The Master of Science concentration in Digital Technologies (MS_DT) was initiated in 2011 as a post-professional degree that offers motivated participants the opportunity to investigate design practices and conduct independent research in computer-aided-design and advanced fabrication techniques. Project-based research provides a "testing ground" for new modes of practice and innovative uses of existing, new and emerging technologies. The program builds upon a tradition of cutting-edge technical research at Taubman College, the University of Michigan, and in the Detroit region. University of Michigan offers unmatched excellence in digital fabrication and access to world-class lab and production facilities and regional linkages to industry. Each issue is assembled by the individual author/architect during their duration at the University of Michigan.

2012

Karl Daubmann
(MS_DT Coordinator)

Mark Meier
Jason Prasad
Mat Schwartz
Ryan Shaban
Fausto Teran
Sonia Tereszczenko
Aaron Willette
Robert Yuen

Universal Design Manikin: Integrative Simulation and Visualization Techniques | Mathew Schwartz

Universal Design Manikin

Integrative Simulation and Visualization Techniques

Mathew Schwartz

MS_DT

Master of Science in Digital Technologies

A. Alfred Taubman College of Architecture and Urban Planning
University of Michigan

MS_DT | 2012

Universal Design Manikin

Integrative Simulation and Visualization Techniques

by Mathew Schwartz

BFA 2011
University of Michigan

Submitted to the Department of Architecture and Urban Planning
in Partial Fulfillment of the Requirements for the Degree of
Masters of Science in Digital Technologies
at the Taubman College of Architecture and Urban Planning

July 2012

Mathew L. Schwartz _____

Universal Design Manikin

Integrative Simulation and Visualization Techniques

by Mathew Schwartz

Submitted to the Department of Architecture and Urban Planning
on July 2012 in partial fulfillment of the requirements for the
degree of Masters of Science in Digital Technologies

ABSTRACT

This research demonstrates methods for integrating simulation and visualization techniques with the current tools used in design work-flows. The techniques are applied to human factors with a concentration on disabilities. A tool named Universal Design Manikin is developed. The tool integrates a virtual manikin and wheelchair with a coresponding graphical user interface. The research covers factors from a human scale of reach abilitiy to a large scale of building navigation. The research presents an opportunity for seamless collaboration between scientists and designers by integrating joint analysis tools with design tools. Methods for simulation and visualization of reach, vision, navigation, and spatial zones are presented.

Index

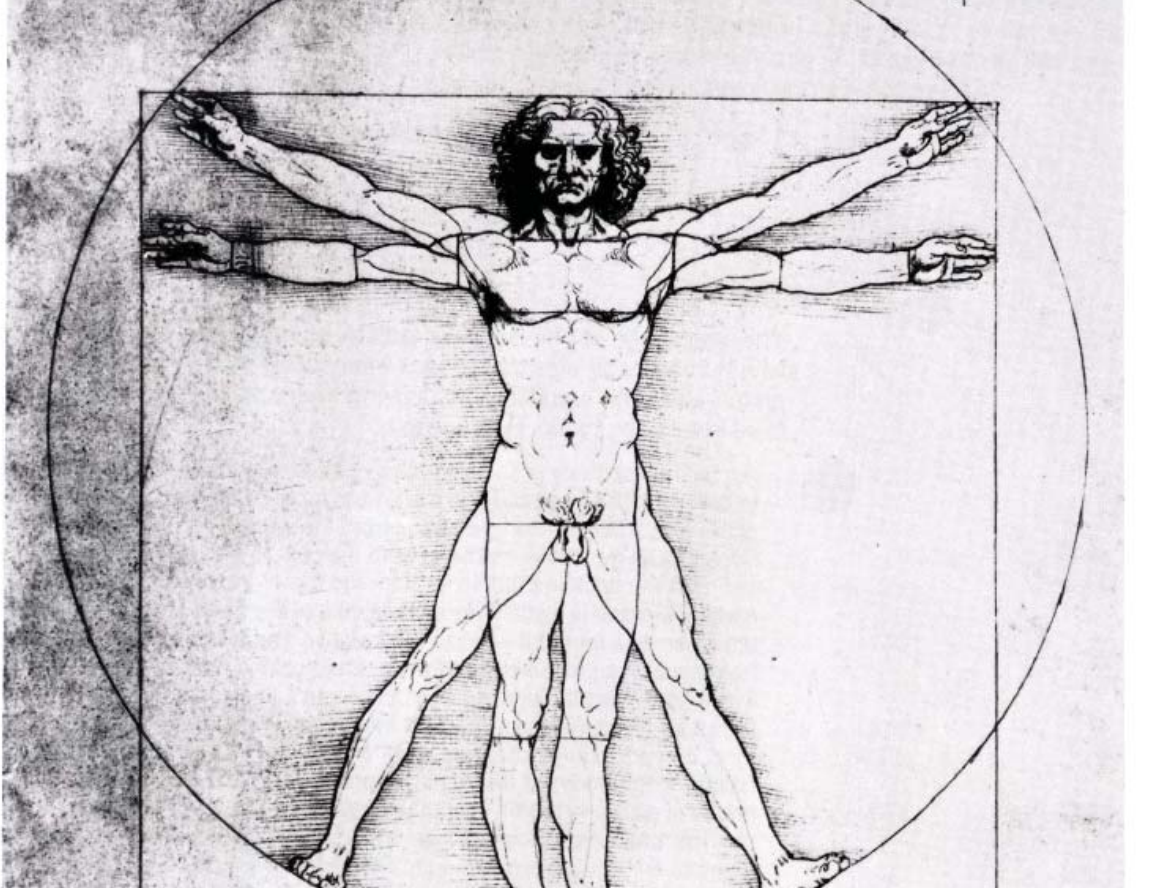


Image Credit: (Zelnik)

Preface

This book is a compilation of human based design research. It covers issues of computation, biomechanics, and overall work-flow. Motivation behind the research into those elements derives from a basic question; What drives design? While the factors that drive design can vary, the following pages outline a historical precedent for design based on the human body. The history behind human design is broken into three time periods; and each of these periods come with a system in which design has been based on. This consistent use of a system greatly motivated the work to move past the creation of a specific design concept and onto a larger issue of creating a platform in which collaboration and human based design can thrive. The following is a brief introduction to some key points related to the history of human based design.

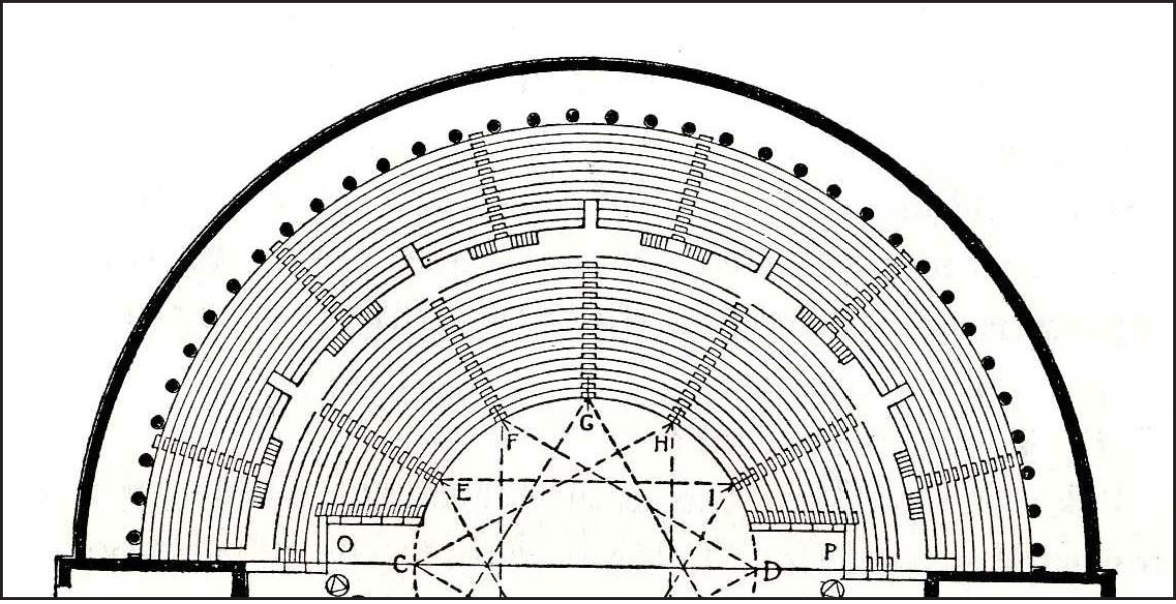


Image Credit: (Pollio)

1|1 Vitruvian Design

"How could the relation of Man to God be better expressed, we feel now justified in asking, than by building the house of God in accordance with the fundamental geometry of square and circle?"
-Wittkower

The start of a human-form centered architectural field can be traced back to Marcus Vitruvius in 1 BCE. His work entitled De Architectura, was broken into 10 books dealing with various aspects of architecture. Although many have seen the Vitruvius man diagram from Leonardo Da-Vinci, few know the history of that diagram. Vitruvius began the first chapter of his third book stating, "Without symmetry and proportions there can be no principles in the design of any temple; that is, if there is no precise relation between its members, as in the case of those of a well shaped man."(Pollio). This comparison of members in the temple to the temple itself is a precedent for the religious work that followed in the Renaissance. Vitruvius continued to lay the groundwork of what Da-Vinci would later make infamous through art, by describing the proportional relationship of man:

"For if a man can be placed flat on his back, with his hands and feet extended, and a pair of compasses centered at his navel, the fingers and toes of his two hands

and feet will touch the circumference of a circle described therefrom. And just as the human body yields a circular outline, so too a square figure may be found from it. For if we measure the distance from the soles of the feet to the top of the head, and then apply that measure to the outstretched arms, the breadth will be found to be the same as the height, as in the case of plane surfaces which are completely square." (Pollio)

It was not until over a thousand years later De Architectura was rediscovered. The book was then popularized and with the rise of the Renaissance, the drawings of the Vitruvian man appeared. It was in this context that Leonardo Da-Vinci created the well known diagram.

It is important to note the comparison between a human-form centered design, in which the design revolves around the form of a human, and an anthropocentric design in which the design is related to human ergonomics. This distinction is important when looking at work from the Renaissance. The wording of Vitruvius and many representations of Renaissance buildings appear to relate the building to the human, but at closer look they are truly about the divine world. The comparison of the Vitruvius man to architectural works of the Renaissance is quite simplified. Alberti's De re Aedificatoria outlines nine geometries that fall in line with the circle that the Vitruvian man describes (Figure 1.1). While using the circle as a base, the proposed geometries are based off this circle, varying angles of the geometry based on radius's of the circle in which they inscribe. (Wittkower) This at first may seem

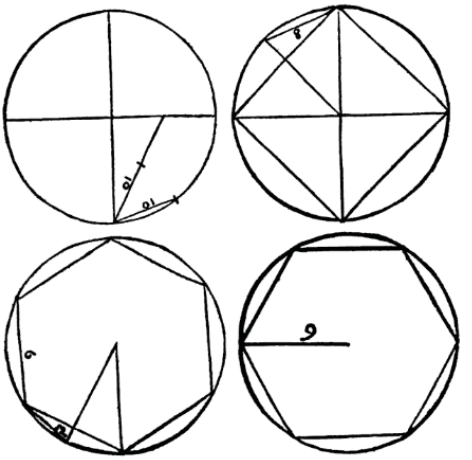


Fig. 1. Construction of square and polygons. From Bartoli's edition of Alberti's 'De re aedificatoria,' 1550.

Figure 1.1: (Alberti)

convoluted, but the rhetorical question posed by Wittkower sheds light on this situation, "How could the relation of Man to God be better expressed, we feel now justified in asking, than by building the house of God in accordance with the fundamental geometry of square and circle?"(Wittkower). So the comparison of human centered design from Renaissance to present day is better explained as human-form centered design to anthropocentric design.

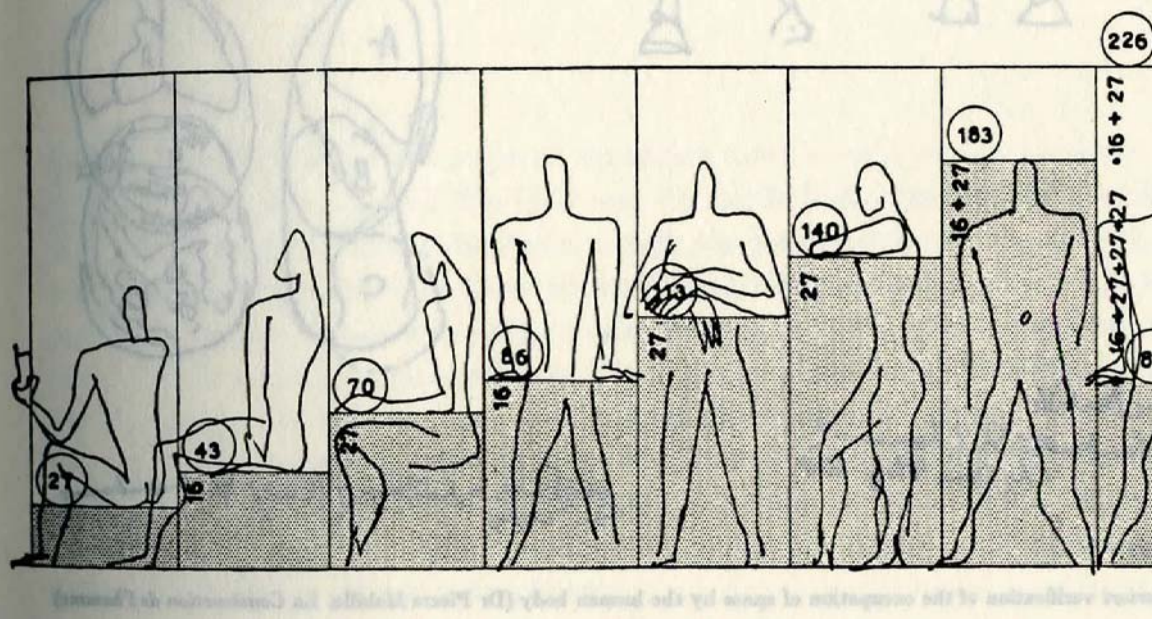


Image Credit: (Le Corbusier)

1|2 **Organic Design**

"If a man set out without knowledge of planning concepts but in identification with nature and thus nature-like, he will always act creatively".
-Hugo Harring

In the early 1900's Le Corbusier began to create a universal measurement diagram. His work on the Modulor attempted to set a standard for design in general. Le Corbusier however, was not necessarily looking at a geometric model that would constrain the designer entirely. Instead, Le Corbusier was looking at the modernization of manufacturing and its widespread reach. With the basis in proportion, Le Corbusier began to create a system in which measurements could unify the different unit systems. Le Corbusier's goal was to create "a common measure capable of ordering the dimensions of that which contains and that which is contained: capable...of offering a solid pledge of satisfaction to supply and demand" (Le Corbusier). While Le Corbusier followed up with a second book, Modulor 2, with many examples of his system being used and praised, the work of the Modulor created controversy in those loyal to free thinking design. The use of proportions (Figure 1.2) and ergonomic representations (Figure 1.3) gave the designer a freedom to work within a constraint, and in doing so focusing on the form rather than trying to understand how the form would function. This

led to a different school of thought, the belief that the use of these constraints limited the natural flow of a design.

Tangent to Le Corbusier, Hugo Haring discussed the relationship between functional and expressive design. He outlines the way in which buildings are separated by form and function. Haring explains that the design of functional forms will in turn satisfy the craving for expression, declaring that our appreciation for machines, ships, cars, and aircraft are routed in this satisfaction of functional forms (Jones). This understanding of design based on human ergonomics is a continuous topic of controversy as designers work through the notion of form following function. Haring argues that all forms are based on an internal path, "even crystals and geometrically-shaped [forms], which allows each to develop according to its own inner plan". This is contradictory to the work of the Renaissance in which the total form is the representation of the human. At the first level of thought, this look at design extending outwards may seem inline with the work of Le Corbusier as would standardize the human proportions in relation to chairs, tables, etc. The split emerges in what Haring describes as "planning concepts". Haring believed, "It is evident that in the same way, our potency in creating and building is limited by the potency of our planning concepts" (Jones). Haring proposes "If a man set out without knowledge of planning concepts but in identification with nature and thus nature-like, he will always act creatively".

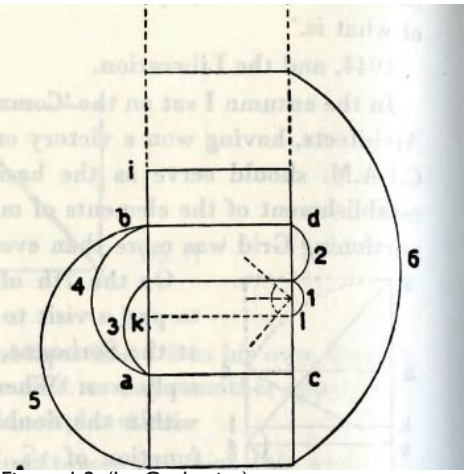


Figure 1.2: (Le Corbusier)

This relationship between a plan and nature draws back to the problem Le Corbusier was trying to

resolve, the emergence of the meter. While a design based on the anglo-saxon measurement of foot and inch were derived by the person directly, the meter is an abstract representation of distance. Le Corbusier himself notes the beauty in which building created on personal measurements, the inch and foot, convey (Le Corbusier). The design flow of the past, using personal measurements to design ones home, is inline with the argument made by Haring, and what Le Corbusier acknowledged as "infinitely rich" (Le Corbusier). However, Haring makes the case that the creation of a diagrammed system of measurements in its very nature hinders the natural flow of a design. He states that the men who act creatively through natural understandings are " in contrast with the men from geometric cultures, who, obsessed with order and limited in their planning concepts, could work fruitfully on so long as their creative effort was poured into the forms of geometry, subordinated to its laws and rules, and so constrained and destroyed.." (Jones).

The concepts created through the work of Le Corbusier and Harring provided extraordinary platforms for new waves of design. Within The Modulor 2 it is clear how drastic the effect of Le Corbusiers system was on modern design (Le Corbusier). To this day the proportions displayed by Le Corbusier are studied and investigated in design. However, the Metric system is widely adopted as the standard unit of measurement, the relative abstracted nature of it has faded, and the field of ergonomics has flooded the world with knowledge of true human proportions.

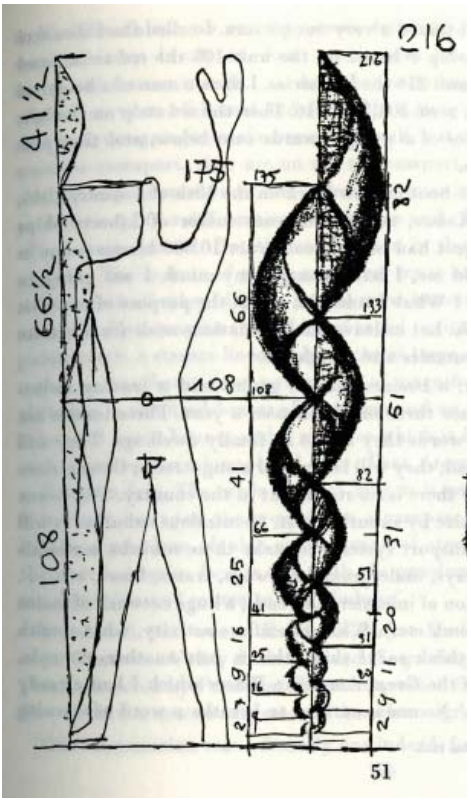


Figure 1.3: (Le Corbusier)

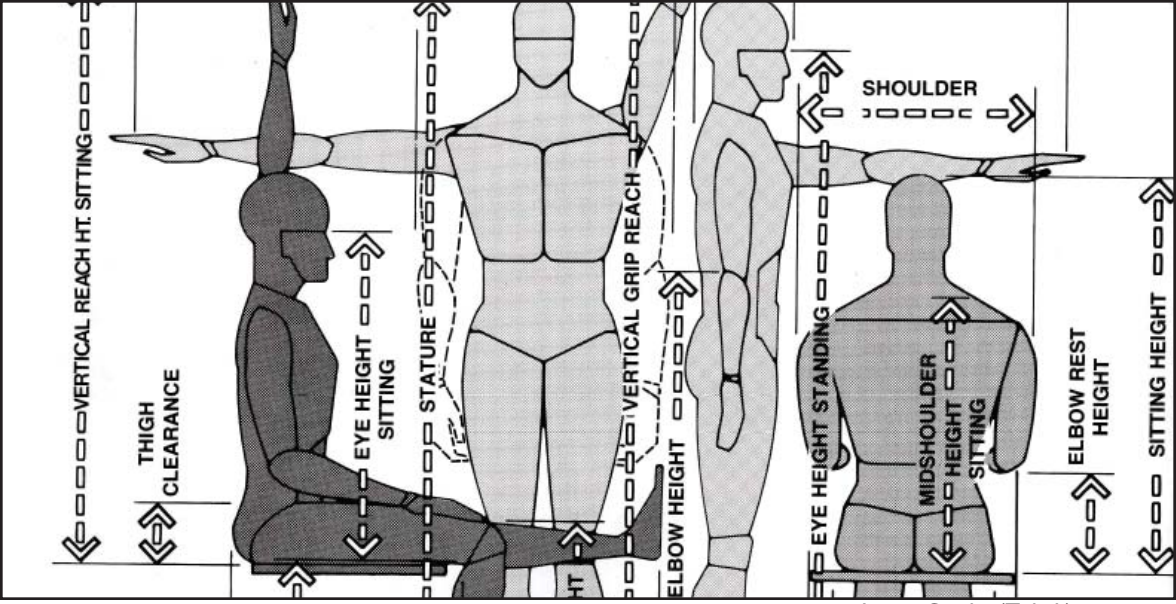


Image Credit: (Zelnik)

1|3 Universal Design

"There is no such thing as an average user"
-Susanna Laurin

The most recent human based design movement is Universal Design. While there are many alternative names, the concept stays the same. Universal Design can be understood through its seven principles:

1) Flexibility in use: The design accomodates a wide range of individual preferences and abilities
2) Tolerance for error: The design minimizes hazards and the adverse consequences of accidental or unintended actions
3) Low physical effort: The design can be used efficiently and comfortably and with a minimum of fatigue
4) Simple and intuitive: Use of the design is easy to understand, regardless of the user's experience, knowledge, language skills, or current concentration level
5) Equitable Use: The design is useful and marketable to people with diverse abilities

6) Perceptible information: The design communicates necessary information effectively to the user, regardless of ambient conditions or the user's sensory abilities
7) Size and space for use: Appropriate size and space is provided for approach, reach, manipulation, and use regardless of user's body size, posture, or mobility
(NCSU)

In contrast with the previous design platforms, Universal Design is segmented into functionality, independent of a specific person. By targeting the principles to function, Universal Design remains open to new understandings of the human body. This openness has helped break down discrimination against people with disabilities, and has done so without idealizing a specific situation.

The Universal Design principles are not specific to disabilities. The principles are meant to help all people through design. More importantly, Universal Design is relevant to all people since every body is different and everybody ages. As fast as people grow out of clothes, so do they grow out of certain designs, and into new ones. People also vary based on ethnicity. For example, the leg length of the average US black male is 2" longer than the average Japanese male (Tilley). Although 2" of leg length while standing may not matter, a chair designed for a US black male could result in a Japanese male not being able to place his feet on the ground (Figure 1.4). A more extreme example being a customized design for an average Northern Nilote would be far to big for a Pigmie as their averages are different by 15.4"(Figure

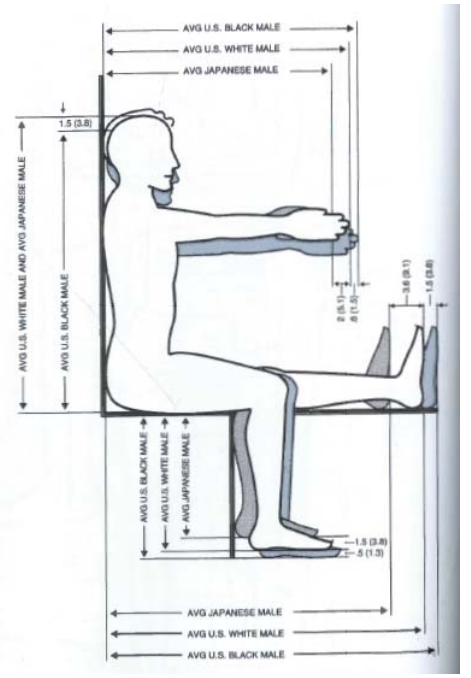


Figure 1.4: (Tilley)

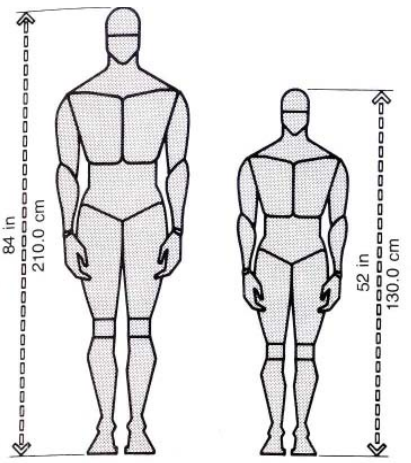
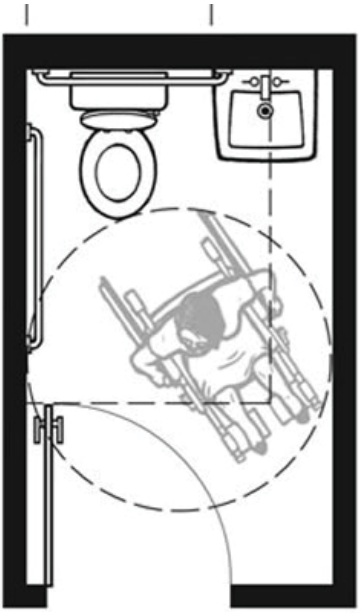


Figure 1–1. Comparison of difference in stature of the tallest Northern Nilote of Southern Sudan with the stature of the smallest Pigmy of Central Africa. Data from Chapanis, *Ethnic Variables in Human Factors Engineering*.

Figure 1.5: (Zelnik)

1.5). Beyond the relevance due to differences between all people, universal design can benefit everyone in unintended ways. For example, putting automatic door openers for wheelchair accessibility has the added benefit of making it easier for someone carrying objects to open the door. Another example is that by providing ramps, all rolling objects, from grocery carts to strollers are easier to use.

In 1990 the creation of the American with Disabilities Act (ADA) made Architects and Employers legally responsible for accessibility. Outlined in a series of legal provisions and diagrams by the Department of Justice, the ADA set a new standard, requiring equal access for people with disabilities. Unlike Universal Design, the ADA needed to consist of specific examples in order to insure a suitable implementation. These examples are considered by the ADA as the minimum requirements. This raises a major problem; without understanding the origin of these examples, how can one design for something to be better than the minimum?

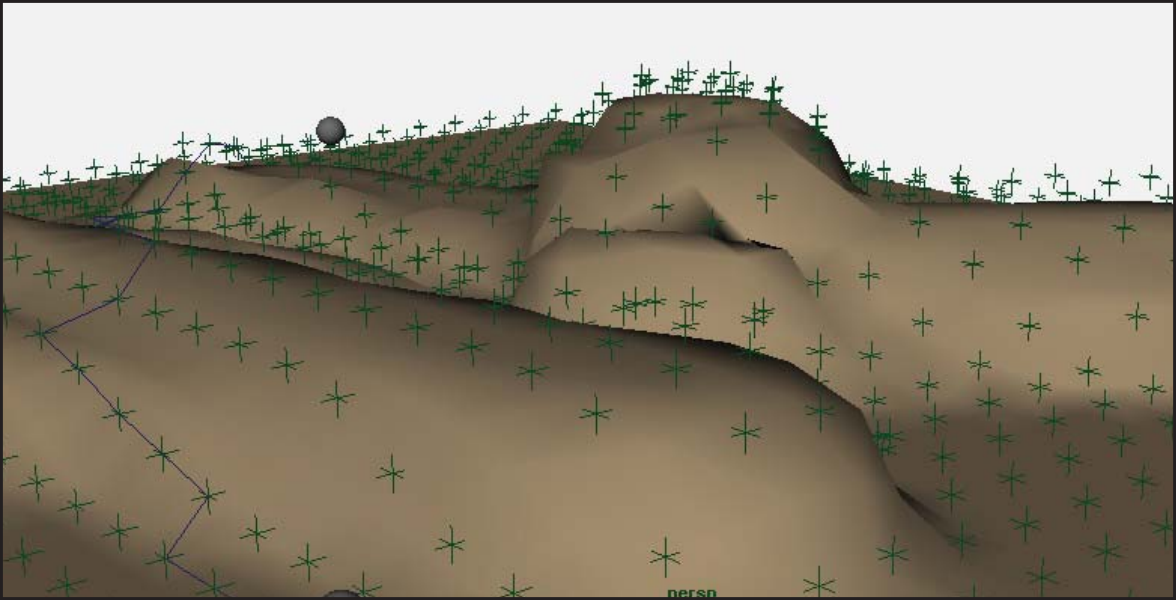


Plan-2A: 1991 Standards Minimum with In-Swinging Door

Figure 1.6: (ADA)

2 Introduction

Through the motivation of human centered design the intention of this work is to use visualization and simulation techniques to aid in the design process. The result of the research provides a platform for which movement scientists and designers can collaborate, and design students can explore and learn about human factors. The platform exists through a tool implemented in a 3D modeling program. The tool consists of different algorithms for simulating and visualizing human factors, file importation of motion capture data, and plotting functions to visualize human joint information. The specific functionality of the tool and algorithms is elaborated in later chapters. In this introduction the key components, simulation, visualization, collaboration, and education are introduced as they relate to the research.



2|1 **Simulation**

sim·u·la·tion
noun

The imitative representation of the functioning of one system or process by means of the functioning of another

The distinction between virtual and physical is increasingly difficult to understand. Through advancements in technology and mathematics we are constantly increasing our ability to recreate the world into a new, Synthetic world. This ability, known as Simulation, helps save lives, money, and provides valuable insight not gained any other way. As a designer, simulation can provide insight as to how a material will perform under stress, how a light will effect a space, or how a crowd of people will exit a building. While simulation has many applications, this book is interested specifically with the ability to emulate human factors in relation to design.

Within the synthetic world of simulation one can strive to emulate the real world, or create new worlds. It is this vast ability of simulation that requires a focused understanding of what a specific simulation does. To achieve this understanding, a series of diagrams can be created to represent the components. To start, a simulation consists of an input, a function, and an output. (Figure 2.1)

. This system can be applied to a wide spectrum of situations ranging from the real (physical world) to abstract (computer simulation) (Figure 2.2).

The spectrum that human factors simulation exists within is much smaller than environmental simulation. With human factors, the information is easier to access in the real world, creating a closer link between the input and output. This work explores two types of simulations for human factors, simulations of the human body and simulations of the body relative to an environment. To contextualize the simulation options the environmental simulation map of Simulating Future Worlds (Figure 2.3) is re-appropriated to the simulation of human factors (Figure 2.4).

Not all simulations are good for all types of information. The combination of inputs and functions effects the end output and it is important to know how. A major fallback with many commercialized human simulation models is the closed access to the functions. Although a human model may seem correct, there may be additional factors that have been left out of the calculation. As such, it can be dangerous to accept the information given by a closed simulation product. The use of a simulation graph creates an easily traceable result with the inputs and function properly understood. For this work, the simulation graph is used to outline places in which the output can be improved through different simulation methods.

Figure 2.1

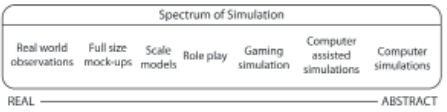


Figure 2.2

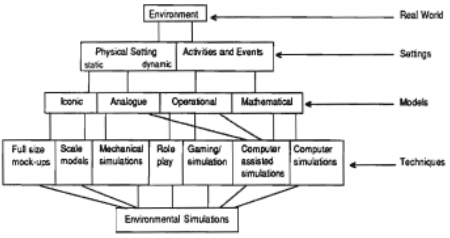


Figure 2.3: (Clipson)

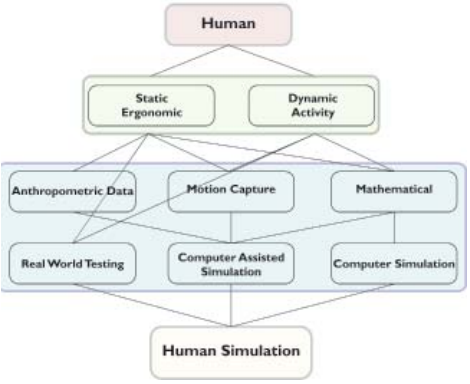
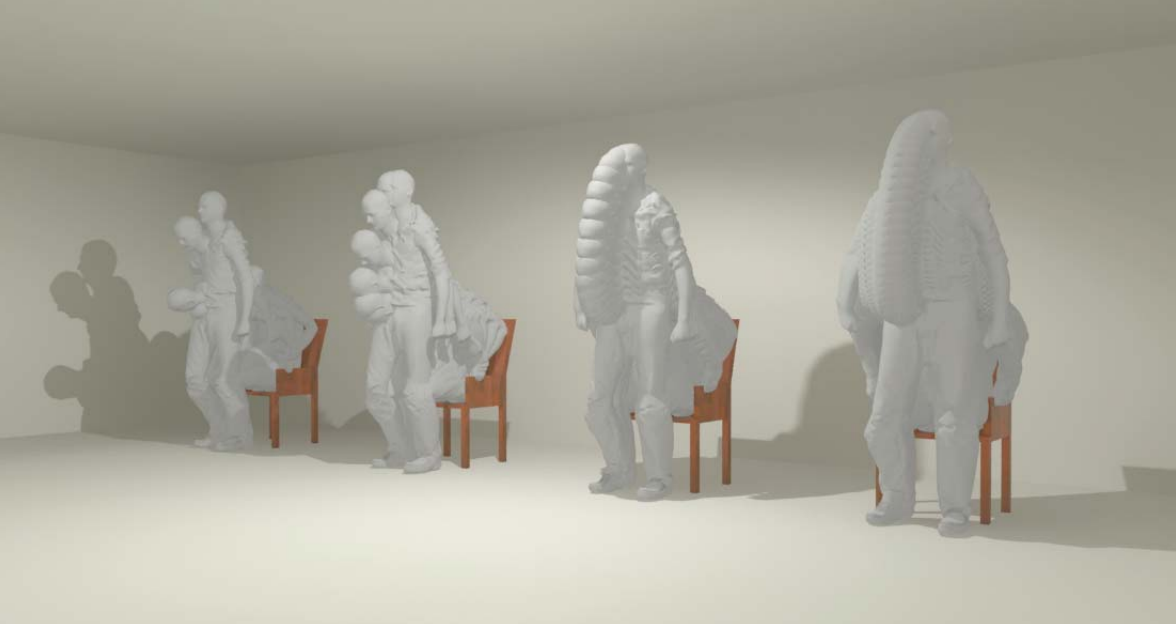


Figure 2.4



2|2 **Visualization**

vi·su·al·i·za·tion
noun

The act or process of interpreting in visual terms or of putting into visible form

Visualizations are how the information gathered from the simulation are represented. These visualizations are created by combining the information given from a simulation, and the purpose of that information relative to human factors. By this intent, there are many ways of achieving the visualization. Understanding the different types of visualizations relative to the human body helps decide how to best display the information. Different fields may refer to their visualizations with different terminology, such as dance referring to it as notation (Laban). Other terms such as diagramming have a slightly different intention. While the visualizations used in this book and many others are ways of representing information, or abstracting the human body into information. Diagrams however, are meant to explain information over just representing it.

Early stages of this research investigated the different methods of diagramming human factors. Through this research, it was clear that the design

field had remained stagnant for quite some time. In order to understand the reasoning for this lack of innovation in human representation, two additional disciplines, Dance and Engineering, were looked at for methods of diagramming humans. The research was not limited to, and is not to be confused with representation of anthropometric data. All types of human diagrams within Dance and Engineering were studied. It was through this exploration that the decisions on how to represent information on human factors within a tool were made.

Dance, a field entirely based around the human body, demonstrates creative ways of diagramming human movement. With two of its systems, Laban Notation and Benesh Movement Notation, dance is able to diagram the movements of a dancer on paper (Page). Using symbols, Laban Notation can represent the direction of movement (Figure 2.5), the body part moving (Figure 2.6), the level of movement (Figure 2.7), and the length of time it takes to do the movement (Figure 2.8). These two systems are by name, notation, yet fundamentally diagram the movements of a dance. This diagram is used to explain how a body should move in order to complete the dance sequence.

The past example demonstrates a method in which a chain of human actions can be explained through a diagram. These diagrams exist as a set of symbols that can be combined in a linear fashion corresponding with the change in time. Alternatively, the display of human information can be approached through an abstraction of humans into machines. The term machine is used as a way to describe

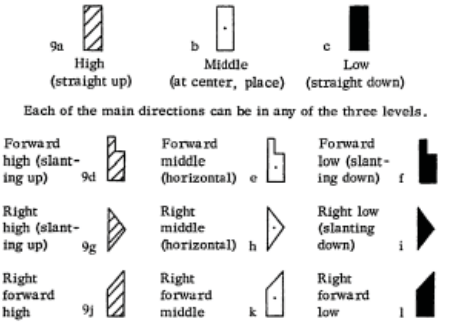


Figure 2.5: (Hutchinson)

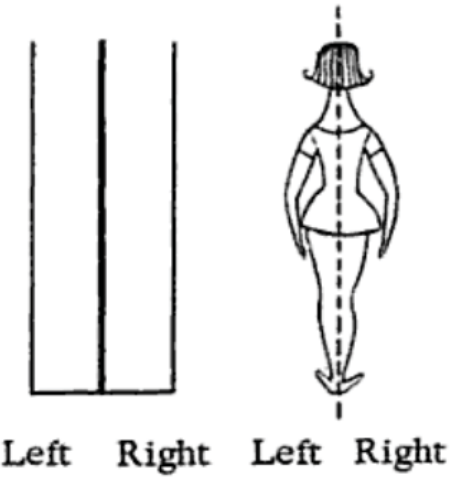


Figure 2.6: (Hutchinson)

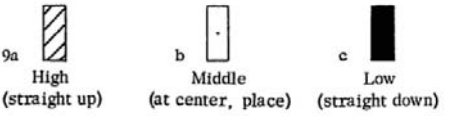


Figure 2.7: (Hutchinson)

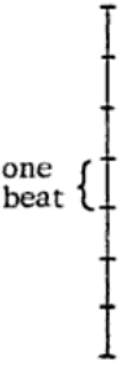


Figure 2.8: (Hutchinson)

the abstraction of the physical person into a systematic or mathematical model that describes a human or a persons actions. On a singular scale, a human action can be described through a mathematical model. As one example, the psychologist Paul Fitts proposed a model for human-computer interaction that predicts the time required to move from one point to another (Figure 2.9). Fitts's model is used to predict the act of pointing (Fitts). This sixty year old mathematical model translated so well to the age of computers that it profoundly impacted the design of the Xerox mouse (Card). The efficacy of this model is in its ability to simulate a persons pointing speed. While the function itself can be utilized as a diagram for a persons pointing speed, it is the usefulness of this function, as part of a system able to simulate, that makes it so powerful.

While dance benefits from an entire diagramming system and engineering has exploited mathematics, design has found more usefulness in larger statistical datasets (Figure 2.10), some translated to graphical form (Figure 2.11). These graphics are by nature stagnant pieces of information in their printed form (Tilley). Stagnant representations of human factors evolved from the large collection of anthropometric data. This data needed to be represented in a way that everyone could understand, as well as repeat. Standardization became key to people in human factors fields as it provided quick statistical information, useful for mass production and design. This anthropometric data was standardized through a series of methods in which the human body

Figure 1. **Fitts' reciprocal tapping paradigm (after Fitts, 1954).**

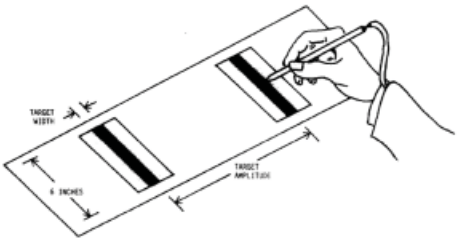


Figure 2.9: (MacKenzie)

Measurement	Number	Mean	SD	Percentiles						
				1st	5th	10th	50th	90th	95th	99th
Weight (lb)	130	152.49	23.19	112	119	124	151	184	192	204
Stature	119	66.28	2.09	61.6	63.3	63.7	66.1	68.3	69.9	70.3
Sitting height, erect	119	34.77	1.21	32.5	33.0	33.2	34.7	36.5	37.0	37.2
Sitting height, normal	131	33.42	1.45	29.7	31.0	31.6	33.4	35.2	35.9	36.5
Trunk height, sitting	131	22.57	1.24	19.8	20.5	20.9	22.7	24.3	24.5	24.9
Knee height, sitting	132	21.19	0.85	19.4	19.9	20.1	21.2	22.3	22.8	23.4
Popliteal height, sitting	131	17.31	0.83	15.4	15.7	16.3	17.2	18.4	18.6	19.2
Span	120	68.50	2.76	63.3	64.2	64.8	68.5	71.5	72.7	75.7
Span akimbo	121	35.69	1.52	32.4	33.4	33.8	35.7	37.3	37.9	39.4
Forward arm reach	118	34.21	1.51	31.2	31.7	32.3	34.2	36.1	37.0	38.4
Shoulder-elbow length	131	14.53	0.66	13.4	13.5	13.7	14.5	15.3	15.6	16.4
Elbow-middle finger length	130	18.27	0.71	16.9	17.2	17.4	18.3	19.3	19.5	20.4
Buttock-popliteal length	131	18.57	1.00	16.5	16.9	17.4	18.5	19.8	20.3	21.1
Buttock-knee length	132	23.26	0.96	21.0	21.8	22.1	23.2	24.6	25.0	25.4
Head length	133	7.74	0.25	7.1	7.3	7.4	7.7	8.0	8.1	8.3
Face length	127	4.96	0.27	4.4	4.6	4.6	5.0	5.3	5.5	5.6
Nose length	133	2.37	0.14	2.0	2.1	2.2	2.4	2.5	2.6	2.7
Ear length	132	2.94	0.19	2.5	2.6	2.7	2.9	3.2	3.3	3.4
Hand length	130	7.41	0.31	6.7	7.0	7.0	7.4	7.8	8.0	8.2
Foot length	132	10.24	0.39	9.2	9.7	9.8	10.2	10.8	10.9	11.3
Biacromial breadth	133	14.90	0.64	13.3	13.7	14.1	14.9	15.7	15.9	16.3
Bideltoid breadth	129	17.07	0.90	15.3	15.6	15.8	17.0	18.2	18.5	19.1
Chest breadth	133	11.84	0.81	9.9	10.2	10.6	11.7	12.7	13.0	13.4
Elbow-to-elbow breadth, sitting	132	17.81	1.32	15.0	15.5	16.2	17.8	19.3	20.1	21.0
Bi-iliac breadth	132	12.28	0.67	10.9	11.2	11.4	12.3	13.2	13.5	13.9
Hip breadth, sitting	131	14.87	0.94	13.2	13.5	13.7	14.8	16.1	16.7	17.2
Knee-to-knee breadth, sitting	129	8.07	0.52	7.3	7.5	7.6	8.0	8.5	8.7	10.1
Head breadth	133	6.07	0.20	5.6	5.8	5.8	6.1	6.3	6.4	6.5
Face breadth	132	5.55	0.23	5.1	5.2	5.3	5.6	5.8	5.9	6.1
Nose breadth	131	1.57	0.15	1.3	1.4	1.4	1.6	1.8	1.9	2.0
Ear breadth	129	1.47	0.13	1.2	1.3	1.4	1.5	1.6	1.7	1.8
Hand breadth	129	3.32	0.15	3.0	3.1	3.1	3.3	3.5	3.6	3.7
Foot breadth	119	3.93	0.19	3.5	3.6	3.7	3.9	4.2	4.3	4.3
Chest depth	133	9.58	0.78	7.9	8.2	8.5	9.6	10.6	10.8	11.2
Abdominal depth	126	10.83	1.32	8.4	8.6	9.1	10.8	12.4	13.2	14.0
Chest circumference, resl	133	37.87	2.98	32.0	33.3	33.7	37.9	41.3	42.0	46.0
Chest circumference, insp.	130	38.42	2.92	32.6	33.5	34.6	38.4	42.1	42.9	46.9
Chest circumference, exp.	130	37.28	3.00	31.5	32.0	33.3	37.4	40.9	42.1	44.9
Waist circumference	108	35.46	3.68	28.5	30.2	30.7	35.2	40.2	42.1	44.1
Upper arm circumference	133	11.28	1.11	8.9	9.5	9.8	11.4	12.8	13.0	14.0
Calf circumference, right	110	13.60	1.07	11.6	12.0	12.3	13.4	14.8	15.2	16.2
Calf circumference, left	109	13.48	1.01	11.7	11.9	12.1	13.4	14.8	15.4	15.8
Head circumference	133	22.34	0.72	21.0	21.3	21.5	22.4	23.2	23.3	23.8

Figure 2.10: (Zelnik)

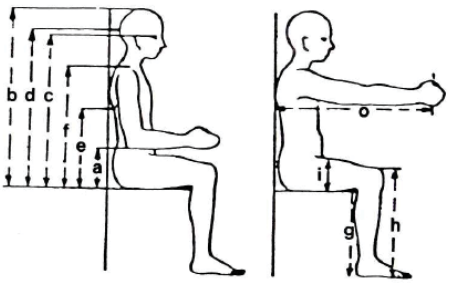


Figure 2.11: (Zelnik)

could be measured (FIGURE data collection tools). It was through the research of diagramming in fields outside of design that led this work to integrate more than just large standardized data sets. The tool developed through this research combines the methods of the anthropometric data representation with the flexibility of mathematical functions to create a diverse and expandable tool useful for simulation as well as pure visualization of human factors.

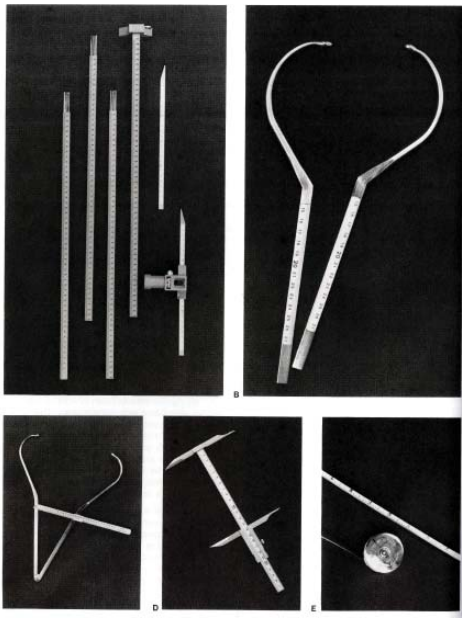


Figure 2.12: (Zelnik)

2|3 **Collaboration**

col·lab·o·rate
verb

To work jointly with others or together
especially in an intellectual endeavor

Creative and innovative solutions are constantly created through collaborative work. The push for collaboration has been seen through the increase of multidisciplinary classes and grants. Much of the work in this book is due to collaborative efforts. The world of knowledge is far to great for one person to know everything. It would also be foolish to assume that all of the useful knowledge for a given subject could be mastered by one person. In this world of human factors, it is clear that a designer must understand the human body in order to design for it. However, it is difficult to know to what end someone must understand the human body. After all, a designer must also know about materials, colors, construction, etc.. This is where collaboration is key, people providing each other with valuable information.

While in many cases the knowledge a designer has of human factors may be sufficient, it is those unique times in which a designer needs additional information on a human in order to design. A common occurrence of this need

is within Universal Design. With a concentration of the ability for a design to be widely accessible, one must know what makes a person disabled. Between diseases, injury, age, and gender the abilities of humans greatly vary (Tilley). Not only does the physical ability of people vary, but so do they instruments and environments people use.

Besides the differences among people, there are times in which the design location is closely linked to human factors. This situation is common when designing for small spaces (Figure 2.13). One example, military vehicle design, is a major sector requiring experts in many different fields. Using a submarine as an example, It is clear that the design must maintain a balance between military efficacy and human comfort. In a military situation, adaptation should not be the fallback excuse for the lack of human usability. The question now is; what hinders collaboration?

The problem that most often hinders collaboration is a lack of a mutual language. This problem runs deeper than just the jargon used. Through the work in this book one obvious issue came up: the tools people use are different. While in design one may use Maya (Autodesk) or Rhino 3D (Rhino), in kinisiology one may use OpenSim (Delp) or Visual 3D (C-Motion). The use of different tools makes the transfer of knowledge extremely difficult. As one side does not know how the others program works, it becomes extremely difficult to bridge. By creating a platform that both disciplines understand, the transfer of knowledge becomes easily achievable.

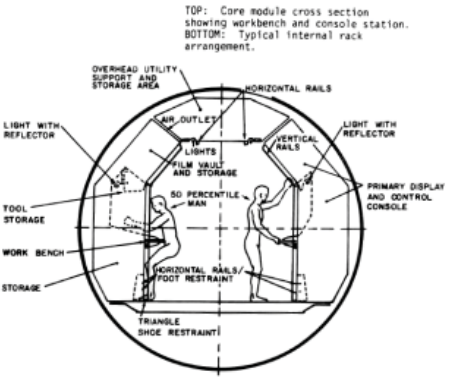


Figure 2.13: (NASA)

Image Credit: (NCARB)

2|4 **Education**

ed·u·ca·tion
noun

The field of study that deals mainly with methods of teaching and learning in schools

Without being exposed to the many situations in which design and human factors are greatly intertwined it is difficult to understand the issues. During the time of this research a class at the University of Michigan on Universal Design was used to understand the challenges people have when designing for something they do not directly experience (Vance). Students learned about different types of disability through in-class exercises initiated by the professor. These exercises were intended for students to experience what the effect of various disabilities was on a person and their environment. In other words, students experienced why someone was disabled.

Why someone is disabled is not very intuitive. What this means is that someone is only disabled as long as they are not able to do something. In relation to design, if a text is too far to read it, it is the text that has dis-abled the reader. In this case, the solution is obvious, move the text closer. Unfortunately most situations are not this easy to understand. It is for that reason universal design classes have physical activities such as putting on goggles to simulate cataracts

and old age (Vance). These activities greatly benefit students, however, it requires a great deal of time to prepare and at worst creates a problem for students who may want to learn but do not feel comfortable participating.

Through simulations of human factors and useful representations, much of the information gained from hands on activities can be learned on the computer. This is not to say that the use of a computer is more beneficial than hands on research, however it does provide an alternative that is not accessible through books. Through the documentation of students engaged in activities and the documents submitted by the students, some important trends are seen. Many of the visuals created by the students either showed a human factor range or showed the passage of time through multiple and overlay images (Figure 2.14). These two common occurrences translate extremely well to the computer. The ability to interact and manipulate a virtual person may not be better than hands on learning but it does provide a useful alternative.

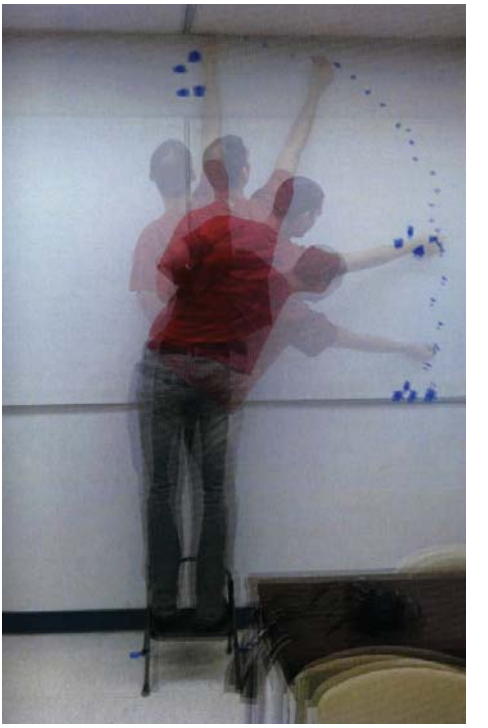


Figure 2.14: (Vance)

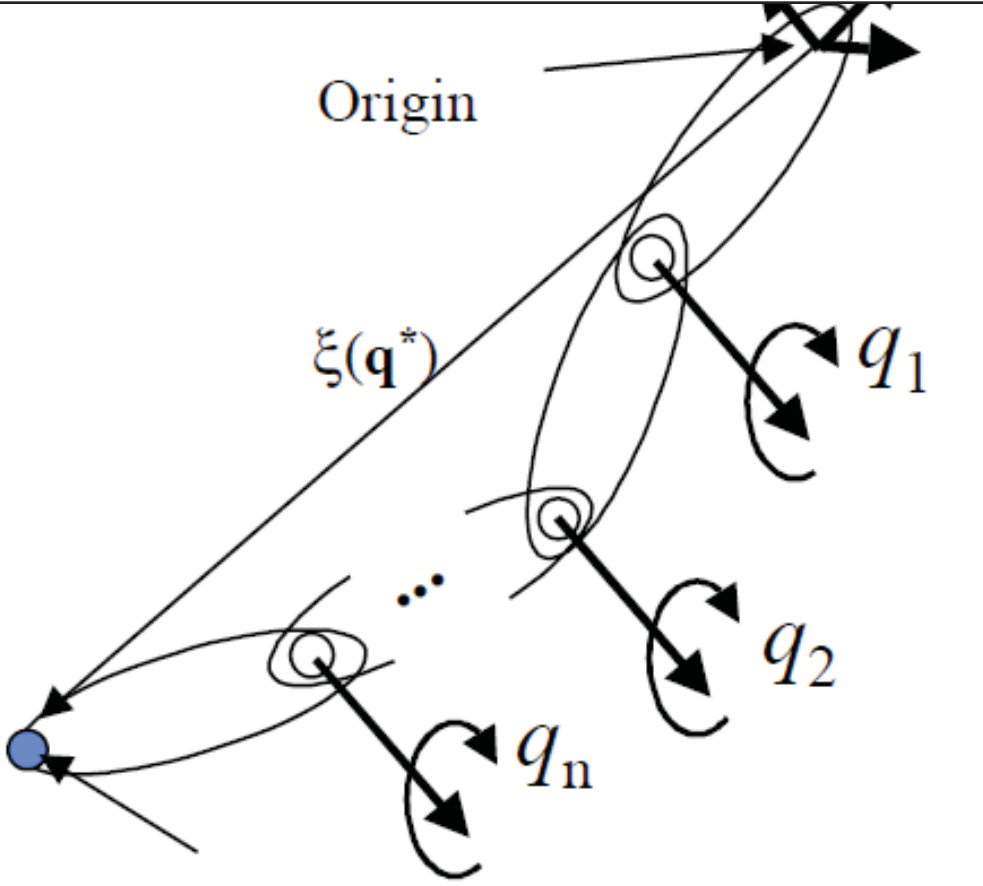


Image Credit: (Abdel-Malek)

3 Technical Concepts

Computer simulation models of human movement can be broken into two subjects, Biomechanics and Computation. Between these two there are numerous considerations when making a simulation. First, Biomechanics can be understood as a way to break down organic movement into structured rules. Computation is then used to translate these structured rules into a function that can take new inputs and generate a new movement. Additionally, human factors other than movement can be integrated into a system for simulation. These systems can use both generic and specific functions to create the simulation. When these functions become complex, the amount of time required to create a simulation can increase. Through different methods of computation the time required for a simulation can be changed. The goal in this research was finding a balance between computational complexity, efficacy of the simulation, and user friendliness. In order to constrain the variables at large, this research has concentrated on human factors within a wheelchair. While this narrows many variables it also introduces some very unique ones.

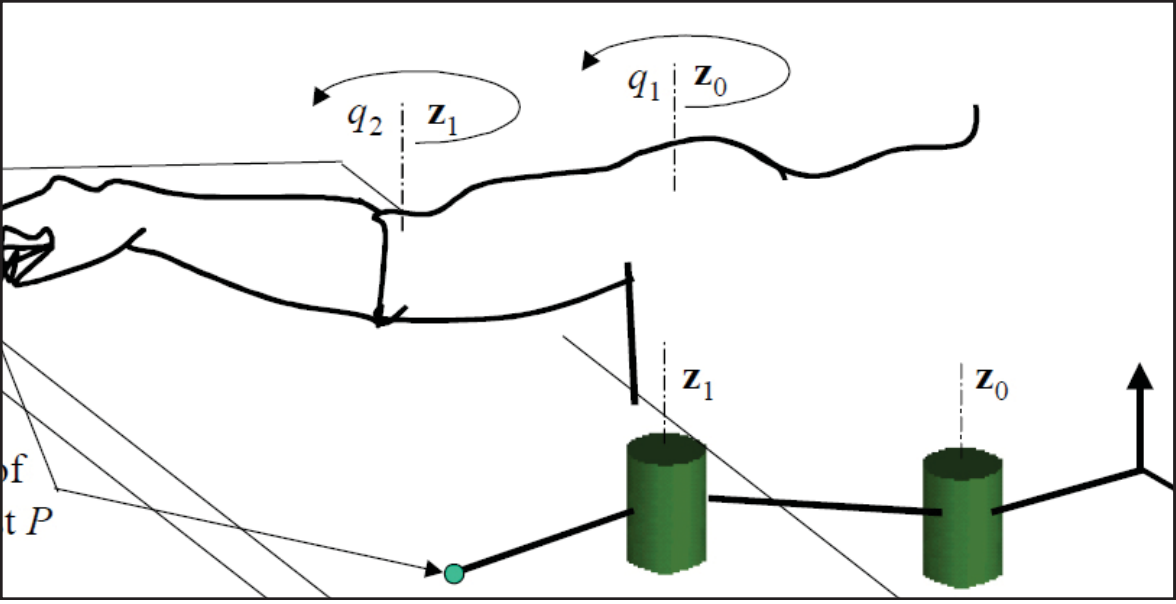


Image Credit: (Abdel-Malek)

2|1 **Biomechanics**

bio·me·chan·ics
noun

The mechanics of biological and especially muscular activity

Biomechanics of human movement can be described by kinematics. The two types of kinematic systems used here are Inverse and Forward. The difference between the two is with the input and output. To relate this to a human, a forward kinematic model is when the should and elbow joint angles are known, and the location of the hand is wanted. Conversely, if the location of the hand is desired, an inverse kinematics model is used to find the angles of the shoulder and elbow joints. This system of shoulder, elbow, and hand joints is called a chain. A chain is made up of links and joints with the end of the last link becoming an end effector (Jazar). In the case of a human arm, the first joint is the shoulder with a link to the elbow and from there a link to the hand, which is the end effector. While an inverse kinematic model can be used to find joint angles, it does not necessarily find good ones. Essentially an inverse kinematic model can find a range of angles that will result in a specific location. The difficulty is in resolving the range of solutions to just one. There are a range of techniques that can be

used. One common technique that is often a result of the given condition includes constraining the angles a joint can rotate, in turn constraining the solutions available. These techniques include constraining the angles a joint can rotate, in turn constraining the solutions available. This method was used at a larger scale, constraining all movement to a wheelchair, and similarly, constraining the solutions available.

Constraining the biomechanical problem to a person use a wheelchair creates an interesting dynamic. This instantly removes the ability to quickly move from one point to another. It also introduces the issue of why someone is in a wheelchair. When looking at spatial design books or the ADA standards it may seem like the only difference is that a person is in a sitting position (Figure 3.1). Unfortunately the problem is much different. There are many diseases that can effect a persons movement such as Multiple sclerosis. The reason Multiple Sclerosis effects the bodies movement so much is through inflammation of the brain or spinal cord (Owens). As the spinal cord acts as a carrier of the bodies nerves, any damage or disruption can change the bodies ability to function. Similarly, spinal cord injuries change a persons ability to move. Hence, broken bones or old age are only a few reasons someone may need to use a wheelchair. To narrow down the problem we look specifically at spinal cord injuries and a persons ability to reach.

The spinal cord is made up of vertebrae, these vertebrae stack up and are split into four groups. The location in which a nerve exits the spinal cord is relational to the body parts it effects (Figure 3.2). What this ends up meaning

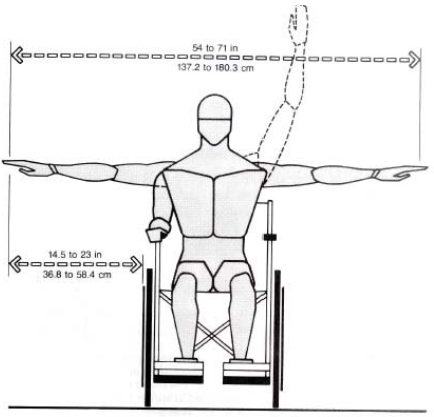


Figure 3–4. Anthropometrics of chairbound people. The front view, showing user and chair, also indicates some of the more critical anthropometric measurements. The source of the bilateral horizontal reach dimensions with both arms extended to each side, shoulder high, was the American National Standards Institute (ANSI Pub. A117-1961, Revalidated 1971). It should be noted that no data were available with regard to sex or precise percentile grouping.

Figure 3.1: (Zelnik)

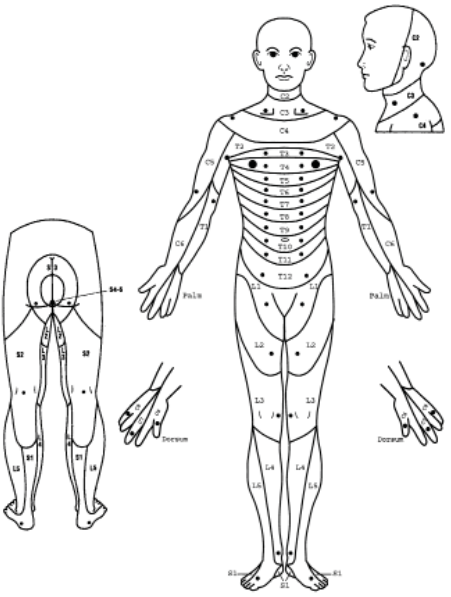


Figure 3.2: (ASIA)

is that a spinal cord injury may not just result in the loss of leg usage, but may change the way other body parts work. For example, a spinal cord injury that occurs between the vertebrae T11 to L1 results in the paralysis and loss of sensation in the hips and legs. This low level injury results in biomechanical movement similar to that shown in the ADA standards. However, an injury in the Thoracic mid range between T5 to T8 results in paralysis from the lower trunk down as well as a loss of sensation below the rib cage. Since the human body is all connected, what may seem like an injury that only effects below the ribcage ends up translating above as well. When looking at the reach ability physics begins to take a role. When a diagram shows someone in a wheelchair leaning forward, it is assuming the person is able to get back up (Figure 3.3). In order to counter act the body weight leaning forward, muscles in the lower part of the body need to contract. The problem then is if a person is either paralyzed or has limited sensation below the ribcage their torso muscles are significantly reduced (Castro). With reduced muscles, the person does not have the ability to compensate for their body weight and is then not able to lean. Some people have such limeted strength and/or balance that they need to be mechanically stabalized (FIgure 3.4). These examples show a population in which the reach ability is far different from the average man. Even further down this road is the high level spinal cord injuries that create reduced sensation of the hands and arms, directly affecting the upper limb movements. This demonstrates the need for a more complete model

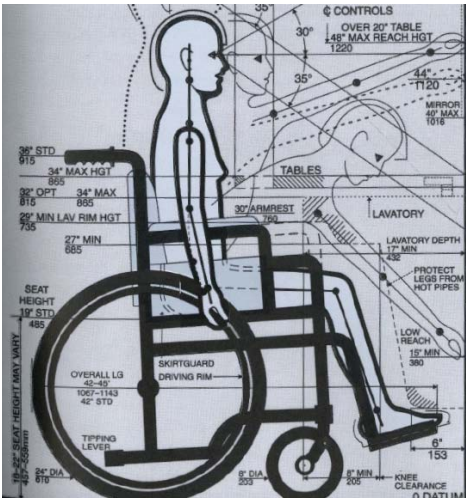


Figure 3.3: (Tilley)


Figure 3.4: (RehabMart)

relating to human ability.

Studying human movement is a different problem than simulating movement. Modern technology has allowed researchers to study the bodies movement in three dimensions with motion capture. Optical motion capture is a common way to track human movement (Moeslund). Using small reflective spheres known as markers, attached to a persons body, infrared cameras record and triangulate the markers position (Figure 3.5). This recording becomes 3D point data on the computer (Figure 3.6). A computer skeleton is then created from the point data. The importance of this creation is the introduction of angles into the data. Similar to the inverse kinematics problem, angular data gives different information than point data. Together, point and angular data provide a complete system in which a researcher can extrapolate the data.

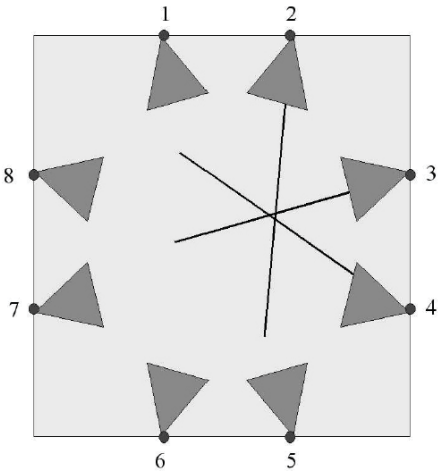


Figure 3.5: (Filho)

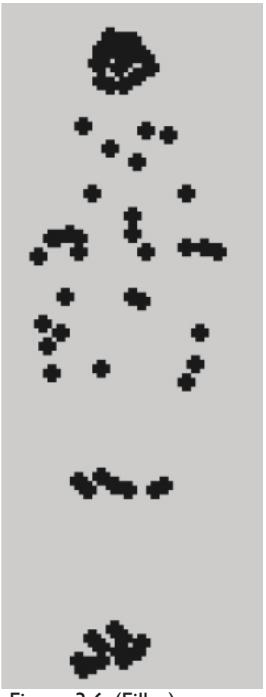


Figure 3.6: (Filho)

A joint is truly a point in which a rotation occurs. Since rotations are measured in angles, a study on human joints would require angles. Additionally, angles are easily transferable between research subjects as, unless there are extenuating circumstances, joint types between people are all the same. Location data however, varies by person because of differences in marker placement, fat content, and height. Therefore, in order to study the speed in which someone stands up, the angle of the hip over time would be wanted, whereas just the location of the hip when standing will greatly vary. Transferring this knowledge to reach ability is a critical step in understanding simulation of human movement. In order to accurately describe where a

person can move a limb, it is not the location of the hand that is important, it is the angle of the joints. By knowing that everyone's shoulder can rotate 10 degrees behind themselves, a computer skeleton at any desired height can be created with the arm in the furthest position backwards (Figure 3.7). From there the information of where the hand is can be gathered. This is a basic example of being able to simulate human factors.

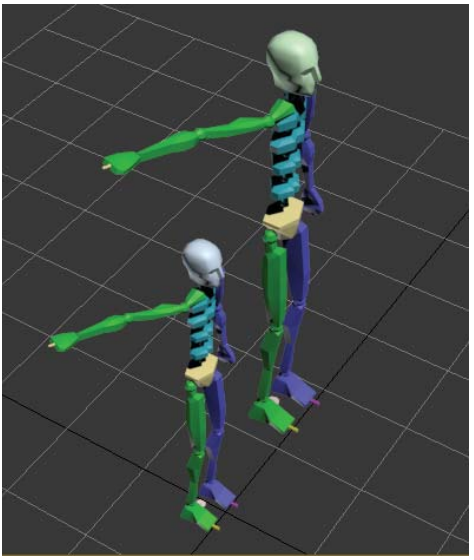


Figure 3.7

```
x = BoxLoc[0]
y = BoxLoc[1]
z = BoxLoc[2]
if ( ( sqrt( (pow((px-qx),2)) + (pow((py-qy),2)) + (pow((pz-qz),2)) )  ) < AccuracyThresh*size ):
    Dist=sqrt( pow((px-qx),2) + pow((py-qy),2) + pow((pz-qz),2))
    #cmds.select(BoxNew[0])
    #Select the targeted box and change its color
    cmds.select(VoxelArray[ReachVars.Box_Index[0]][ReachVars.Box_Index[1]][ReachVars.Box_Index[2]][0])
    if (ReachVars.type==1):
        cmds.polyColorPerVertex( rel=True, g =0.04, a=0.1, r=-0.05)
    if (ReachVars.type==2):
        cmds.polyColorPerVertex( rgb=(0.0, 1.0, 0.0),a=1.0,colorDisplayOption=True )
    #Track how many times each box is reachable
    tmp=1
    try:
        VoxelArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz][2]=(VoxelArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Ind
    except IndexError:
        VoxelArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz].append(tmp)
    #******************** Full Reach *****************
    if ( (VoxelArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz][0]) not in ReachVars.BoxQ):
        ReachVars.BoxQ.append(VoxelArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz][0])
        ReachVars.QIndex.append([ReachVars.Box_Index[0]+ix,ReachVars.Box_Index[1]+jy,ReachVars.Box_Index[2]+kz])
ds.select(IK_Handle.Current)
(ReachVars.MayaHIK==True):
    mel.eval("hikManipStart 1 1")
    cmds.move( ReachVars.Box_targetPOS[0],ReachVars.Box_targetPOS[1],ReachVars.Box_targetPOS[2], ws=True)
    mel.eval("hikManipStop")
se:
    cmds.move( ReachVars.Box_targetPOS[0],ReachVars.Box_targetPOS[1],ReachVars.Box_targetPOS[2], ws=True)
llReach(VoxelArray,size, moveDirect,AccuracyThresh):
the movejoint function, then compare the boxes.  Any box that passes test gets in line to become the main box
n time a box is found true during the compareLoc function, it adds to the value
ile (ReachVars.Q < len(ReachVars.BoxQ)):
    Box_target = ReachVars.BoxQ[ReachVars.Q]
    ReachVars.Box_Index = ReachVars.QIndex[ReachVars.Q]
    JointAbility(VoxelArray,Box_target,size, moveDirect,AccuracyThresh)
```

2|2 **Computation**

com·pu·ta·tion
noun

The action of mathematical calculation

Simulations of human factors is by no means trivial. On top of the computation needed for the simulation, 3D visualizations require a way to be seen and utilized. Two sides to implementation through computation are speed and user friendliness. On one side are the low level programming languages such as c++. These programming languages are closer to machine code than others and are required to be compiled (Ousterhout). This means that the code written must be reformatted by the computer into a computer readable state before executing the code. The benefit of this is the speed that the computer can run the functions. On the downside, integrating newly written code with pre-compiled code is difficult and the syntax is far more complex. On the other side is programming languages like Python, which have an easy to use syntax, easily integrated with other code, yet is painfully slower (Ousterhout). Generally, modern technology is able to run human simulations relatively quick. For example, character animation

programs using optimized algorithms, full inverse kinematic systems can be computed in real-time (Tolani).

Programs like Maya, which are more than just character animation programs as well as other 3D modeling programs have begun to offer access to an Application Programming Interface (API). An API allows people to program functions while using pre-exisitng ones from the program itself (Figure 3.8). Currently many 3D modeling programs offer access to the API through Python. This is beneficial for human simulation in two ways, most importantly this allows for quick and easy access to visualization functionality. Without an API it would be nearly impossible to truly integrate simulation techniques in a designers work-flow without creating an entirely new program. Another benefit is the ease in which python is able to reference additional python files. In Python, these separate files are called modules. The separation of functions into different modules assists in the legibility of the code but more importantly allows for an extremely portable function. Since different API's require specific ways of accessing the internal code, a function using one API will not work in a different one. This difference in API's is where the use of separate modules for code becomes useful. By separating the simulation function from the visualization function, transferring from one program to another is much simpler. In addition to being able to easily transfer the simulation, creating modules that exclusively contain universal code provides a straight forward display of the simulation technique.

Although currently many

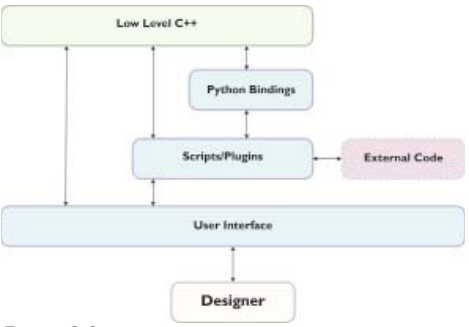
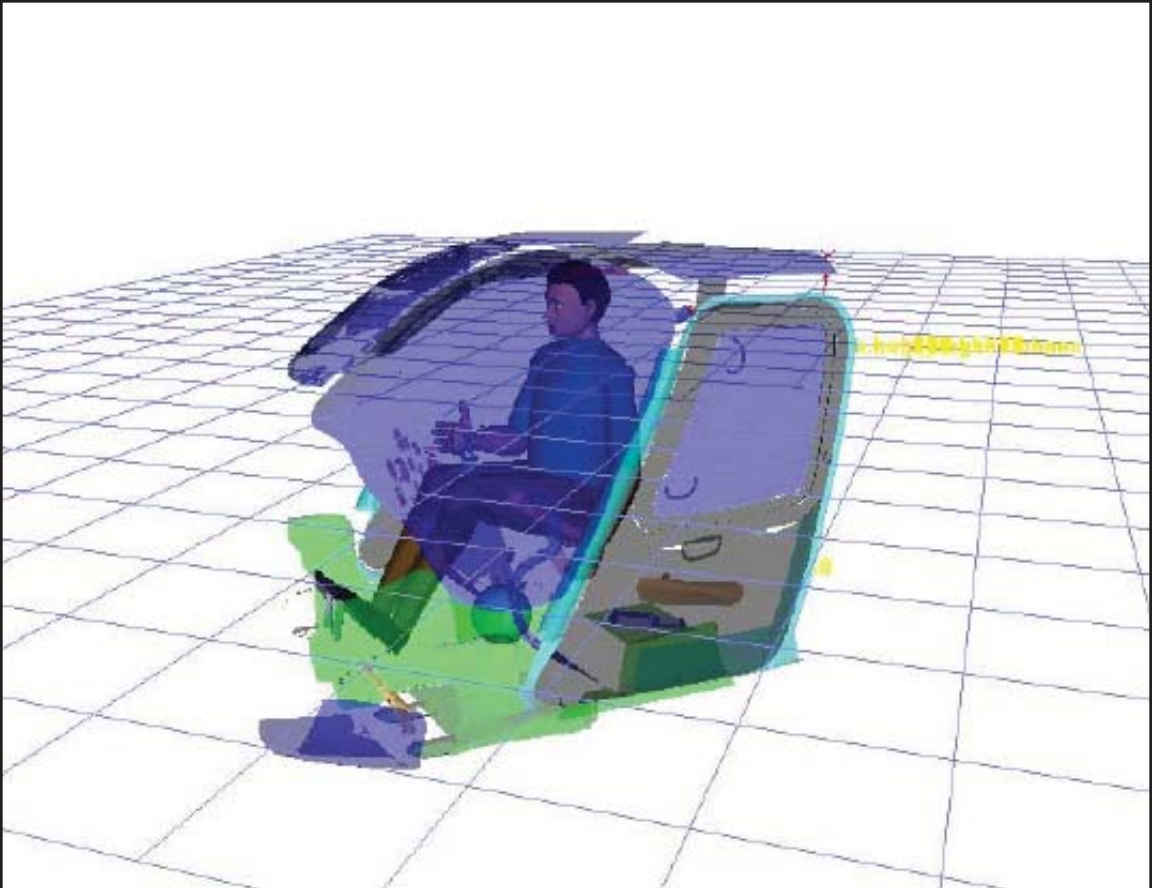


Figure 3.8

programs designers use have access to an API, it has not always been this way, especially with python. The reason the integration of an API is so valuable to this research extends beyond the technical hurdles of programming a simulation and visualization from scratch. The integration of Python programming capabilities within the tools designers use is also an integration of additional tools and techniques. The past twenty years has been fruitful with human simulation programs. These include methods for virtual navigation, kinematic simulation, and BIM analysis (Boeykens). While these examples provide legitimate methods for simulation and/or visualization, they are all developed in an independent program. There are a few possible reasons for this. Either the research is specifically targeted at solving a problem and is less concerned with the utilization of the program, or the creation of an open source program is used as leverage against the high cost of the available programs (Eriksson). The former reason is opposite the goal of this work, and the latter is in part agreeable. The high cost of specialized human simulation programs like JACK do not make for a very accessible tool. However, this work focuses on current designer work-flows, which relatively low cost tools such as Maya, Rhino, and 3D Studio Max are most common, and all have Python integrated.

In addition to the computational issues with programming languages and API's is a broader issue of algorithm efficiency. Although there are some instances in python that can never be as fast as a low level programming language, there are many ways to make python even slower. Especially when using python with an API, the balance between using premade scripts in the API and custom writing them is important. On a general note, algorithms themselves should be implemented in a way suited to the programming language.



4 Precedent

Image Credit: (Blanchonette)

In addition to the conceptual research of human centered design are four precedents that demonstrate a combination of conceptual and technical work as it relates to human factors. These works come from various fields and have been used as a main source of inspiration and comparison to a greater extent than the references. Each of these works are linked to a period of time in which many of the technical challenges vary from those today.

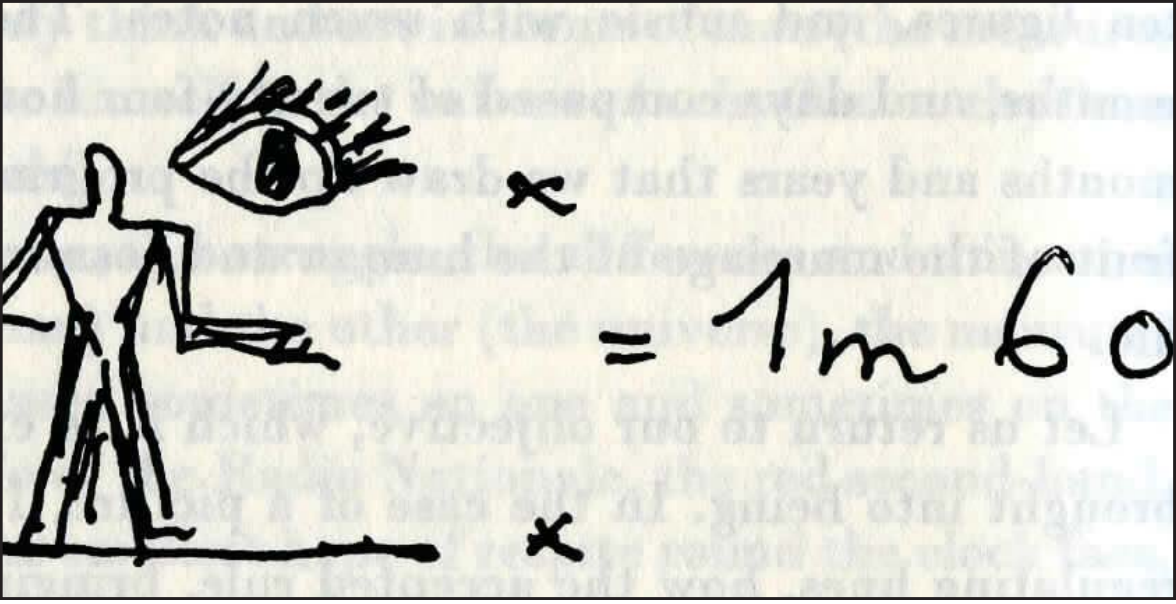


Image Credit: (Le Corbusier)

4|1 The Modular

"These chapters contain no scientific argument. It is simpler that way; I am no scientist."
-Le Corbusier

As discussed in the preface, The Modulor was a system developed by Le Corbsuier describing the human proportions. These proportions where linked to the golden ration. The influence of The Modulor is from its combination of visualization and integration with design. By the drawings Le Corbusier created, buildings were given a direct link to the human proportion. Similar to the Vitruvian era, The Modulor took a close interpretation of the human proportion and attributed it to a system in which the human can be represented. From this works perspective, the main result of The Modulor was a surge in buildings and design that took 2D human proportions and extruded them (Figure 4.1). Le Corbusier took compliment to this style as he wrote on the many buildings he had influenced in his second book. It is the intention of this work to bring an understanding that the human body is in no way two dimensional, and through the understanding of biomechanics, humans do not simply scale. In fact, through studying the effect of a spinal cord injury on a person, it is known that human factors are a four

dimensional problem involving not just location but strength of muscles, introducing velocity.

Figure 4.1: (Seidler)

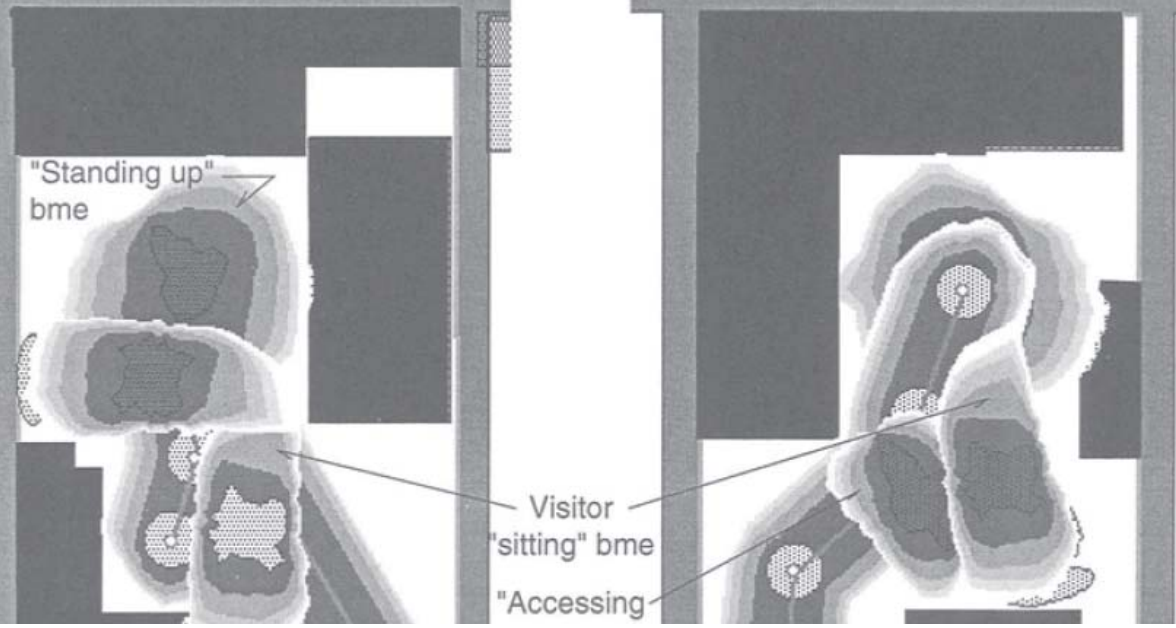


Image Credit: (Lantrip)

4|2 **ISOKIN**

"A person's physical disability is given meaning and value only when it is found to interfere with some desired activity."
-David Lantrip

In 1993, a few years after the ADA was established, David Lantrip published a program called Mac IsoKin. With support from NASA and Steelcase, he worked on a program that displayed the body-motion envelope (Figure 4.2). By taking video of users performing a task such as the removal of a jacket, Lantrip created outlines of human extents. This system built upon the currently available sectional description of ergonomics by incorporating the available technology, such as video recording and computer GUI's. MacIsoKin uses a database of body-motion envelopes and displays them within a space described by sectional computer drawings. The designer is able to use this as a tool to understand the interior space in relation to the tasks it is designed for. From the floor plan created and the body motion envelopes selected, quantitative analysis can be run on the space. Similar to the outlined environmental simulations in this books introduction,

MacIsoKin simulates the perceived discomfort in an environment. In this case, the input is a predefined envelope and a floor plan (Figure 4.3). The simulation is a function of the space the envelope takes up and the surrounding objects existing within that space. IsoKin was highly referred to during the ideation of this research due to its unique approach to quantification of design relative to human movement.

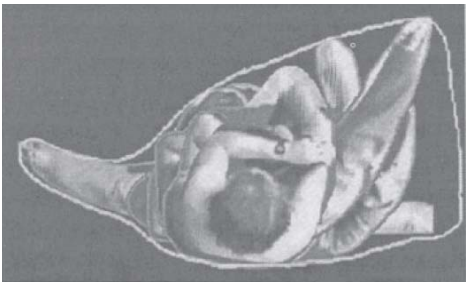


Figure 4.2: (Lantrip)

An conversation with Dr. Lantrip provided insight to the challenges IsoKin faced, technically as well as with integration into a designers work-flow. One theory in which Dr. Lantrip agreed was the limited use of computers during the time of the program being written. The focus of this work in relation to the integration of a tool within a designers work-flow is attributed to the vast difference in current computing technology and past research. New technology provides a fundamental benefit, and a different way in which IsoKin may be approached if re-done today. Optical motion capture systems have become widely accessible and provide highly accurate and comprehensive information. As discussed in the Technical Concepts chapter, motion capture data can be parsed into angles and positions. In this case, a modernized version of IsoKin would be using the position information to create a body-envelope directly of a specific person. Using this method would require multiple recording sessions to get different body-envelopes of the same motion. Although not all movements would work this way, such as taking off a coat, using angular data from the motion capture instead of the position data would allow for a fast

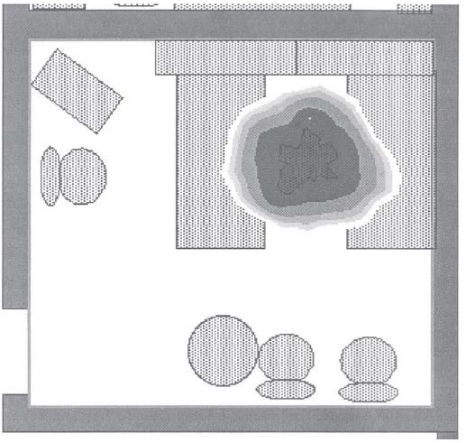


Figure 4.3: (Lantrip)

creation of people of multiple sizes.

IsoKin is being used here as the main example, however many other examples exist in which the program was developed independently of any tools used by designers. Some of these examples are referred to in the Technical Concepts chapter of this book. It is the example of IsoKin and others that demonstrate a continuous research field of human factor simulation to benefit design. However, they all share the same problem, a tool is only beneficial when it is used.

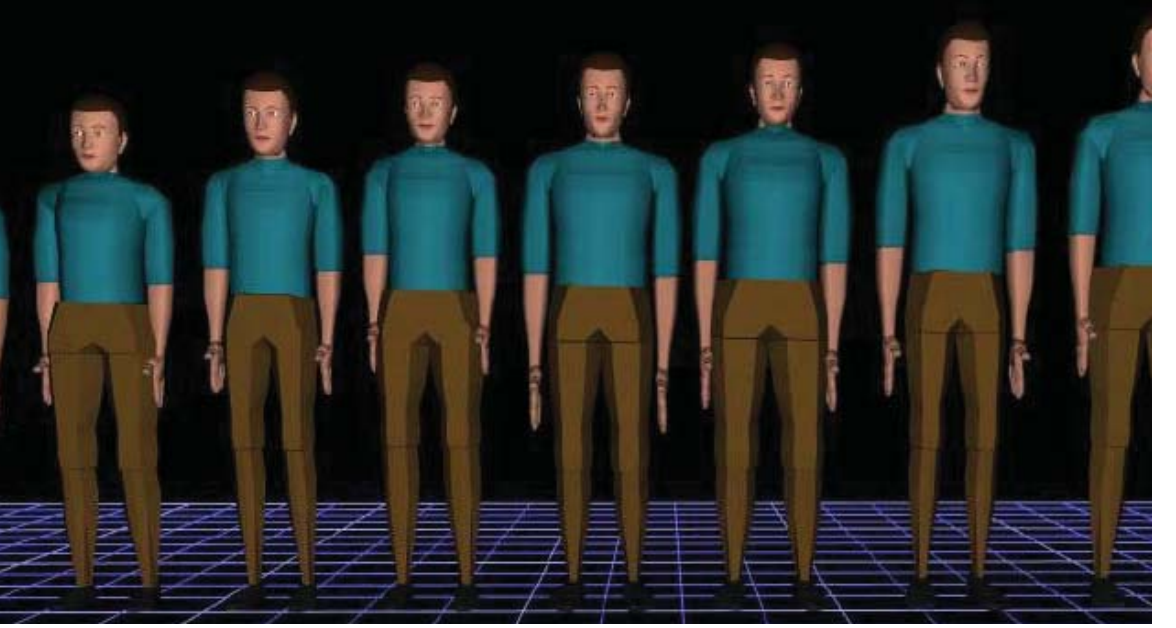


Image Credit: (Blanchonette)

4|3 Technomatix Jack

One of these expensive human model tools describe earlier is Tecnomatix JACK. Originally developed at the University of Pennsylvania, JACK is now a commercial product of Siemens targeted to the engineering field (Blanchonette). JACK provides a human model that can be positioned in relation to a design (Blanchonette). The model is used to view the reach envelope and a few other factors such as the field of view. JACK is one of the most popular human model programs and comes at a significant premium. For this reason, and the difficulty in creative modeling, JACK is rarely used by designers. JACK does however provide a range of analysis tools. These tools include injury risk, timing, user comfort, reachability, lines-of-sight, energy expenditure, and fatigue limits. As discussed in the introduction to this book, these various simulations are difficult to trust as the source of the function is unknown.

JACK has not always been closed source. When JACK was being developed, many of the algorithms created were openly published. JACK provided significant advancements in human simulations. The original work is highly referenced to this day. What JACK did lack, and still does, is a robust model that can be adjusted to accurately simulate special cases. In previous research, the ability to accurately describe the capabilities of persons with spinal cord injuries was tested in JACK. The results showed that an extraordinary amount of work was needed to simulate the special case reach ability. The number of steps needed to create the simulation questions the validity. It is known that JACK was never developed for simulating spinal cord injuries. By not knowing the underlying code in JACK now, along with the knowledge that it was never designed for special cases suggests that even if the mannequin can simulate the movements, any of the other analysis may be invalid.

The biggest problem with JACK is not in the algorithm for simulation but in the method for visualization. As with many reach ability visualizations JACK

uses an ISO-Surface (Figure 4.4). Other research has added layers to the ISO-Surface to represent multiple extents, such as the furthest reach or comfort reach (Figure 4.5). The problem with this ISO-Surface is that it tends to imply an inclusive volume. However, human reach ability is not an inclusive volume. The combination of joint types, muscle, fat, tendons, and ligaments create a complex map of reach ability impossible to convey in a single ISO-Surface.

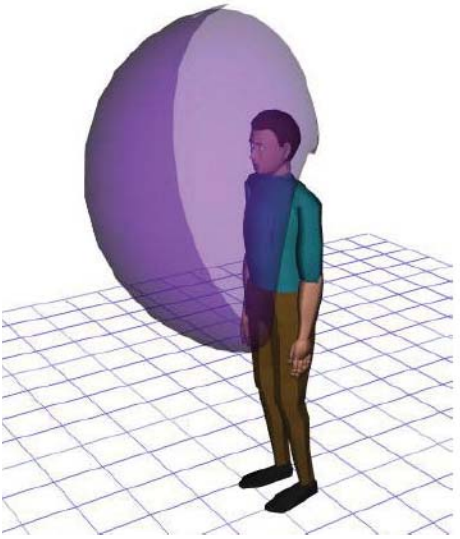


Figure 4.4: (Blanchonette)

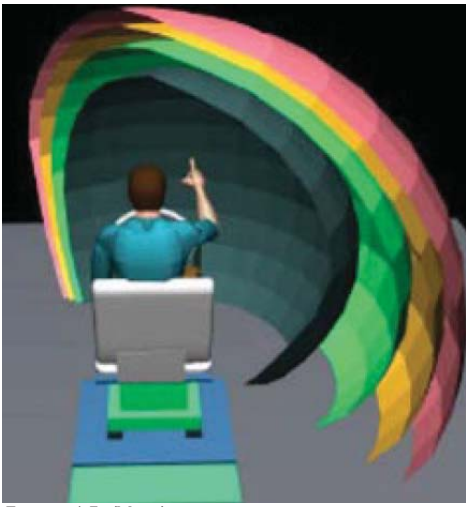


Figure 4.5: (Yang)

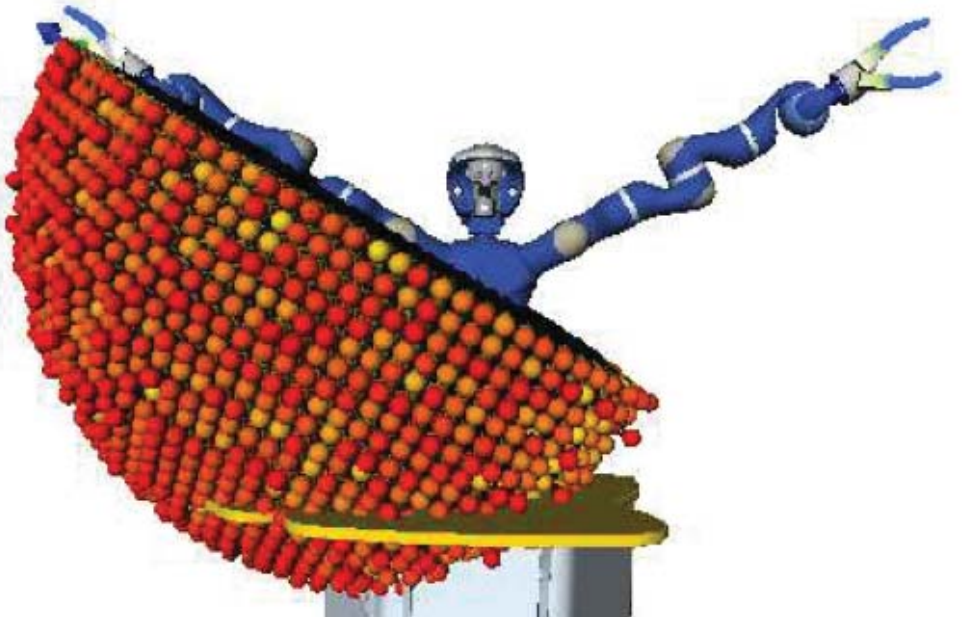


Image Credit: (Zacharias)

4|4 **JUSTIN Humanoid**

"In general, every robot arm is designed differently, and therefore has different kinematic capabilities."
-Franziska Zacharias

The humanoid robot JUSTIN demonstrates creative methods for visualizing reach ability and path planning. In the case of humanoid robotics, biomechanics plays a pivotal part in the link between human visualization and robotic inverse kinematics. There is a very interesting dynamic between the way humans are modeled and the way humanoids are modeled. For a human, the goal is to convert the joints and bones into a mechanical kinematic system in order to solve for the angles. Due to the soft tissue and flexibility of a human, complex chains can be combined in order to get a higher resolution solution (Sancho-Bru). In the case of a humanoid, the goal is to translate the human dynamics into a joint system that will physically be moved like a human, yet may not be the same anatomically. Franziska Zacharias has demonstrated creative ways of making robots move like humans. Using the rapid upper limb assessment (RULA) system, Zacharias tests a position given by the solution of a kinematic solver (Zacharias). The test shows

if the position is a good ergonomic position. This can be abstracted to a person would be comfortable in this position, and as such the humanoid can adjust to the best score (Figure 4.6). The technique used here can translate to the many parameters an IK solver can have. These parameters, adjustable within Maya, include joint limits, weights, and best position. These types of inputs give a solver more information and reduces the number of solutions (Jazar).

The strongest reference to JUSTIN is in the visualization techniques. Zacharias demonstrates a spherical reach ability map that can include approach directions (Figure 4.7). This type of reach ability map provides a much higher resolution than a reach envelope. Resolution is referring to the information on what places are reachable. In a forward kinematic system, a resolution would be the increments of degrees in which to test the end effectors location . In an inverse kinematic system the resolution is the distance between target points on a grid (Zacharias). Similar to the technique of this work, Zacharias uses color gradation to visualize the reachable areas. However, the opaque colors present a problem for fast interpolation. As such, the reach map must be displayed in sections (Figure 4.8). These sections make understanding the zones easier on paper, however, there must be a different method for 3D programs in order to make an easy to use and understand visualization.

Figure 4.6: (Zacharias)

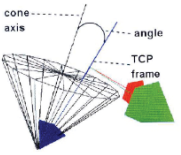


Fig. 4.14 For a TCP frame it is tested whether it is represented by the shape primitive.

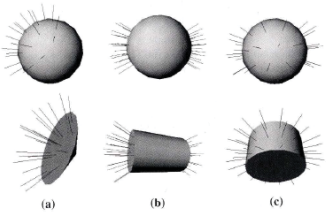


Fig. 4.15 Cones (a) and two cylinder types to capture structures. (b) Cylinder type C1, (c) Cylinder type C2

Figure 4.7: (Zacharias)

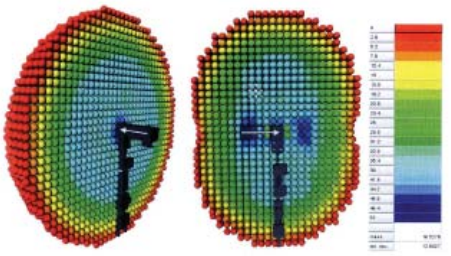
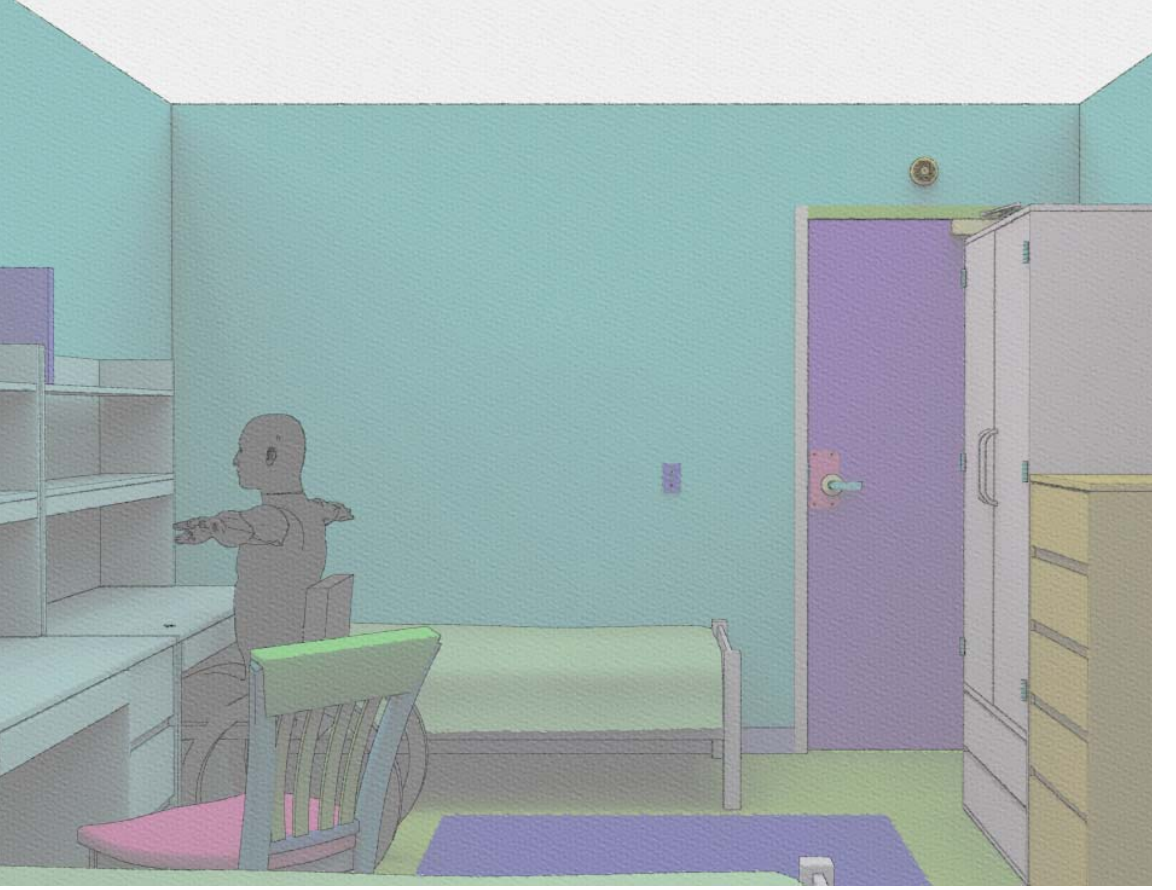


Fig. 5.2 Capability map for the Schunk 6 DOF light weight robot is shown. The first rotation axis of the robot arm is shown by a white arrow. The color encodes the reachability index with z-orientations D_z. The mean reachability index is low.

Figure 4.8: (Zacharias)



5 Process

This section discusses some of the overall methodology as well as some early work that has led to the final research. The chapter is broken into three sections: recording and analyzing movement, designing a graphical user interface, and using python for scripting. The early work with motion capture was barely used in the final research. However, the ideas as to how bio-mechanics could be incorporated into design tools came from the initial work with motion capture. In order to present the research in a way that showed its efficacy being implemented, a Graphical User Interface (GUI) was developed. QT Designer was used to translate the interface into python. The code was all written in python. Each segment of research was developed in a separate python module. These modules were linked together and with the user interface through one main module.

5|1 Motion Capture

Motion capture technology was used very early on in the process of this research. The initial use of the motion capture sessions was to collect data that would be parsed and analyzed in OpenSim. The movements were simple interactions with a chair and an initial test on a reach envelope. The virtual skeleton created from the data was used to drive a character rig of a laser scanned body (Figure 5.1). From this movement data, different methods of visualizing the movement were explored. The act of recording the session, translating the data, and working between programs is what inspired the research to not just look at visualizing biomechanical data, but create a new work-flow and workspace which assisted in translating movement data.

The motion capture session was done with an eight camera optical system. The subject being captured wears a velcro body suit with spherical reflective markers attached to it. The markers are placed in strategic areas that allow for accurate human reconstructions (Crane). The cameras emit infrared light, and record the reflections. These reflections are triangulated to create 3D locations for each marker. One of the biggest issues with optical systems is that each marker must be visible to more than one camera at a time (Moeslund). When studying the movement of sitting in a chair, the chair becomes an object that occludes the markers, making it impossible to track. With limited time and resources, in order to have a usable set of data that could be analyzed, a custom chair was made that made removing parts such as the arm rest easy (Figure 5.2). This customizable chair enabled a more consistent tracking of the markers as well as offering a variety of chair configurations in which the subject could interact with.

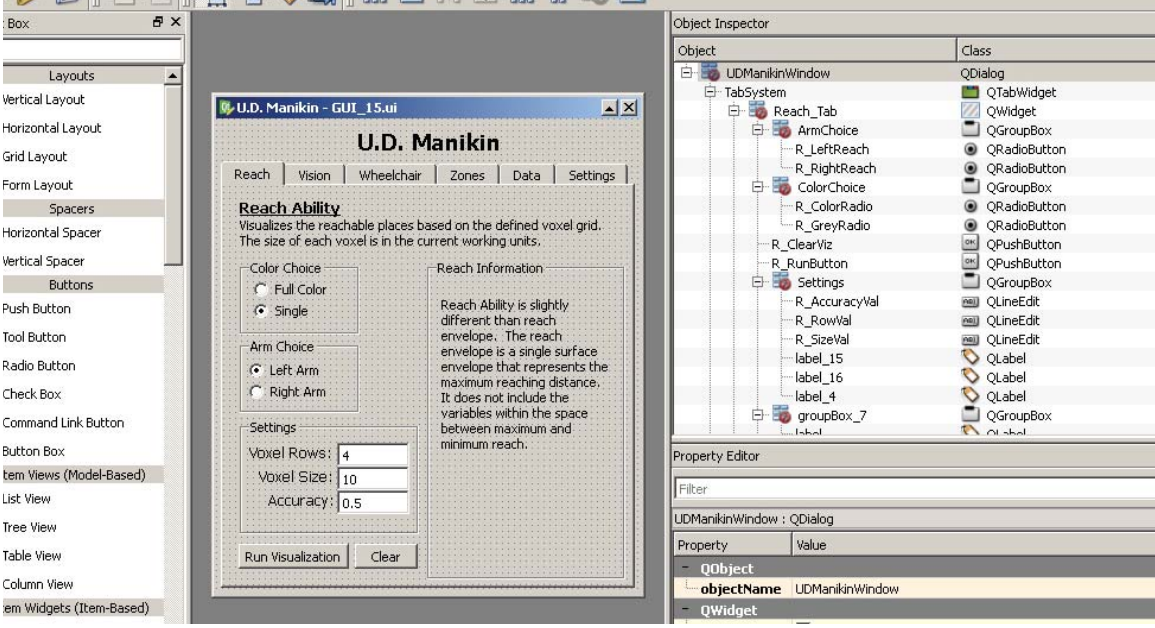
The markers are recorded at a specific frame rate. After the recording, each frame consists of a set of 3D marker locations. At this time, these marker locations are all independent. Blade (Vicon), the motion capture software,

attempts to automatically track each marker and assign it a unique name. At the first frame, each marker is labeled with a unique name corresponding to the location relative to the body. From this set of names, the software attempts to follow each marker throughout all of the frames, and maintain the markers name. Although this can be completely resolved, with an eight camera setup, especially with an action that involves occluded markers, there are labor intensive processes required for the data to be completely processed and usable.

After the data cleanup OpenSim was used to create a skeleton in which kinematic data could be extrapolated. This workflow is common in Kinesiology, however, most designers have never used OpenSim. Comparitively, the motion capture data was also brought into Motion Builder (Autodesk). Motion Builder is easily integrated with Maya. The constructed skeelton from MotionBuilder was imported into Maya and with Python sciprts, the joint angles were written into a CSV file. These files were used to demonstrate the researches ability to represent the movement data in a scientific way within Maya.

Figure 5.1

Figure 5.2



5|2 Graphical User Interface (GUI)

The difference between what someone considers a script, plugin, or program varies. For this work, the terms are defined as such; a script is a series of coded lines that can be run to perform a specific task, a program is a compilation of algorithms (or scripts) that has an interaction between the code and user, a plugin is a program developed to interface with a program or takes advantage of low level concepts used by a program to add functionality with it. The focus of the user interaction with the research was to create a seamless experience with their current work-flow. This meant, for not just the reasons of using an API, the tools developed in the research would be best used in an embedded way. For one, this type of interaction required a representation that moved past text and click input. This type of input is most often seen in Rhino scripts (Figure 5.3), especially with the integration of python. In Rhino, the work-flow of a basic script is to have a back and forth interaction with the user. The script asks something, the user selects some options, selects the object to perform the operation on, and then resumes the script. This linear work-flow is similar to how a text based interaction ideally works. In a python interpreter, the program is able to wait for input from the user. This means there is nothing else happening, and the program will wait until it receives input. For Maya, and many programs like it, the interaction between the user and program is more complicated.

When a program loads, almost all of the functionality loads with it. This allows different parts of the program to interact, as when the user presses a button, an event happens. Unlike a linear text based system, the program does not wait for one specific input, instead it must be explicitly told ahead of time what interactions will happen. At first this makes the transition difficult, for instance, if the program is meant to loft two curves, the user must have already selected the two curves, run the program, and the program can complete the loft operation.

If the user does not select the curves, the program does not know how to just wait for that input and will fail.

This is where graphical interfaces come into play. The graphical interface acts as a communicator between the user and program. By having this layer, interactions like clicking buttons, are able to be linked ahead of time to a portion of the program. Instead of the program automatically executing everything, the program is first told that when a button is pressed, it should execute a specific part of the code. This is referred to as a call-back (Rossum), as the button must call back to a function to be executed. A call-back setup is then always running, allowing the user to select the curves, and then specifically click the button to perform the loft. This is the basis of the GUI setup used in the research. Once the approach was resolved, the physical implementation and design of the interface was done.

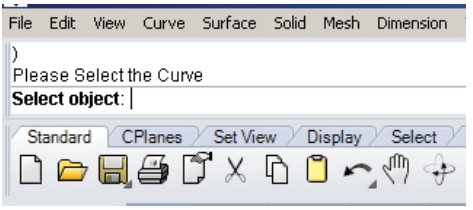


Figure 5.3

Maya has internal GUI options available through the API. The GUI options can be programmed inside the Maya Python interpreter with any code that needs to run. This method was quick for creating buttons or input fields, but designing the GUI became a tedious task. The original GUI creations were long stacks of options (Figure 5.4) that made interaction with the GUI difficult and confusing. To expedite the implementation process a GUI design program, QT Designer, was used. Through many iterations, the final design is a setup of layered windows. The windows are separated by tabs, allowing the screen space taken up by the entire program to remain minimal. Each of the distinct

Figure 5.4

areas of research was given its own window. Although separated, the tabs consist of a simimilar setup. The top of the tabbed windows explains some technical specifitcations of the tool. The left side contains the options for simulating and/or visualizing, with execution buttons underneath. The right side of the window displays information on the subject matter being worked with (Figure 5.5).

Once the GUI was designed and appropriately named, PYQT (Summerfield) was used to translate the QT File into Python. With the GUI in Python, a new module was made that integrated all of the simulations and visualizations with the GUI objects. This was done by using Signals and Slots, which created the callbacks. The GUI was then able to be implemented as a custom button in Maya (Figure 5.6).

Figure 5.5

Figure 5.6

```
2]
 (pow((px-qx),2)) + (pow((py-qy),2)) + (pow((pz-qz),2)) )  ) < AccuracyThresh*size ):
t( pow((px-qx),2) + pow((py-qy),2) + pow((pz-qz),2))
lect(BoxNew[0])
the targeted box and change its color
ect(VoxelArray[ReachVars.Box_Index[0]][ReachVars.Box_Index[1]][ReachVars.Box_Index[2]][0])
hVars.type==1):
.polyColorPerVertex( rel=True, g =0.04, a=0.1, r=-0.05)
hVars.type==2):
.polyColorPerVertex( rgb=(0.0, 1.0, 0.0),a=1.0,colorDisplayOption=True )
ow many times each box is reachable


lArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz][2]=(VoxelArray[ReachVars.Box_Index[0]+ix][ReachVars.B
ndexError:
lArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz].append(tmp)
************** Full Reach *****************
oxelArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz][0]) not in ReachVars.BoxQ):
hVars.BoxQ.append(VoxelArray[ReachVars.Box_Index[0]+ix][ReachVars.Box_Index[1]+jy][ReachVars.Box_Index[2]+kz][0])
hVars.QIndex.append([ReachVars.Box_Index[0]+ix,ReachVars.Box_Index[1]+jy,ReachVars.Box_Index[2]+kz])
IK_Handle.Current)
.MayaHIK==True):
("hikManipStart 1 1")
e( ReachVars.Box_targetPOS[0],ReachVars.Box_targetPOS[1],ReachVars.Box_targetPOS[2], ws=True)
("hikManipStop")

e( ReachVars.Box_targetPOS[0],ReachVars.Box_targetPOS[1],ReachVars.Box_targetPOS[2], ws=True)
oxelArray,size, moveDirect,AccuracyThresh):
oint function, then compare the boxes.  Any box that passes test gets in line to become the main box
x is found true during the compareLoc function, it adds to the value
Vars.Q < len(ReachVars.BoxQ)):
et = ReachVars.BoxQ[ReachVars.Q]
s.Box_Index = ReachVars.QIndex[ReachVars.Q]
lity(VoxelArray,Box_target,size, moveDirect,AccuracyThresh)
```

5|3 Python

A thorough explanation of python is not in the scope of this process section, however, since the use of python within Maya was significant, I will go over a few points. The usage of python can take place in two different ways. First, python can be used in its raw form, as a portable scripting language. Second, python can be used as a syntax, in which the commands and algorithms accessed by the programmer are done so in a python syntax, but are not necessarily pythonic ways of doing things. When using the Maya Python interpreter it is easy to see when standard Python is being used (Figure 5.7) and whne Python is being used to access Maya functionality (Figure 5.8). For a script to be completely portable it must be completely based on raw python and not rely on any API. Most likely a custom plugin will need to access an API specific to one program, in this case the core of the plugin, the algorithms and such, can be written in one module, and the parts specific to the API written in another.

In addition to the computational theory of algorithms, python introduces additional challenges. As it is not a low level language, and when used as a wrapper for an API, python can be extremely slow. In Maya, python can be used by itself, to access Maya scripts, use Maya commands, or to access the Maya API. Each of these options provides a different level of speed for a given task. While using python to call pre-made Maya scripts saves time on the programming side, it may drastically slow down the execution time. For example, an invisible 3D grid of 50x50x50 boxes exists in space. An object with a random start position is moved by the user and a program should tell the user which box is closest to the object by making the box visible. Each time the object is moved a new box is visible, and if the object is moved outside of the volume, no box is made visible. This can easily be done by using internal Maya commands. For example, there is a Maya command to create a box, move a box, query a box location, and query an object

location. The program could first create a box grid of 50x50x50, then make them invisible. When the user moves the object the program can then query the object, query each box location and then find the one with the shortest distance and make it visible. While from a programming standpoint this is a quick way to implement the function (and not very intelligent), the execution time will be unnecessarily slow. This is because it is not python that is handling calculations, but rather python asking a Maya script to have the low level program do a calculation and send it back up. This adds enough complexity that for an operation to happen multiple times, like searching all the boxes in a large volume, the execution time is slowed down. A faster and more portable way would be to handle the calculations in the code. Instead of creating the volume and using the API to query the location, the algorithm can just query the location of the box, and use that xyz point to check against an array of xyz points representing the location of a box. This eliminates the need to spend computational power creating all of the boxes ahead of time, but requires more math implemented in the algorithm.

Throughout the research the algorithms were constantly re-written to improve efficiency, capability, and clarity. The challenge in creating optimal algorithms is the reliance on internal Maya structures. The creation of an indpendent program for efficiency is valid, however, useful simulation and visualizations do not necessarily need to be built the optimal way. This research is in a unique position. Python programming is a major part of the

```
if result =='OK':
    joint_file = open(filenam
    try:
        fieldnames={}
        fieldnames=joint_dict
        writer = csv.DictWrit
        writer.writer.writerc
        writer.writerow(joint
    finally:
        joint_file.close()
```

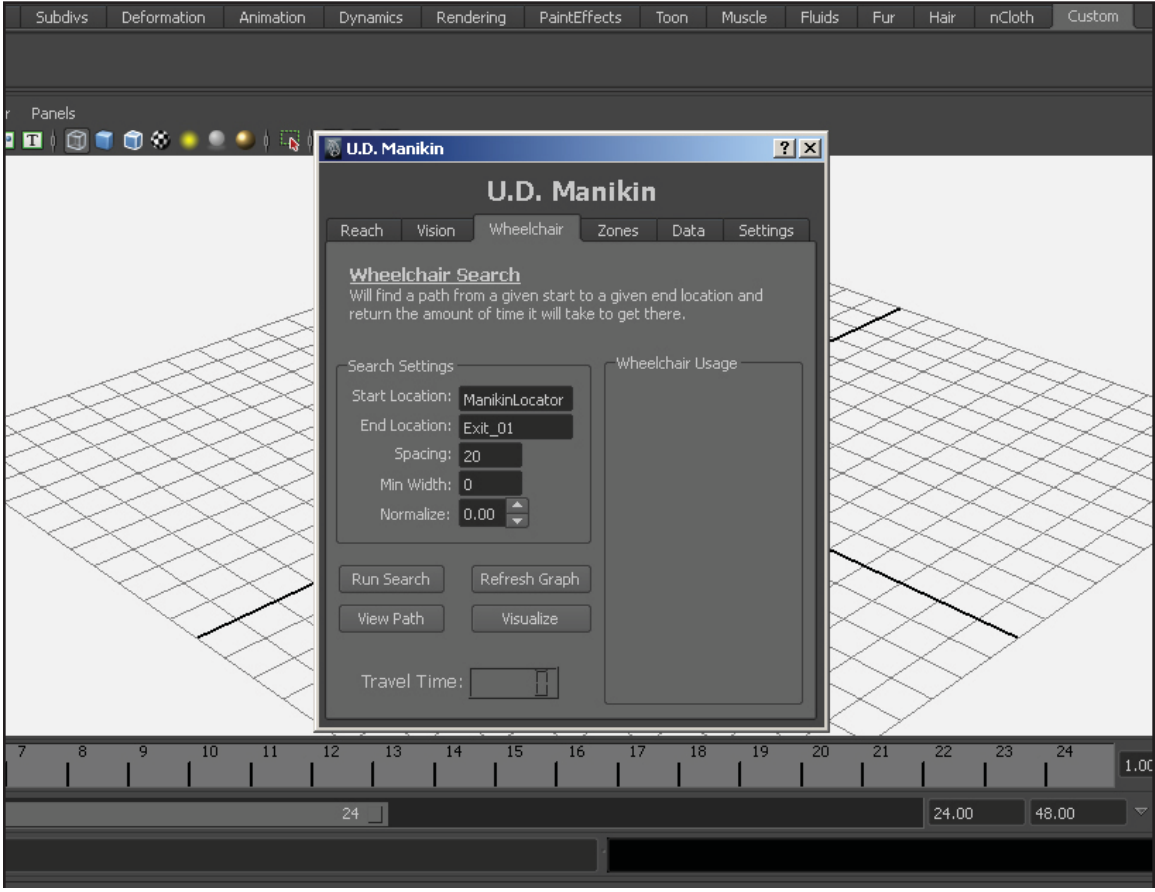
Figure 5.7

```
cmds.text( label='Y-Axis Limits',
cmds.intField(LFArm_Yhi, minValue
cmds.intField(LFArm_Ylo, minValue
cmds.text( label='Z-Axis Limits',
cmds.intField(LFArm_Zhi, minValue
cmds.intField(LFArm_Zlo, minValue

#Left Forearm Sliders
cmds.setParent('..')
cmds.floatSlider(LFArm_Y, min=-18
cmds.floatSlider(LFArm_Z, min=-18
```

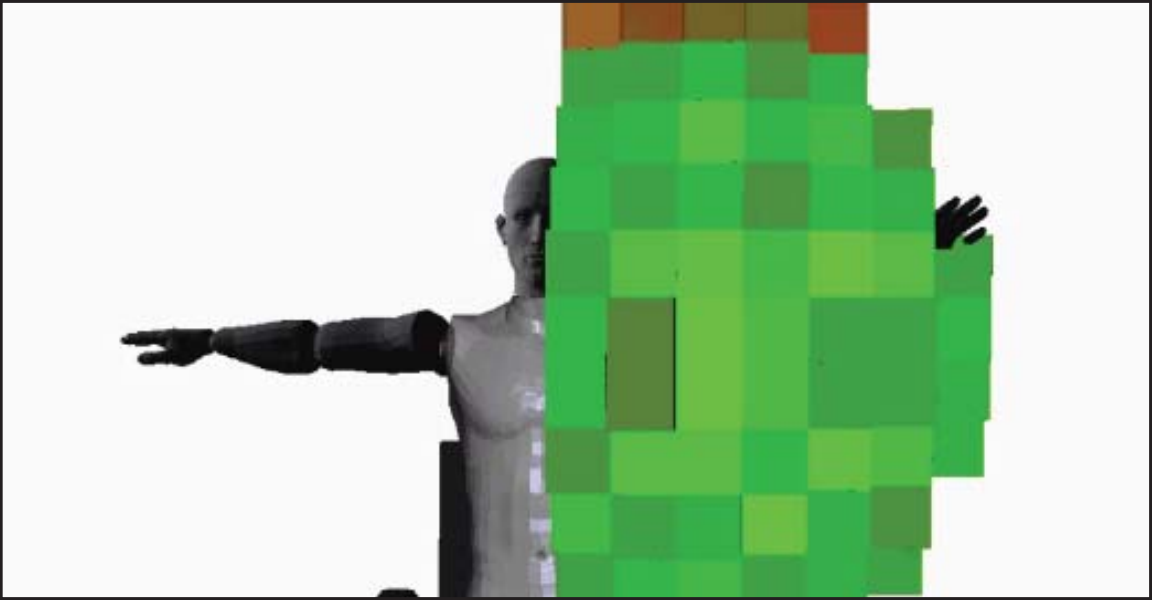
Figure 5.8

research, however, the main objective is in creating a usable tool. Hence the algorithms need to only be as efficient as a usable tool requires. This position does not however, mean none of the algorithms are efficient.



6 Implementation

The culmination of the research resulted in a Maya plugin named U.D. Manikin. The plugin interacts with a human model which for this research, is sitting in a wheelchair. The GUI is a fixed size and floats on top of the Maya UI. Each section in this chapter provides examples of the tools functionality. Currently the human model used with the plugin is rigged with the Maya joint tool. The arms have an inverse kinematic solver applied. Although the reliance on the Maya IK solver limits portability, a custom IK solver was beyond the scope of the research. The Maya IK provides a robust set of options meant for animation, which makes specifying conditions easy. Although implementing external solvers, either by writing a custom one or using OpenRave (Diankov), is the next step for continued research. The implementation started with the reach simulation. After the reach research was finished, additional factors were added to the tool. The final set of factors is not the end list. More human factors can be added, however, this research explored a few that would give a clear understanding as to how they could be implemented and used.



6|1 **Reach**

Reach
verb

Make a movement with one's hand or arm in an attempt to touch or grasp

From the early stages of the research human reach ability was the main focus. Simulating and visualizing reach ability presents a few challenges. For simulation, kinematic models can be used to recreate the ability of the arm. Early stages of research resulted in a forward kinematic model that described the reach ability in relation to maximum reach (Schwartz). This type of simulation brought up a very specific question in what the application of simulating reach is for. Simulating a persons ability to reach as far as they can is useful for different situations than simulating the complete reach ability. Although there are situations in which a full reach ability simulation is needed, the most immediate example for needing the simulation is when the reach ability is a-typical. A unique reach ability is frequent on people with a spinal cord injury (Curtis). These unique situations, as well as the natural bio-mechanics of the body demonstrate that a simulation of the human reach extent is not all-inclusive. This means that while someone may be able to reach to a certain distance, they may not be able to reach every

point from their body to the extent. When designing on the human scale, the knowledge of where someone can reach is extremely important.

Simulation of reach ability is done through a kinematic model. While the first tests in kinematics were done with a forward kinematic model, the preceding tests used an inverse kinematic model. One problem with inverse kinematics is the difficulty of generating a solution specific to a geometry with reasonable computation time (Jazar). This difficulty stems for a couple of reasons, some of which can be reduced. There is not a one size fits all model for inverse kinematics. The number of joints and length of links requires a unique kinematic model. Although there are ways to build up kinematic chains, the complexity requires large amounts of computational time. Specific models aim to solve this complexity. In the case of redundant chains, where there is more than one way for the end effector to reach a specific point, multiple kinematic solutions exist (Figure 6.1). In terms of positioning a chain, only one of the multiple solutions can be used. There are many ways for deciding which solution is used, the easiest being to use the solution closest to the previous solution.

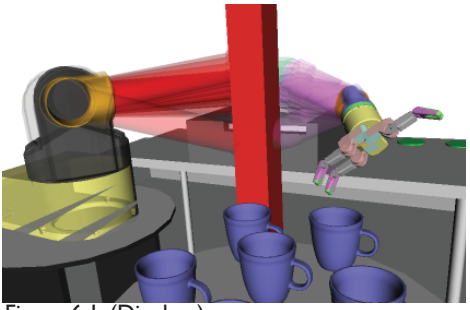


Figure 6.1: (Diankov)

Inverse kinematics are used mostly in robotics and biomechanics. The two subjects usually have very different applications, however, the field of humanoid robotics blurs the line between robotics and biomechanics. Research on humanoid robots strives to find efficient inverse kinematic solutions while finding solutions that mimic the movements of a human.

When looking at the automotive industry the current solutions and research are highly invested in the reach envelope. This reach envelope is similar to the body envelopes used in ISOKIN. These envelopes represent the extents of the reach ability. Mostly representing the envelope of furthest reach ability, they may also represent the envelope of most comfortable reach ability (Figure 4.5). For many situations these envelopes are ideal as the situation only requires knowledge that someone is able to reach a specific point. For a more thorough understanding of the reach ability of a person a full reach simulation must be completed. These full reach simulations are much less common, and only a few papers have been found on the subject (Rodriguez, Zacharias).

As most of this research is meant to be applied to spinal cord injury simulation, the use of reach envelopes may be misleading for a designer. While good at demonstrating the furthest reach capabilities, these envelopes give no indication to the designer of what the interior reach ability is. When dealing with a situation in which the human being simulated has no physical disability, and there are no movement limiting factors, an envelope may be sufficient and any missed information may not have many consequences. However, in the case of a physical disability, or more broadly any situation in which the human being simulated has limited movement capabilities (Figure 2.13), such as being in a tight space (airplane, rocket, submarine, car), understanding the kinematics of the upper body and its full reach ability are critical.

Current product solutions do not offer a robust enough system to simulate special case conditions for spinal cord injuries. In some cases enormous amounts of time can be spent attempting to create a simulation mimicking the special case (Hamameh). The issue with this is the user must already have the knowledge of how the simulation should work. The goal of this research is to expedite the users research and understanding by providing a useful simulation that can be trusted.

The inverse kinematic solver used for this research was through Maya. The end effector could be manipulated through the Maya UI or through scripting. Once moved, the Maya IK solver moves the joints to the solved angles. If the end effector is moved to a position which can not be solved for, the end effector is automatically moved to the closest position that has a solution. Since Maya will always return one solution, the issues of multiple solutions are not addressed here. However, when Maya moves the end effector to the closest position, it is really returning a null solution for the specific target. The way Maya handles the IK solver was deeply integrated with the logic of the reach simulation.

The most important note on the reach simulation is that only the shoulder and elbow are creating the reach map. This is for two reasons. First, the use of a 3+2 axis chain make the IK simulation less computationally expensive by treating the hand as the end effector instead of the fingers. Second, the research on biomechanics suggests the reach ability should be

more focused on joint ability in the arm than the entire body since some people may not be able to move their torso freely.

A voxel grid is used for supplying location input to the simulation. The end effector is moved to each voxel in the grid. If after solving the new location the end effector is in the desired location and has not moved, the voxel is given a value. This is a simple explanation, the full method is more involved. Since the IK solver is returning one solution per location, the number of solutions at each coordinate can not be used as values for reach ability. The IK solver returns a value very close to the previous value when being moved. This behavior is in line with techniques to reduce solutions by eliminating drastically different solutions from the previous one. The use of the manikin as a design tool takes advantage of this behavior by having the user position the manikin for simulation (Figure 6.3). When the user positions the manikin, the movements to all nearby voxels will be based on the original position.

Since the simulation does not use the wrist as a joint for determining approach angles, a different method was created to decide the value of a voxel. From the starting location, the end effector moves in all directions away from the center (Figure 6.4). If a solution is found, the new location is added to a queue. Each element of the queue is accessed, the end effector moves to the voxel, and then tries to access the bordering voxels. Each time the end effector lands inside a voxel, the voxels value is incremented. Each voxel can be accessed 27 ways, or from 27 of the bordering voxels. The advantage of

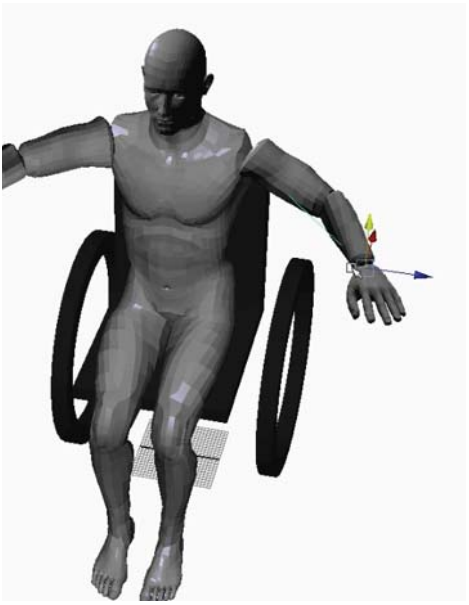


Figure 6.3

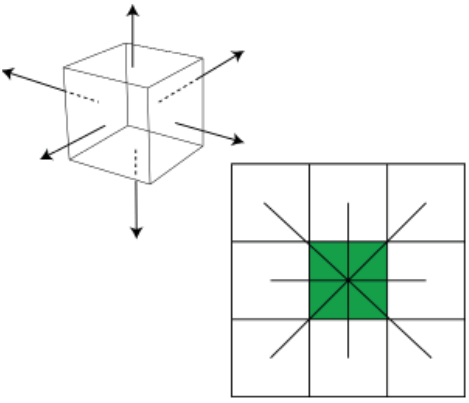


Figure 6.4

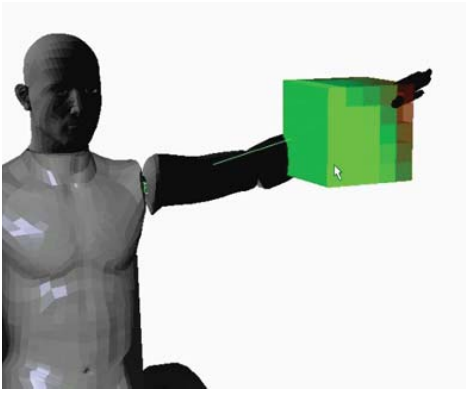


Figure 6.5

the user positioning the start location is the higher probability of accurate results.

The voxel array is created with the center point being the start location (Figure 6.5). This method makes the queue stack centrally located so the solver does not need to compute large movements that may require path planning to give realistic joint angles. The centralized stack also makes smaller volumes more consistent than randomized movements.

Visualization of the reach simulation is done by representing good values with green, bad with red, and a gradient between. To resolve the issue of required slicing seen in Zacharias, bad values are transparent while good values are opaque. Through the logic of a good value being a voxel accessible from all sides, the furthest voxels had somewhat low values (Figure 6.6). As the low valued voxels are more transparent, the user is able to see through the low valued exterior voxels to the higher valued interior. This allows the designer to instantly see if an object is outside the reachable volume, in a hard to reach volume, or hidden in the easy to reach volume (Figure 6.7).

The most important discovery in this system is the unique areas of difficult reach. Although the reach envelope is a complete arc based on the shoulder pivot, there are areas which are within the envelope that can not be accessed without torso movement (Figure 6.8). It is these areas, along with the many other unique situations with spinal cord injuries that makes reach simulation and visualization important.

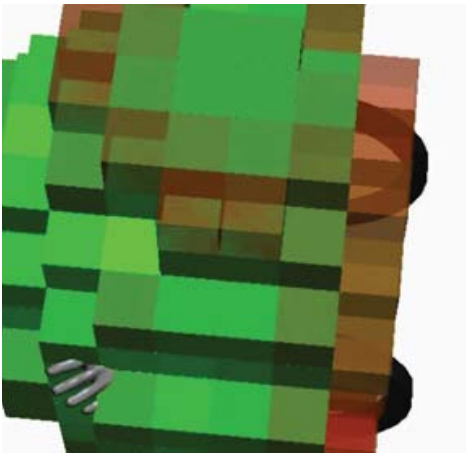


Figure 6.6

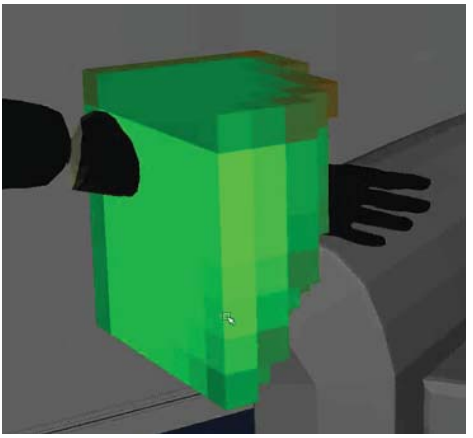
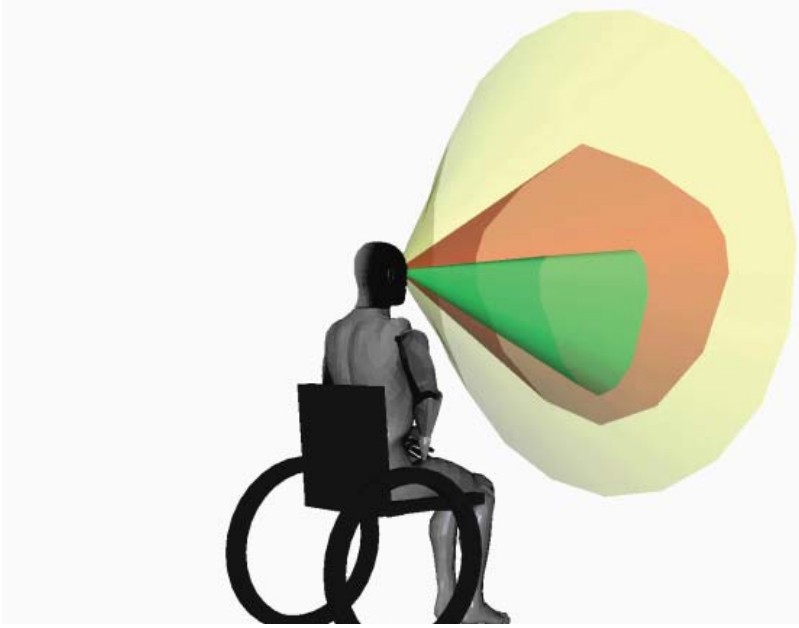


Figure 6.7

Figure 6.8



6|2 **Vision**

Vision
noun

The faculty or state of being able to see

Human vision capabilities can vary greatly. Unlike near or far-sighted vision problems, many other vision issues can not be fixed with glasses. Constraints on vision is also not limited to the eyes. Being able to rotate the head and lighting conditions affect the way humans can see the world. Even with perfect or corrected vision, the eye is only able to interpret certain information in specific zones. These zones include what is legible text, what is comfortable for eye rotation, what is the maximum the eyes can rotate, and more. As designers, knowing where someone is looking is nearly impossible. When designing a space, signs are given specific criteria for where they need to be. However, other than legal signs (ADA), the placement of visual cue's can be a challenging task. During the original motion capture session done for this research there was a surprising find. When tracking the head movement of someone getting out of a chair, it was noticed that the head was largely aiming down as the person stood up (Figure 6.9). This finding led to the research done on vision envelopes. By including vision envelopes on a manikin

within the model, the designer is able to see not only the extent of vision, but key areas such as the legible text zone.

The diagrams found in design books explaining human vision were overwhelmingly complex (Figure 6.10). The amount of information being conveyed was much less that it first seemed. This was partly due to the need for multiple angles and sections to describe 3D Zones. To make an algorithm that could stay flexible, some trigonometry was used to describe the view cones (Algorithm 1). This algorithm can take angles of extents from top and side view and create the zone they describe. This allows for one short algorithm to be used for creating all of the zones.

In addition to human limitations, external sources such as lights play a key role in vision. Lights placed within a certain range can cause glare. Glare can not only cause discomfort but can also make task completion difficult. If designing an office space, lights placed in certain places may make it difficult for workers to see, resulting in a lack of productivity.

To test for lights in a glare zone the program searches for objects with the name "Light" (Code 1). Each light is then compared against the location and direction of the manikin (Algorithm 2). If the light is found to be within the glare range, a red sphere is placed around it to notify the user (Figure 6.11).

Multiple vision cones can be displayed at once (Figure 6.12), or individually (Figure 6.13). The cones are always linked to the manikins head so the designer can interactively design with vision in mind.

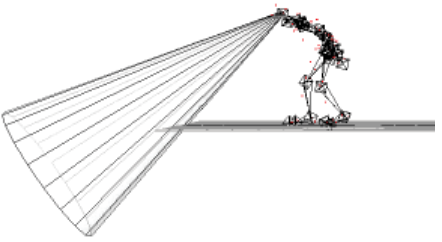


Figure 6.9

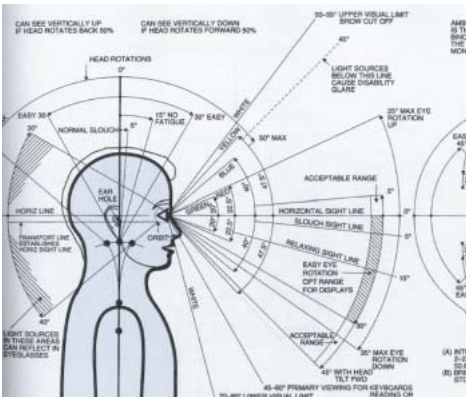


Figure 6.10

$$\sqrt{\left(\frac{distance}{\sin(90-Limit)}\right)^2-\text{distance}^2}$$

Algorithm 1

```
SceneLights = cmds.ls(type='light')
for light in SceneLights:
    try:
        light_id=light_id+1
        light = 'Light_%d' % (light_id)
    except:
        print 'exception: no lights available'
        return 0
```

Code 1

$$\tan^{-1}\left(\frac{Light\ Height}{Light\ Distance}\right)<90$$

Algorithm 2

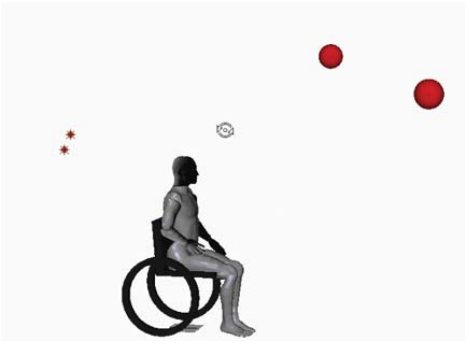


Figure 6.11

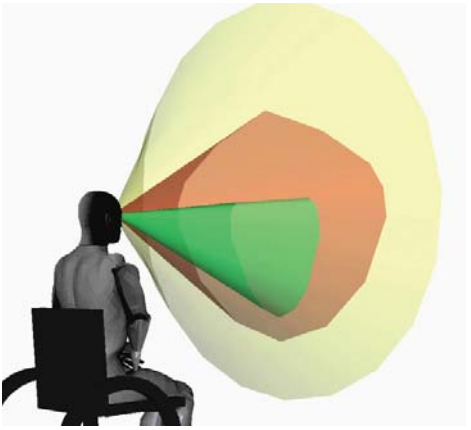
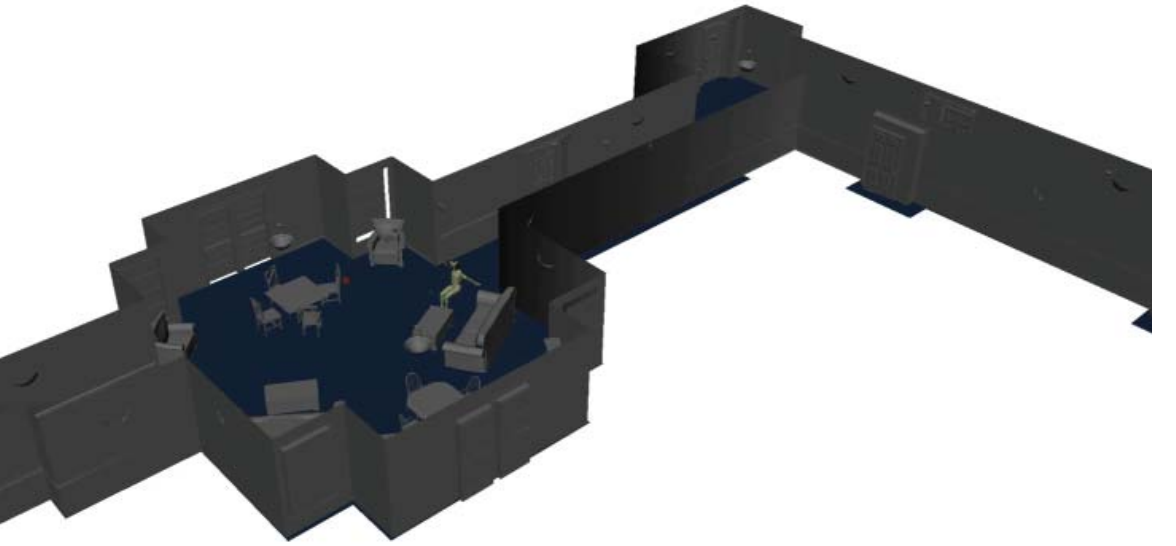


Figure 6.12

Figure 6.13



6|3 Wheelchair Navigation

Navigation
noun

The process or activity of accurately ascertaining one's position and planning and following a route

The human factor research conducted here has largely revolved around disabilities. Specifically, the use of a wheelchair constrains many problems with human factors research and allows for visualizations and simulation methods that may not be valid when a wheelchair is not being used. To understand the problem as more than just the human factors while being in a wheelchair, this research looks at the human and wheelchair as one entity. This brings the scale from local design to large building design and planning. The idea is to setup a platform for which placements of elevators and stairs, as well as handicap accessible places can be strategically and thoughtfully planned. As a general guide, this could be used to make sure the entrance of the building is not too far from an elevator, or the elevator is not on the opposite side of the building from a wheelchair accessible bathroom. This work focuses on wheelchairs, however, there are more complex situations in which this research could be applied to. When looking at someone

using crutches, or a prosthetic, the energy required to walk up stairs or take an elevator is much more complex than the single option of a wheelchair taking an elevator (Cerny). The decision for someone at the entrance to walk up the close set of stairs, or walk further down the hallway to get to the elevator, is a common and highly difficult problem. Understanding these issues, with a simple way of testing them not only can inform in the design process, but also provide a statistical understanding of the building that can inform users of what the best route for them to take is.

Way finding is a problem for all people. When dealing with disabilities, way finding becomes a more critical problem. For someone who is in a wheelchair, knowing how to get to the second floor, or find the nearest bathroom suitable for their needs can become a lengthy process. As a designer, knowing when these situations happen may lead to better methods for creating spaces that help inform way finding. Informing a user with navigation clues can be anything from stickers and signs to changes in building materials. Using search algorithms, a designer can place a virtual user in their model and find out what the shortest path for that user to take is. From there a designer could either redesign, knowing that the path is extraordinarily long, or help put navigational clues that expedite a person finding the path and minimize the time the user spends navigating to the desired location.

There are many different algorithms for finding the shortest path from one point to another. Different situations require different algorithms for various reasons including speed and complexity. Relative to buildings, the most common usage of a search algorithm would be Egress. However, Egress algorithms are more based on crowd studies and location based decisions than strictly a search algorithm (Kuligowski). Search algorithms can be useful in a variety of other situations including design. The extensive use of the Dijkstra's search algorithm in all different fields, led to this work using it as the basis for a wheelchair search. Dijkstra's algorithm, although not as fast as the A* predecessor, is a robust algorithm that finds the guaranteed shortest path based on a grid containing nodes and edges (Dijkstra). This work looks at the methods in creating the grid and calculating weights for the edges.

Dijkstra's algorithm is guaranteed to return the shortest path. For simulation, the most accurate result is priority, and only if it is unobtainable should we settle for less. In many situations Dijkstra's algorithm is too slow. When traversing a large network and updating in real-time, as is the case in many electronic games, Dijkstra's may run too slow and require the use of A*, which gives a significant performance boost but does not guarantee the most accurate result.

Although Dijkstra's algorithm is known, the edge weights and implementation are a much harder problem. First we look at creating the nodes. It may look trivial, but conducting a search algorithm on a building in a 3D model can not just be done. The search algorithm needs inputs. For a building, these inputs should be locations, and a value it takes to go from one location to

another. In a simplified graph, a node could be placed at each key location, and a value created to go from each of those locations (Figure 6.14). This however, requires a knowledge of the building, something an algorithm does not have. To the algorithm this graph is a Python dictionary. The top level keys are locations with the values being another dictionary pair. The secondary key is a location accessible by the main key with its value being the edge weight between the top level key and secondary key (Code 2). By just giving it the Cartesian coordinates, the algorithm can only create a direct distance, with no knowledge of floors or walls, the algorithm cannot create an accurate representation of the cost associated with moving from one point to the next (Figure 6.15). In order to solve this problem the algorithm must be given a version of the building it can understand.

In order for the algorithm to be useful in a variety of ways and not require involvement from the user, a method for recreating the building into a graph was developed. Using the internal capabilities of Maya and its raytracing functions, a node based graph is created from a model by sampling vector intersections (Figure 6.16). The user must define a resolution they would like to work at. The higher the resolution, the longer both the graph creation and search algorithm take. The former being the most time consuming, yet needs to be done only once per building design. When the user selects a resolution, the location of the human manikin is used as the start location for the graph creation. From this point, a vector is shot down from the height of the manikin, if the

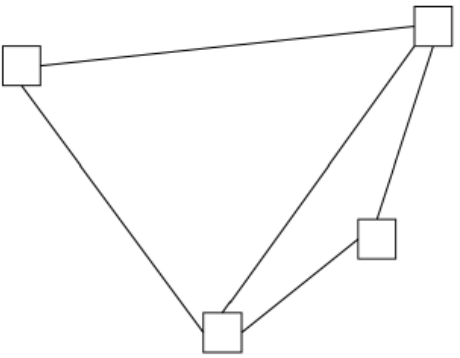


Figure 6.14

```
building_graph = {
    'Bathroom_01':   {'Room_01':50},
    'Room_01':      {'Bathroom_01':14, 'Lou
    'Lounge':        {'Bathroom_01':435, 'R
    'Exit_01':     {'Room_01':350, 'Lounge'
    'Exit_02':       {'Lounge':190, 'Exit_0
    'Exit_03':     {'Exit_01':210, 'Exit_02
    'Room_02':       {'Room_03':4675},
    'Room_03': {'Room_02':4675},
}
```

Code 2

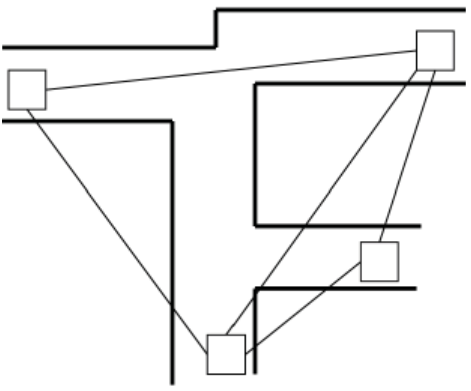


Figure 6.15

first object the vector intersects with is the floor, a node is created at that location. A queue is then formed of all possible move directions from the valid point. Additional valid points are added to a new queue. If a vector does not intersect the floor first it could mean there is an object, such as a table in the way, or the point has moved off of that floor and is outside the wall. There are a few options for increasing the accuracy of this grid. Each time a node is created a line can be drawn from that node to the previous, if the line intersects a wall then the node is not valid. This particular example added more computational complexity than necessary and was not implemented. Instead of focusing on representing the entire building accurately, constructing a graph based on the intents of the search allowed for a highly accurate and faster running search algorithm.

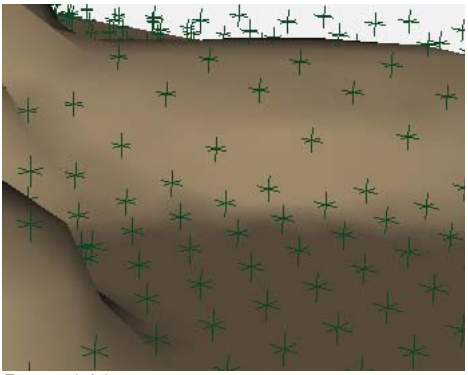


Figure 6.16

A wheelchair needs a specific amount of space in order to pass through a hallway. Additionally, a wheelchair has a specific turning radius required to turn around. These two requirements allow the graph to be minimalized for the search algorithm. If a hallway is too small for a specified wheelchair to pass through, the search algorithm should not take it into consideration as a possible node to move through. One method for telling the algorithm not to pass through the node is to add information to the node. For example, if every node has a value of 0 to 1, with an added value of hallway width, the algorithm can check the required hallway width against the nodes hallway width and if it is too short, the node is assigned an infinite value (99999999 for programming). This method is valid, and

may be effective if the wall calculations where done as well, however, this research took a different approach. In order to keep the complexity of the search algorithm down, the only nodes in the graph are ones that can be passed through. This means on creation of the nodes, a node should only be placed if the hallway width at that point is large enough for the specified wheelchair. This was done using the circumference of a circle with the diameter matching the width (or turning radius) of the wheelchair (Figure 6.17). If a point passed as valid, multiple extra points (minimum four) are then checked along the circumference of the required width. If a point on that circumference is not valid, the original point is not considered a valid node. The additional check on each point slightly changes the algorithm by adding an additional queue set. This reduces the nodes in the graph and speeds up the search algorithm.

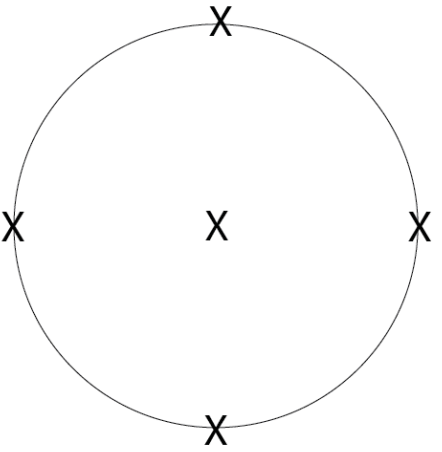


Figure 6.17

The edges that connect each node determine what the shortest path is. For human movement, the shortest path is not always the best path. Best is decided as the optimal balance between distance and ability. For example, although the stairs uses a shorter path than an elevator, a wheelchair is not able to use the stairs, so the shortest path is not the best. Dijkstra's algorithm is not designed for these cases, however, the information can be calculated before Dijkstra's algorithm analyzes the paths. For this, more factors than only node distance were used to calculate edge connections. Instead of separating the distance of each node and the difficulty to move to each node, every edge value is calculated individually. Although the actual numbers used

are not ideal, further research on this specific issue will provide insight to the most accurate configuration while still utilizing this method. The edge values are considered a combination of the distance and angle from the nodes (Code 4). Additional factors such as floor material could be factored in as well to give a more robust assessment. For this base system the 3D distance between the nodes is calculated, then the angle from the start node to the end node is calculated and factored in. As a starting point, the angle factor is the natural log of the angle, added to the distance. For angles that are negative, meaning the edge is going downhill, half of the angle calculation is added to the distance. This is for the ease of moving down hill, yet the still present energy requirement to prevent a wheelchair from rolling down hill, making it more difficult than moving on level ground.

Previous research has found a dramatic increase in the energy needed to move a wheelchair across a floor with a compound angle (Brubaker), further research into this can be factored in to the edge values. Currently this extra energy is only visualized through the color coding of nodes (Figure 6.18). Compound angles mean the node has a higher value on each of its edge connections, when these are added together, the higher value means that the node is on an area that has multiple angles. This higher value is represented when the user visualizes the graph, but is currently left out of the edge calculations as it drastically increases the computation.

In addition to returning the time required to move from one point to another, the user is able to visualize the

```
p = current
q = candidate
dist = abs( distFunc(p,q) )
if p[1]==q[1]: angle=0
if p[1]!=q[1]:
    side_c = dist
    side_a = distFunc((p[0],0,p[2]),(q[
    side_b = sqrt( (side_c**2) - (side_
    if side_a==0:
        side_a=1
    angle = degrees( atan(side_b/side_a
if angle!=0: angle=log(angle)
if p[1]>q[1]:
    angle=(angle)/2
if angle>1: angle = 5
if angle>1.4: angle=20
if angle>1.7: angle=99999
edge = dist+angle
```

Code 4

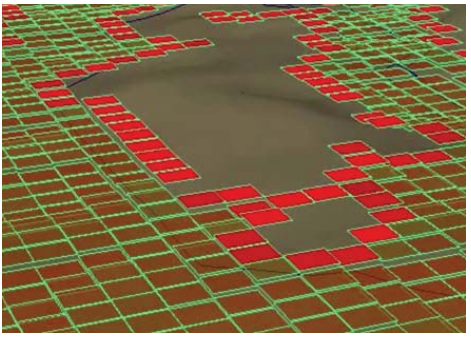


Figure 6.18

graph. The visualization is based on the value of each node, which is calculated by adding all of its edge values together. The nodes are then normalized from zero to one. The value dictates the translucency and color. High values are more costly and are opaque red, while low values are less costly and a translucent green (Figure 6.19).

When dealing with requirements and standards these methods of calculating edge weights are very useful. When dealing with wheelchair ramps, one can set the algorithm to create an extremely high value for any edge with a slope higher than 12:1. The combination of the slope value and research on energy expenditure allows for a tool to not just analyze a design, but interactively help a designer understand the energy needed to navigate their design. This can be used for instance, in designing a wheelchair ramp. If a wheelchair ramp can be placed on two different sides of a building, one with a longer ramp, and one with a shorter ramp, the best choice is not necessarily known (Figure 6.20). A longer ramp will allow for a smaller incline, yet extends the amount of time needed to finish the ramp. By incorporating more research on the subject, this type of algorithm provides a great way for designers to improve their design in a quantifiable manner.

Figure 6.19

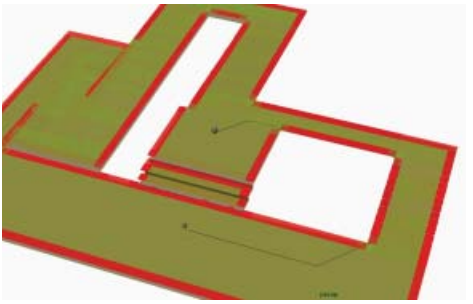


Figure 6.20

6|4 **Zones**

Zone
noun

An area or stretch of land having a particular characteristic, purpose, or use, or subject to particular restrictions

The amount of space someone takes up depends on not just their weight and height, but the tools they use in everyday life. These tools can range from a backpack full of books, to a wheelchair, crutches, or walking stick. There are many references for designers on the size people take up, including references for disabilities and clothes (Figure 6.21). In fact, the references vary from general sizes to extremely specific, taking into account winter sized clothes, personal space, touch-zones, and on. Knowledge of the size required of someone is useful in a variety of situations, even more beneficial is a quick way of seeing this requirement in the design. Just having the knowledge of the size required can not last through the entire design process. At some point the space needs to be measured against the required space for a person. This is especially true in bathroom settings that have strict laws on accessibility. Less in the realm of simulation and more towards pure visualization is the ability to view these zones right inside the design. Additionally,

this collection of zones provides a reminder to designers as to what is not just the minimum, but what is the real situation in which design should aim for. For example, although the ADA specifies a turning radius for a wheelchair, it does not include the situation in which the person in a wheelchair has an assistant that must be included in the turning radius. Additionally, while the space required for a wheelchair to pass through a hallway may be known, the designer must remember that in a long hallway, leaving just enough room for a wheelchair to pass may cause a problem when a wheelchair is passing and another person is walking in the opposite direction, let alone two wheelchairs passing each other. These additional cases make memorization of size requirements an extremely difficult thing to do.

The space taken up by someone can be represented in different ways. The physical space taken up by someone is different than the space needed for an action to occur. A main influence in this part of the research was the dissertation by David Lantrip on quantifying space design (Lantrip). His research consisted of cataloging different movements through video. These movements turned into a body envelope that was integrated into a program called IsoKin. The program itself is interesting in comparison to the current research as a reference to how technology and workflows have changed over twenty years. What has not changed is the relevance of understanding how a desired human action results in the perceived comfort of the space. The answer is not always as simple as making something bigger.

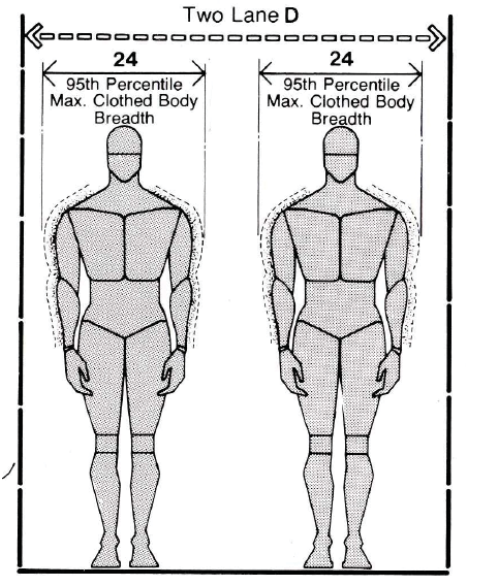


Figure 6.21

Figure 6.22

Besides for the obvious fact that making everything bigger requires more space and money, it also means everything is further apart and takes more time to go from one place to another. Although this distance may make the environment seem less cramped, it does not necessarily make the space more comfortable as discomfort may arise from aggravation in the length of time required to complete a task.

There are many types of wheelchairs. Unfortunately the knowledge of these wheelchairs and how they function is not known by most designers. Besides the legal standards, there are various wheelchair sizes, and just as important, various sized people that use wheelchairs. At a local hospital, three different wheelchairs were being used. With low-cost technology the chairs were 3D scanned to database different sizes (Figure 6.22). The turning radius of a wheelchair is not just a calculation of wheelbase and length, but must also include the foot length of the user, and the users arm. When an assistant is needed to push the wheelchair, the turning radius is drastically increased. Simulation comes into this section when looking at the vast array of wheelchairs. A designer working for either a health center that uses a few types of wheelchairs, or designing for a single client with one wheelchair type, benefits from being able to simulate the wheelchairs space. With memorization of the entire legal standards for accessibility a designer is still not able to provide an optimal design for a client if the wheelchair is different than the standard. Most drastically is a client with a motorized chair compared with the wheelchair

Figure 6.23

Figure 6.24

used for legal standards. Everything from the turning radius to seat height is different and must be designed for. There are two ways of adding the various wheelchairs in a visualization/ simulation program, first is to have a catalog of various wheelchairs and the designer chooses the appropriate one. Second, the designer could enter basic information about the wheelchair and have the simulations done after entering the information. These methods are not mutually exclusive and could provide an incentive for community based data sharing. As specific wheelchairs are measured, they could be added to a larger database in which designers pull the premade simulation data for visualization in their designs. Additionally, the database is not restricted to designers, meaning manufacturers and user of wheelchairs can enter the data to contribute to a growing collection of wheelchair data.

Using a similar technique as the vision visualizations, a single algorithm was implemented so different zones could be easily added. The zones are based around the manikin and can be visualized individually (Figure 2.23) or in multiples (Figure 6.24). Although the manikin uses a wheelchair, standing zones are also implemented. These zones, represented in design books as a way to design space for numerous people (Figure 6.25), are used to remind the user of how much space a wheelchair needs in comparison (Figure 6.26).

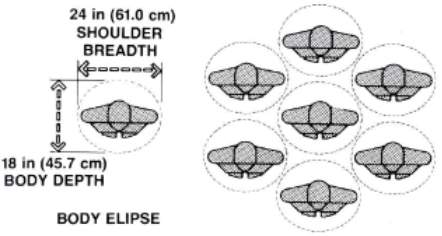


Figure 6.25

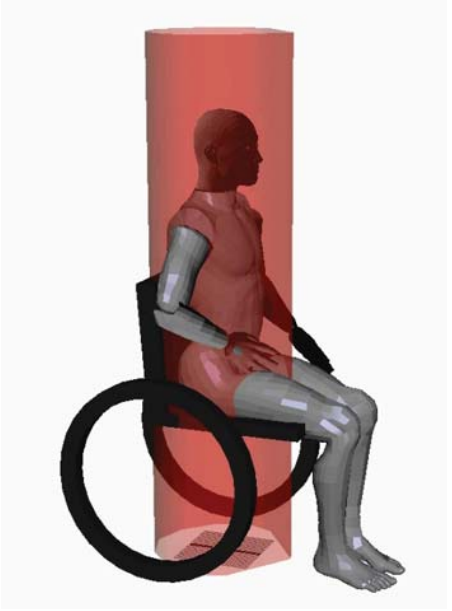
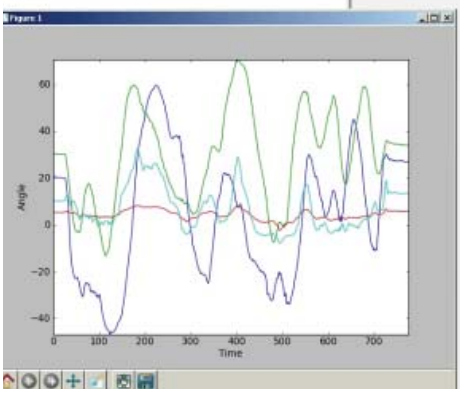
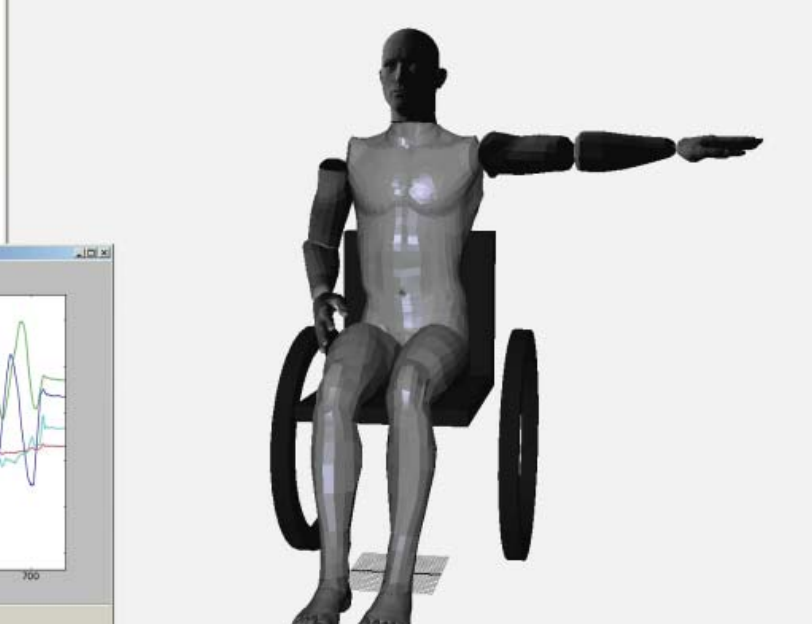


Figure 6.26





6|5 **Analysis**

Analysis
noun

Detailed examination of the elements or structure of something, typically as a basis for discussion or interpretation

Creating a way for designers and scientists to work together was a priority of this research. Through the motion capture work done at the beginning of the research it was clear that many of the methods and techniques scientists used could be integrated in design programs. The first obstacle was that not all modeling programs have character animation built in. Although this work was developed in Maya, which did have character animation, the goal was to create a system that could be ported to any modeling program with scripting capabilities. The lack of animation capabilities was resolved by reading and writing .CSV files. This concept came from the way in which movement science software packages already play animation. The files containing motion capture data are usually a spreadsheet of markers and xyz locations throughout time. Instead of using xyz marker locations the files in this research use joint angles. This can be done on either the movement science or 3d modeling side. In movement science someone can use a program

such as Visual3D to read mocap files, construct a skeleton and export the joint angles. On the 3d modeling side Motion Builder can be used to take motion capture data and attach it to a skeleton. The skeleton can be brought into Maya and the angles recorded. Either way, the joint angle information can be analyzed through plot graphs.

A key part to this research was in finding existing libraries with functionality that would benefit the design practice. One of these examples is a matlab style plotting library called matplotlib (Hunter). This library has a plethora of plotting styles that can take data and compare it in ways from line graphs to animated 3D graphs. The library was successfully imported within Maya. Integration with the manikin tool provides a quick way to compare joint angles over time.

7 Conclusion

This work demonstrates a variety of functions, algorithms, and user layouts that can better integrate human factors with the design workflow. There are many ways to continue this work both by the author and readers. Each of the algorithms has potential to be both improved with speed and expanded on for functionality. The current works sets a groundwork for how to integrate the new functions that may become useful to designers and architects. With the continuation of work involving BIM, factors such as the human body and the relationship it holds to the building will hopefully become an integral feature of future BIM applications. As discussed in the work, the functions presented are not limited to BIM applications. By bringing human factors to the forethought of designers and architects, the way and style in which designs are created can be greatly influenced, while also being more user friendly.

The goal of this work is to continue refinement within the algorithms and create the most portable library for designers to integrate with their own work. Some additional work and side projects have been inspired by going through this

work. For one, a library that takes basic python functions used in a variety of different programs and extrapolates them to a module to act as a wrapper for cross platform building would be very useful. By adding a module that handles the communication with the program, such as Maya or Rhino, the UD Manikin could be built with simple functions abstracted from program specific language, and with a simple variable switch the library could output to the program either Maya or Rhino style python.

If at the very least, this work should help people think more about the human experience of their designs, not just for the average user, but for people that have physical disabilities that the designer may not have thought of, such as a different reach ability. This extends not only to the design process, but building information that could be displayed and known by the occupants that can assist in someone attempting to navigate or use the current facilities.

8 Acknowledgments

Many people have helped in this work in a variety of ways. From helping with technical problems with programming to guidance on the overall work. I could not have done it without these people. A few names that I would like to mention have had a strong direct impact on the work:

Karl Daubmann
Forest Darling
Melissa Gross
Steffen Heise
David Lantrip
John Marshall
Josh Marshbanks
Mojtaba Navvab
Sean Vance

I would like to give a special thanks to Patrick Slater, who set me on the right path many years ago.

9 References

'2010 ADA Standards for Accessible Design, Title II (28 CFR part 35) and Title III (28 CFR part 36)', Department of Justice.

Abdel-Malek, K.; D, P.; Yang, J.; Brand, R. & Vannier, M. (2001), 'Towards Understanding the Workspace of The Upper Extremities'.

Alberti, L.; Rykwert, J.; Leach, N. & Tavernor, R. (1991), On the Art of Building in Ten Books, MIT Press.

ASIA, 'Standards for Neurological Classification of SCI Worksheet', Technical report, American Spinal Injury Association.
Autodesk www.autodesk.com

Blanchonette, P. (2010), 'Jack Human Modelling Tool: A Review.', .

Boeykens, S. (2011), Using 3D Design software, BIM and game engines for architectural historical reconstruction'Designing Together - CAADfutures 2011', Les Editions de l'Université de Liège, .

C-Motion www.c-motion.com

Card, S. K., 'user interface research', PARC.

Castro, M. J.; David F. Apple, J.; Staron, R. S.; Campos, G. E. R. & Dudley, G. A. (1999), 'Influence of complete spinal cord injury on skeletal muscle within 6 mo of injury', Journal of Applied Physiology 68, 350-358.

Cerny, K.; Waters, R.; Hislop, H. & Perry, J. (1980), 'Walking and wheelchair energetics in persons with paraplegia', Journal of the American Physical Therapy Association 60, 1133-1139.

Clipson, C. & of Michigan. Architectural Research Laboratory, U. (1988), Simulating future worlds: a review of simulation techniques for research planning and design, Architecture and Planning Research Laboratory, College of Architecture and Urban Planning, University of Michigan.

Corbusier, L. (2004), The Modulor and Modulor 2, Birkhäuser Basel.

Crane, E.; Gross, M. & Rothman, E. (2009), Methods for Quantifying Emotion-Related Gait Kinematics, in Randall Shumaker, ed., 'Virtual and Mixed Reality', Springer Berlin / Heidelberg, , pp. 23-31.

Curtis, K. A.; Kindlin, C. M.; Reich, K. M. & White, D. E. (1995), 'Functional reach in wheelchair users: The effects of trunk and lower extremity stabilization', Archives of Physical Medicine and Rehabilitation 76(4), 360 - 367.

Delp SL, Anderson FC, Arnold AS, Loan P, Habib A, John CT, Guendelman E, Thelen DG (in press). (2007). OpenSim: Open-source Software to Create and Analyze Dynamic Simulations of Movement. IEEE Transactions on Biomedical Engineering.

Diankov, R. & Kuffner, J. (2008), 'OpenRAVE: A Planning Architecture for Autonomous Robotics', Technical report, Robotics Institute, Carnegie Mellon University.

Dijkstra, E. W. (1997), A Discipline of Programming, Prentice Hall PTR, Upper Saddle River, NJ, USA.

Eriksson, J. (1998), 'Planning of Environments for People with Physical Disabilities Using Computer Aided Design', PhD thesis, Lund Institute of Tech.

filho, G. B. G. (2005), 'Optical motion capture: Theory and implementation', Journal of Theoretical and Applied Informatics (RITA 12, 61--89.
Hamameh, R. (2010), 'Digital Human Models Of People With Disabilities', Master's thesis, Wayne State University.

Hunter, J. D. (2007), 'Matplotlib: A 2D Graphics Environment', Computing in Science and Engineering 9, 90-95.

Hutchinson, A.; Guest, A.; Balanchine, G. & Laban, R. (1987), Labanotation: The System of Analyzing and Recording Movement, Taylor & Francis.

Jazar, R. N. (2010), Theory of Applied Robotics, Springer New York Dordrecht Heidelberg London.

Jones, P. (1999), Hugo Haring: The Organic Versus the Geometric, Ed. Axel Menges.

Kuligowski, E. D. & Peacock, R. D. (2005), 'A Review of Building Evacuation Models', Technical report, National Institute of Standards and Technology.

von Laban, R. & Lange, R. (1975), Laban's principles of dance and movement notation, Macdonald & Evans.

Lantrip, D. B. (1999), Measuring Constraints to Inhabitant Activities, in Edward Steinfeld; G. Scott Danford; Michael Feuerstein & Anthony J. Goreczny, ed.,

'Enabling Environments', Springer US, , pp. 139-164.

MacKenzie, I. S. (1992), 'Fitts' law as a research and design tool in human-computer interaction', Hum.-Comput. Interact. 7(1), 91--139.

Moeslund, T. B. & Granum, E. (2001), 'A Survey of Computer Vision-Based Human Motion Capture', Computer Vision and Image Understanding 81(3), 231 - 268.

NASA (1978), 'Anthropometric Source Book Volume I: Anthropometry for Designers'(NASA RP-1024), NASA.

NCARB, 'Schematic Design Study Guide'.

Ousterhout, J. K. (1998), 'Scripting: Higher-Level Programming for the 21st Century', Computer 31(3), 23--30.

Owens, T. (2003), 'The enigma of multiple sclerosis: inflammation and neurodegeneration cause heterogeneous dysfunction and damage', Current Opinion in Neurology 16, 259-265.

Page, J. (1990), A Comparative Study of Two Movement Writing Systems: Laban and Benesh Notations, University of Sydney.

Panero, J. & Zelnik, M. (1979), Human dimension & interior space: a source book of design reference standards, Whitney Library of Design.

Pollio, V.; Morgan, M. & Warren, H. (1914), Vitruvius, the Ten Books on Architecture, Harvard University Press.

RehabMart www.rehabmart.com.

Rhino3D www.rehabmart.com.

Rodriguez, I.; Peinado, M.; Boulic, R. & Meziat, D. (2003), Reaching volumes generated by means of octal trees and Cartesian constraints, in 'Computer Graphics International, 2003. Proceedings', pp. 324 - 329.

Rossum, G. (1995), 'Python tutorial', CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, The Netherlands.

Sancho-Bru, J. L.; Pe?rez-Gonza?lez, A.; Mora, M. C.; Leo?n, B. E.; Vergara, M.; Iserte, J. L.; Rodri?guez-Cervantes, P. J. & Morales, A. Klika, D.V., ed. (2011), Towards a Realistic and Self-Contained Biomechanical Model of the Hand, Theoretical Biomechanics.

Schwartz, M. & Viswanathan, J. (2012), 'A Framework for Visualizing Biomechanical Movement for Designers Within 3D Modeling Programs"36th Annual Meeting of the American Society of Biomechanics'.

Seidler, H. (1975), 'Convent of La Tourette'.

Summerfield, M. (2008), Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming, Prentice Hall.

Summerfield, M. (2008), Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming, Prentice Hall.

Tilley, A. & Associates, H. D. (2001), The Measure of Man and Woman: Human Factors in Design, Wiley.

Tolani, D.; Goswami, A. & Badler, N. I. (2000), 'Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs', Graphical Models 62(5), 353 - 388.

University, N. S. (2010), What Is Universal Design, North Carolina State University College of Design.

Vance, U. S. (2012), equilibrium Universal Design Primer, University of Michigan Taubman College of Architecture and Urban Planning.

Vicon www.vicon.com

Wittkower, R. (1971), Architectural Principles in the Age of Humanism, W.W. Norton.

Yang, J. & Abdel-Malek, K. (2008), 'Human Reach Envelope and Zone Differentiationfor Ergonomic Design', Human Factors and Ergonomics 19, 15-34.

Zacharias, F. (2012), Knowledge Representations for Planning Manipulation Tasks, Springer.