreerd worden op de handgreep aangezien deze zowel op vlak van positie en oriëntatie beschreven is. Dit is een goede referentie om de precisie van ieder algoritme te beoordelen.

Configuratie 1: JacobianTranspose

In figuur 6.6 ziet men het voor- en bovenaanzicht van de handgreep voor twee tijdstippen binnen de shakefase.

Figuur 6.6: Configuratie 1: Handgreep[9]

[8]De keuze van IK algoritme heeft hier weinig belang, CCD werd hier willekeurig geselecteerd. De samenwerking tussen Fuzzy Logic en IK is belangrijker voor de beoordeling van deze configuratie.

JacobianTranspose heeft een hoge precisie doorheen de volledige fase. Doorheen de volledige handdruk animatie heeft JacobianTranspose geen uitzonderlijke afwijkingen. JacobianTranspose haalt zijn precisie met een grotere kost. De uitvoeringstijd van één iteratie is relatief hoger dan die van andere algoritmen. (Zie tabel 6.5)

Configuratie 2: CCD

In figuur 6.7 ziet men de handgreep met CCD op twee tijdstippen.

Figuur 6.7: Configuratie 2: Handgreep[10]

CCD heeft een lagere precisie dan JacobianTranspose, maar de uitvoeringstijd is dan ook lager. (Zie tabel 6.5) Er is ook een gelijke afwijking bij zowel het begin van de approachfase als het eind van de retreatfase, zoals weergegeven in figuur 6.8.

[9]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[10]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Figuur 6.8: Afwijking CCD[11]

Deze afwijking is waarschijnlijk het gevolg van de manier waarop het CCD algoritme werkt. Het algoritme probeert de end effector uit te lijnen met de doelpositie. Dit kan opgelost worden op twee manieren: de begrenzing van de pols vernauwen of de oriëntatie van de hand vastleggen op de andere fases net als bij de shakefase.

Configuratie 3: Fuzzy Logic

In figuur 6.9 ziet men de handgreep met Fuzzy Logic op twee tijdstippen.

Figuur 6.9: Configuratie 3: Handgreep[12]

[11]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Hier ziet men wederom een lage precisie, deze kan verbeterd worden door de Fuzzy Logic controller te verfijnen. Deze verfijning wordt best gedaan met hulp van een animator. Aan het eind van de approachfase is er een afwijking, zoals aangegeven in figuur 6.10.

Figuur 6.10: Afwijking Fuzzy Logic[13]

De Fuzzy Logic controller werkt op basis van verschilhoek tussen end effector en doelpositie, dit betekent dat torsie in de gewrichten niet in rekening wordt gebracht. Dit effect is duidelijk zichtbaar in de approachfase, het wordt echter opgelost door de geforceerde rotatie tijdens de shakefase. De Fuzzy Logic kan in de toekomst uitgebreid worden om dergelijke informatie te verwerken en te gebruiken om torsie op een correcte manier toe te passen. Hierbij is ook voor de eerste maal een ander probleem zichtbaar geworden. Aangezien bij de shakefase een vaste oriëntatie van de hand wordt opgelegd, zal de hand van de oriëntatie uit de approachfase 'springen' naar de oriëntatie uit de shakefase.

[12]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[13]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Configuratie 4: Gemengd

In figuur 6.11 ziet men de handgreep met links Fuzzy Logic en rechts CCD op twee tijdstippen.

Figuur 6.11: Configuratie 4: Handgreep[14]

Er is een duidelijk verschil in de manier waarop beide armen de approach en retreatfase overlopen, dit wordt niet aanzien als fout en is dus aanvaardbaar. Tijdens de shakefase is er enerzijds geen perfecte precisie zoals bij Fuzzy Logic en CCD apart al het geval was. Anderzijds is er ook afwijking in relatieve positie tussen beide handen tijdens de shakefase. Deze afwijking is te wijten aan het reactief gedrag van de Fuzzy Logic controller t.o.v. het exacte karakter van IK. De Fuzzy Logic controller reageert altijd met vertraging op een verandering in doelpositie. De reactietijd kan verbeterd worden door de snelheid van beweging te verhogen. Dit maakt de controller echter instabieler: er komt gevaar om de doelpositie voorbij te schieten. Er moet een evenwicht gezocht worden tussen snelheid en precisie bij het opbouwen van een Fuzzy Logic controller.

[14]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

6.3.2 Configuraties van karakters

In dit stuk wordt het effect van positie, schaal en persoonlijkheid van de karakters op de animatie bekeken. In figuur 6.12 wordt de invloed van variatie in positionering tussen beide karakters weergegeven in bovenaanzicht.

Figuur 6.12: Effect van translatie op de handdruk[15]

De rotatie en positie van de handdruk past zich telkens aan volgens het model beschreven in hoofdstuk 4. In figuur 6.13 ziet men de handdruk bij een verschil in schaal tussen beide karakters.

Figuur 6.13: Effect van schaal op de handdruk[16]

De handdruk vervormt zich mee indien de karakters veranderen van grootte. Hier is slechts de positie van de handen bij een handgreep niet optimaal, er moet nog onderzoek gedaan worden naar hoe handgrepen veranderen bij een verschil in grootte van handen. In figuur 6.14 wordt het effect van persoonlijkheidsverschil op de handdruk weergegeven.

[15]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[16]Gemaakt door Gaétan Deglorie voor gebruik in dit document.

Figuur 6.14: Effect van dominantie op de handdruk[17]

Beide karakters hebben een karaktertrek "dominantie", die van het linkse karakter is op een hogere waarde geplaatst dan van de ander. Hierbij wordt de handdruk verschoven richting het meer dominante karakter.

6.3.3 Algemene analyse

In tabel 6.5 ziet men een samenvatting van de analyse voor de verschillende types aan controllers.

	Inverse Kinematics		Fuzzy Logic
	CCD	JacobianTranspose	
Ticks per frame[18]	300	1250	800
Complexiteit animatie	matig	matig	laag
Complexiteit implementatie	laag	laag	laag-hoog

Tabel 6.5: Analyse animatiecontrollers

Het aantal "Ticks per frame" geeft aan hoeveel tijd de CPU nodig had om de animatiecontroller te updaten voor één frame. De waarden, die hier gegeven zijn, zijn benaderende waarden afgeleid uit metingen binnen het raamwerk. De uitvoeringstijd voor zowel Inverse Kinematics en Fuzzy Logic is afhankelijk van de ingestelde parameters. Voor Inverse Kinematics is dit afhankelijk van het aantal iteraties dat de berekening herhaalt wordt voordat het antwoord wordt doorgegeven. Dit verhoogt de stabiliteit en precisie van de gebruikte algoritmes. Voor Fuzzy Logic hangt de uitvoeringstijd vooral af van het gebruikte defuzzificatie algoritme. In het geval van een centroid defuzzifier wordt de oppervlakte van de afgesneden lidmaatschappen bepaald via integratie. De berekening hiervoor wordt vereenvoudigd door de Riemannsom te gebruiken. Het gekozen aantal deelintervallen bepaalt dan ook de uiteindelijke uitvoeringstijd van de volledige controller.

[17]Gemaakt door Gaétan Deglorie voor gebruik in dit document.
[18]Metingen uitgevoerd op een Intel(R) Core(TM) i7-2600 3.40GHz.

De complexiteit van animatie is lager bij Fuzzy Logic aangezien er kan gewerkt worden met statische doelposities t.o.v. de verplichte animatiepaden van Inverse Kinematics.

De complexiteit van implementatie bij Inverse Kinematics is laag indien men dit implementeert met behulp van een externe bibliotheek zoals in dit raamwerk. De complexiteit ligt vanzelfsprekend hoger indien men ervoor kiest om de algoritmen zelf te implementeren. De complexiteit van implementatie bij Fuzzy Logic is volledig afhankelijk van hoever men wil gaan. Binnen dit raamwerk is de implementatie vrij licht, maar indien gewenst kan men de controller uitbreiden.

Hoofdstuk 7

Conclusie en toekomstperspectieven

7.1 Conclusie

In dit werk werd een model vooropgesteld van een handdruk, dit uitgewerkt op twee manieren: Inverse Kinematics en Fuzzy Logic. Inverse Kinematics is een gekende techniek bij animatoren, Fuzzy Logic een techniek gekend in de regeltechniek maar uiterst weinig toegepast bij animatie. Beide technieken werden onderzocht voor hun potentieel op vlak van gebruik bij modelleren van interactieanimaties.

Inverse Kinematics is een techniek die specifiek ontworpen is voor gebruik op ketens. Dit maakt het perfect voor het animeren van ledematen. Om tot een animatie te komen, moet de positie van de end effector bepaald zijn op ieder ogenblik. Dit wordt bereikt met behulp van animatiepaden. De getrouwheid van de animatie is voor een groot stuk afhankelijk van deze paden. Door zijn specifieke toepassing is de techniek echter moeilijk uitbreidbaar en minder breed toepasbaar.

Fuzzy Logic wordt desondanks zijn brede toepasbaarheid zelden tot nooit toegepast bij animaties. In tegenstelling tot IK kan het op meer dan alleen ledematen worden gebruikt. De implementatie die hier werd vooropgesteld kan gebruikt worden met of zonder gebruik van animatiepaden. De numerieke ingangs- en uitgangsvariabelen maken het moeilijker om toe te passen, hiervoor is wat voorverwerking nodig.

Het animeren van interacties vergt extra logica binnen het raamwerk. Deze logica maakt het mogelijk om interacties te starten, accepteren en weigeren. Om de handdruk hierbij te gebruiken, moet deze opgesplitst worden in fases. Of dit nodig is voor andere interactieanimaties wordt overgelaten als toekomstig werk.

Dit werk probeert een algemeen model naar voor te brengen, het probleem is echter dat er een gebrek aan standardisatie is binnen Computer Graphics en meer specifiek animatie. Skeletten van karakters worden vaak gedefinieerd afhankelijk van de applicatie, het aantal gewrichten kan dus sterk variëren van project tot project. Desalniettemin kan dit werk

vrij eenvoudig aangepast worden aan de variërende behoeftes van deze projecten.

7.2 Toekomstperspectieven

Het vooropgestelde model van de handdruk is ver van perfect, om deze te kunnen verbeteren moet er een grotere casestudy gedaan worden over handdrukken. Zo kan men preciezere resultaten bekomen en daarbij een realistischer model definiëren. Bepaalde aspecten die beter moeten beschreven worden: mogelijke niet-lineariteit van de shakefase, invloed van dominantie op de vorm van de shakefase, toekenning van (nieuwe) karaktertrekken aan alle dimensies van de handdruk.

Om de animatie te vervolledigen moet hand, of meer specifiek de vingers, ook nog geanimeerd worden. Dit kan weer als een volledig onderzoek opgenomen worden, waarbij specifieke controllers kunnen gemodelleerd worden om objecten van verschillende vormen mee vast te grijpen. In zake dit model wordt er best ook nog onderzoek gedaan naar handgrepen met handen die sterk verschillen in grootte, zoals een kind die een hand geeft aan een volwassene.

Aangezien de animaties die hier zijn ontworpen specifiek inwerken op de rechterarm en niet verder, kan ervoor gekozen worden om blending toe te passen. Hierbij zou de handdruk animatie samengevoegd worden met een neutrale animatie[1] via blending.

Door het opsplitsen van de animatie in fases is bijvoorbeeld het synchroniseren vergemakkelijkt, het maakt echter de overgangen tussen de fases een stuk onnatuurlijker. Indien de oriëntatie van de hand in een fase vastlegt maar in de fase ervoor niet, dan zal bij de overgang de hand springen van één rotatie naar een ander. Op de overgangen moet er dus meer aandacht besteed worden aan de gewrichtsposities/rotaties, een voorstel voor deze overgangen natuurlijk te maken is door gebruik te maken van blending. Of alternatief door de oriëntatie te parametriseren voor de volledige animatie (alle fases) en deze altijd toe te passen.

De toepassing van Fuzzy Logic bij animatie bevindt zich op ongekend terrein. Uit dit werk is gebleken dat de techniek veel potentieel heeft. Enerzijds kan de controller, die hier beschreven is, geoptimaliseerd worden. Anderzijds kan er verder onderzoek gedaan worden naar andere toevoegingen die de animaties verder kunnen verbeteren. Een voorstel hiervoor is de toevoeging van snelheidsverschil als uitgangsvariabele. Dit laat toe om acceleratie toe te voegen aan het model waardoor de fysieke geloofwaardigheid van de animatie verbetert. Deze uitbreiding kan ook bekomen worden via IK, door het tijdsverloop van animatiepaden te parametriseren.

[1] Dit is beter gekend onder de naam: "*Idle Animation*".

In hoop van de stabiliteit van de Fuzzy Logic controller te verhogen, is men gekomen op een nieuw systeem. Dit systeem zou het cyclische aspect van CCD gebruiken, waarbij de Fuzzy Logic controller itereert over de verschillende joints en telkens alle ingangen en uitgangen herberekent bij iedere stap. Hier moet echter nog meer onderzoek naar gedaan worden.

Tijdens het evalueren van de handdruk voor de gemengde configuratie (IK en Fuzzy Logic) werd er een slechte precisie opgemerkt. Aangezien IK van nature een hoge precisie haalt is een nieuw systeem bedacht geweest. Waarbij er gewerkt kan worden met een *master-slave* structuur. Fuzzy Logic zou de master animatie uitvoeren en IK zou als slave de hand van de andere als doelpositie gebruiken tijdens de shakefase. Dit zou potentieel een hogere precisie leveren.

Ten slotte was het doel van dit onderzoek om methoden te vinden om interactieanimaties te modelleren. Dit is specifiek toegepast geweest op handdrukken. Deze animatie is van een specifiek subtype en geldt enkel voor twee karakters, de implicaties van het animeren van meer dan drie karakters volgens dit model zijn nog onbekend. De weg is nu geopend om verder dit type van animatie te onderzoeken en implementeren.

Bibliografie

[1] J. Stuart Blackton. Humorous phases of funny faces. `https://archive.org/details/Humorous_Phases_of_Funny_Faces_1906`. Geraadpleegd November 2013.

[2] Rick Parent. *Computer Animation, Third Edition: Algorithms and Techniques*. Morgan Kaufmann Publishers Inc., 2012.

[3] R.R. Everett. The whirlwind i computer. `research.microsoft.com/en-us/um/people/gbell/Computer_Structure__Readings_and_Examples/00000157.htm`. Geraadpleegd November 2013.

[4] Simon Vreeswijk. A history of cgi in movies. `www.stikkymedia.com/blog/history-cgi-movies`. Geraadpleegd November 2013.

[5] D.S. Cohen. Cathode-ray tube amusement device - the first electronic game. `classicgames.about.com/od/classicvideogames101/p/CathodeDevice.htm`. Geraadpleegd November 2013.

[6] Stelian Coros et al. Generalized biped walking control. *ACM Transactions on Graphics*, 29(4):Article 130, 2010.

[7] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. In *IEEE Journal of Robotics and Automation*, pages 681–685, 2004.

[8] L.-C.T. Wang en C.C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991.

[9] Marcelo Kallmann. Analytical inverse kinematics with body posture control. *Comput. Animat. Virtual Worlds*, 19(2):79–91, May 2008.

[10] Seyoon Tak en Hyeong-Seok Ko. Example guided inverse kinematics. In *Conference on Computer Graphics and Imaging - CGIM*, Las Vegas, Nevada, USA, 2000.

[11] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, August 2004.

[12] Eimei Oyama et al. A modular neural network architecture for inverse kinematics model learning. *Neurocomputing*, 38-40(0):797 – 805, 2001. Computational Neuroscience: Trends in Research 2001.

[13] Henk Scholten. *Regelen met Fuzzy Logic en PID*. Uitgeversmaatschappij Elektuur B.V., 1992.

[14] Lucas Kovar and Michael Gleicher. Flexible automatic motion blending with registration curves. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 214–224, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[15] Pascal Pohl. Motion graphs in character animation. *Saarland University*.

[16] Norman Badler. Virtual humans for animation, ergonomics, and simulation. In *IEEE Workshop on Non-Rigid and Articulated Motion, Puerto Rico*, Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE, pages 28 – 36, San Juan, Puerto Rico, 1997.

[17] Frank van Marwijk. Groeten bij komen en gaan. `http://www.lichaamstaal.nl/groet.html`. Geraadpleegd December 2013.

[18] Xna framework math conventions. `http://msdn.microsoft.com/en-us/library/bb203910.aspx`. Geraadpleegd November 2013.

[19] Digitalrune math conventions. `http://www.digitalrune.com/Support/Blog/tabid/719/EntryId/98/3D-Math-Conventions.aspx`. Geraadpleegd December 2013.

[20] Michael Zarozinski. Fcl. `http://ffll.sourceforge.net/fcl.htm`. Geraadpleegd Januari 2014.

[21] Jane Wilhelms and Allen Van Gelder. Fast and easy reach-cone joint limits. *J. Graph. Tools*, 6(2):27–41, September 2002.

[22] Donald D. Hearn et al. *Computer Graphics with Open GL*. Prentice Hall Press, 2010.

[23] Seongmin Baek, Il-Kwon Jeong, and In-Ho Lee. Implementation of virtual crowd simulation system. In *SICE-ICASE, 2006. International Joint Conference*, pages 2713–2716, Oct 2006.

[24] Linbo Luo, Suiping Zhou, Wentong Cai, Malcolm Yoke Hean Low, Feng Tian, Yongwei Wang, Xian Xiao, and Dan Chen. Agent-based human behavior modeling for crowd simulation. *Comput. Animat. Virtual Worlds*, 19(3-4):271–281, September 2008.

[25] Ramon Mas Sanso and Daniel Thalmann. A hand control and automatic grasping system for synthetic actors. *Computer Graphics Forum*, 13(3):167–177, 1994.

[26] Simon Pilgrim et al. Progressive skinning for character animation. *Computer Animation and Virtual Worlds*, 18(4-5):473–481, 2007.

[27] Taesoo Kwon et al. Two-character motion analysis and synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):707–720, 2008.

[28] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 251–260, New York, NY, USA, 2001. ACM.

[29] Ronan Boulic, Nadia Magnenat-Thalmann, and Daniel Thalmann. A global human walking model with real-time kinematic personification. *Vis. Comput.*, 6(6):344–358, November 1990.

[30] Boon-Seng Chew et al. A fuzzy clustering algorithm for virtual character animation representation. *IEEE Transactions on Multimedia*, 13(1):40–49, 2011.

[31] Zsofia Ruttkay, Zhisheng Huang, and Anton Eliens. The conductor: Gestures for embodied agents with logic programming. In *Proceedings of the 2nd Hungarian Computer Graphics Conference*, pages 9–16, 2003.

[32] Marcelo Kallmann en Daniel Thalmann. Modeling objects for interaction tasks. In *Proc. Eurographics Workshop on Animation and Simulation*, pages 73–86, 1998.

[33] Craig Reynolds. Steering behaviors for autonomous characters. In *Proceedings of Game Developers Conference*, pages 763–782, 1999.

[34] John Funge, Xiaoyuan Tu, and Demetri Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 29–38, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[35] Rachel Heck en Michael Gleicher. Parametric motion graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, pages 129–136, New York, NY, USA, 2007. ACM.

[36] Jen-Yao Chang and Tsai-Yen Li. Simulating virtual crowd with fuzzy logics and motion planning for shape template. In *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*, pages 131–136, Sept 2008.

[37] Christopher Peters, Simon Dobbyn, Brian MacNamee, and Carol O'Sullivan. Smart objects for attentive agents. In *WSCG*, 2003.

[38] Samuele Vacchi, Giovanni Civati, Daniele Marini, and Alessandro Rizzi. Neo euclide: A low-cost system for performance animation and puppetry. In *Gesture Workshop*, pages 361–368, 2003.

Lijst van figuren

Lijst van tabellen