

MSc Program "Building Science & Technology"

Analysis of Design Support for Kinetic Structures

A master's thesis submitted for the degree of
"Master of Science"

Supervisor: ao.Univ.-Prof Dipl.-Arch.Dr. Georg Suter
Department of Building Physics and Building Ecology

Angeliki Fotiadou

Vienna, June 2007

Affidavit

I, **Angeliki Fotiadou**, hereby declare

1. That I am the sole author of the present Master Thesis "Analysis of Design Support for Kinetic Structures" and that I have not used any source or tool other than those referenced or any other illicit aid or tool, and
2. That I have not prior to this date submitted this Master Thesis as an examination paper in any form in Austria or abroad.

Vienna, June 2007

Signature

ANALYSIS OF DESIGN SUPPORT FOR KINETIC STRUCTURES

FOTIADOU Angeliki,
Department of Architecture, Vienna University of Technology

Submitted to the Institute of Building Physics and Human Ecology, School of Architecture, in partial fulfilment of the requirements for the degree of Master of Science in Building Sciences and Technology at the Vienna University of Technology

Vienna, June 2007

Keywords:

Kinetic architecture, 3D designing software, scripting, programming

Abstract:

This thesis attempts the formation and systemization of a basis of knowledge and information, which is indispensable to turn a design support for kinetic structures into representation by means of a 3d animating program. Representation of kinetic structures by means of the existing ordinary software sources is possible; Nevertheless, such representation lacks of different important features and functions and results eventually in the total absence of a real model of the construction, which is valuable to the user of the program especially in the field of the kinetics, where everything depends on the movement: design not only requires, but demands for visualisation. A personal interest in kinetic architecture and therefore in the physical movement of structural elements in a building, as well as an attempt to "fathom" the possibility of changing this concept to visualization and modern reality by the use of a software are the main incentives of this master thesis.

First, a general research will be performed in order to check the existence of similar or semi-similar proposals. The area in which the research will be held is the Bibliography in kinetic architecture and parametric design. A comparison of animation and 3D prototype software in well-known programs will focus on whether virtual weather conditions are considered as a parameter to the animation of the structure of the programs and case studies of several existing kinetic structures will be performed, in order to point out flaws and/or helpful commands in the programs in connection with the presentation of kinetic architecture. Criteria for the choice of the software: ability to customise and to produce geometric modelling, animation in relation to time (video animation) and the simulation after taking into consideration weather factors. Finally, using the computer and the scripting language, based probably on the theory of parametric design and primitive instancing, a realistic simulation of different elements will be performed in relation to variable measurements of luminance, ventilation and temperature so as to render feasible the construction of a whole structure

The results of the thesis will be used in the future as the basic knowledge in the creation of software for simulation of kinetic architecture. This program will be used as a tool for the architect to present a building, where kinetic architecture will be applied and to create simulation of the kinetic movement through a library of the existing prefabricated elements which will be created with the help of this thesis

TABLE OF CONTENTS

Abstract	3
1 INTRODUCTION	6
1.1 Motivation	6
1.2 Background	7
1.3 Classification of kinetic structures	9
1.3.1 Typology of kinetic structures	9
1.3.2 Examples of kinetic structures	12
2 METHODOLOGY	14
3 EVALUATION OF ANIMATION SOFTWARE	16
3.1 Concept	16
3.2 Modelling kinetic structures in animation applications	17
3.3 Evaluation criteria	19
4.2 Comparison and choice of the software	20
4 CASE STUDIES	24
4.1 Approach	24
4.2 Generalization of steps needed for the creation of a kinetic structure	24
4.3 Case study 1: Kuwait Pavilion	30
4.3.1 Introduction	30
4.3.2 Construction of Kuwait Pavilion	32
4.4 Case study 2: Alcoy Community Hall	36
4.4.1 Introduction	36
4.4.2 Construction of Alcoy Community Hall	38
4.5 Case study 3: Folding Egg	44
4.5.1 Introduction	44
4.5.2 Construction of Folding Egg	46

5	ASSESSMENT	52
5.1	Benefits and limitations	52
5.2	Enhancements	55
6	CONCLUSION	60
	References	62
	Image sources	64
	Glossary	66
	Appendix A: Description of the software	74
	Appendix B: Description of scripting languages	86

1 INTRODUCTION

1.1 Motivation

Kinetic architecture, though it is not a newly discovered concept, it has been scarcely applied until recently. The explosion of technology enabled kinetic architecture to make a spectacular comeback and attack traditional architecture, due to the powerful combination of manufacturing and use of technologies, sustained by kinetic architecture. Increasing needs for time saving and use of technological inventions of different mobilities and automations, which promise to optimise and provide to the inhabitant a better living, set off a new era in building design. (1) The use of robotics in a building, either during the construction or for inhabitancy needs, (2) the “smart houses” which, with the use of computers and sensors, are designed to satisfy the inhabitants’ basic needs (i.e. ventilation, sun protection etc.), and the (3) transformation of the shape of a construction by mechanisms which allow adaptation either to environmental conditions or to the will of the user, and many more constitute expression of kinetic architecture. The term “kinetic architecture” is difficult to be defined, as a lot of architectural functionalities are described as such. Each one of them seems to correspond to different parts of a construction, to the inner part, the outer, specific elements like shades or even appliances and therefore terminology can only be different for every case.

However, focusing on one particular category of kinetic architecture is the theme of this thesis. This category involves with the movement of large scale elements of kinetic structures, named Embedded Kinetic Structures (see: Chapter 1.3, Clarification of kinetic structures). The Embedded kinetic structures seem to become main trend in the construction field. Many examples can be seen all around the world, like the famous Planetarium in Valencia, which, by combining the functionality with the aesthetics, is classified among the most important architectural achievements of the century. Nevertheless, it seems that this specific area of kinetic architecture is less developed than the others.

Throughout the bibliography and internet, it is evident that a lot of researches are being conducted over the study of kinetic structures, either by practical or computational way, regarding those which concentrate either to inner-construction mobility, or to the limited outer-shell mobility like sensor-based façade deformers, inner spatial arrangers, projects like House-N and software that are needed for the study or the function of these constructions. Nevertheless, the examples of kinetic structures which can adjust part of their figure in association with different factors scarcely appear and those that exist are only practically performed, meaning that models are being built in appropriate scale to taste the performance of the construction. Moreover, no appropriate software seems to exist in order to support the study of these constructions; such software will act as simulator and movement analyzer. Nowadays, when every single everyday life manifestation is strongly related to the use of a computer, from the simple use of internet through a medical operation, the lack of the supporting software appears as the major disadvantage.

There is dissension about whether the lack of software is a sign of the overall doubt regarding utility of kinetic architecture. However, taking into account the many possibilities and the advantageous features of kinetic

architecture, it is easily foreseen that kinetic structures could clearly meet the needs of modern society like practical, aesthetical, interacting with the environment etc. as well as support and accomplish multiple functions; maybe all these at the same time.

It is my strong belief that kinetic architecture is the future of architecture and substitute static and not reformable, in other words “traditional” architecture. The automation and mobility offered can provide to the inhabitant of a building new forms of freedom and comfort in an artificial environment. As the society and the needs of modern man are becoming more unstable, not in terms of danger but as constantly changing, the buildings and housing must follow the new way of living. This becomes real only if the buildings can also transform by use of kinetic architecture, without losing though the fundamental concerns of traditional architecture.

If kinetic architecture is born out of transformation and change, it is only natural that its way of “visualization” should respond to the same basics: the supporting software can effectively present a kinetic structure by following the “kinesis” – movement- of such structure. In this case, simulation would be completed. Its data would comprise of a library of the existing prefabricated elements that affect the environmental conditions and therefore any tangible result of them, meaning a construction. This way, the adjuvant software will become a tool for the architect, who will be allowed to a preview, to a closer and more detailed view and to an experimental “in vitro” test of the architectural creation and its potentials, feasibility and realization.

1.2 Background

A closer view through the bibliography and internet into what kinetic architecture is and what it includes, has given the chance to form a general overview over this area of research but also to make some observations. It become easily noticeable that even though many books and internet sites refer to this kind of architecture, by giving examples of buildings and constructions, however there is no documentation of the rules that this architecture follows during its creation.

Many books of special editions presenting the modern structures or referring entirely to this kinetic architecture, offer a wide source of information over the way of construction, the way of moving and even the way of implementation over the users or the inhabitants. But there is no record of the design rules that the architects used through the conception and realization of the structure, or generally there is no information, which can clearly state the rhythm that this architecture follows and by which it can be described. It is undoubted that these constructions and this kind of architecture corresponds to a different, unique field, which, as every architectural rhythm such as Bauhaus or Minimalization, needs to have its own features and identification.

The only associated book found, which clearly describes the genesis of kinetic architecture and its different applications, it is written in 1970, looking in architecture future way in front of its time, by William Zuk and Roger H.Clark, titled “Kinetic architecture”. This book is, as described by the writer, “a compilation of existing pertinent material on adaptable architecture furthered by some new ideas for the future. The concepts discussed in the book are evolutionary and are based upon reasonable predictions of trends.” (William Zuk) However, most of the things described in the book as future ideas have become true.

Other similar information are given by Kinetic Design Group and Michael A.Fox from MIT University, accompanying different projects that Kinetic Design has introduced such as “Responsive Skylights”, “Folding Egg” etc **Figure 1-1**, all relevant with different types of kinetic architecture.



Figure 1-1. Project “Responsive Skylight”

The information given regards the discrimination of kinetic architecture in three typologies, the definition of what Intelligent Kinetic Systems are and how they are converged and finally which are the control mechanisms in intelligent environments.

However, more relevant information about the area of kinetic architecture that this thesis is interested in, could not be found. Even researches that have been performed do not seem to be equivalent with the attempt to combine the kinetic architecture with the animation software. Most of them relate to one of the two subjects and not to the combination. The only thesis that seems to be in the same line with the proposed theme, is the one presented by Javier Garcia De Jalon and Eduardo Bayo for the University of Navarra and CEIT in Spain, titled “Kinematic and dynamic simulation of Multibody Systems” but where the name “multibody” stands as a general term that encompasses a wide range of systems such as: mechanisms, automobiles and trucks, robots, etc. Therefore, though the general outline seems to be similar with the proposed one, the specific area of kinetism used for this thesis is completely different.

Therefore, searching under the subject of kinetic architecture has shown that no relevant researches have been performed with the proposed theme. The same results have been occurred also from reviewing the state of art of the animation software. No design tool or generally an animation software which could support the formation or study of this specific architecture has been found. Those that exist are either relevant with the design and simulation of machines or with the animation of characters, such as Maya, 3dMax, etc, which are the most well known and frequently used, even in architecture.

Though a representation of a kinetic structure can be implied with the character animation software sources that already exists, different functions in them seem to be missing, those that will provide the user of the programs, always to the direction of kinetics, with a real model of the construction. The software that exists today has the ability of representing a structure and its kinetic movement, by creating an animation. However, this animation does not correspond to reality or cannot give realistic results when other parameters such as forces, or material properties, have to interfere with the construction. What the animation can do is just give an overall representation of the kinetic movement, but when a simulation for this field of construction is needed, the software seems to be missing functions.

Maybe the lack of definition of design or conception of procedure, as aforementioned, can be the reason why such a tool or a software has not been implemented. Since there is no evidence of which rules exactly this architecture follows, there can be no easy recognition of the needs that it has over an animation software. In modern technology, where the standardization

for avoiding time consuming has appeared in every aspect of our daily life, even in architecture, the use of computer and specific software is considered as prerequisite in the overall procedure of designing and actualization of kinetic construction. Nevertheless, all these indicate that a new area for research exists, the one that this thesis is concentrating on.

1.3 Classification of kinetic structures

1.3.1 Typology of kinetic structures

As mentioned earlier, kinetic architecture is a wide field that can include and refer to many subjects. The proposed subject of this thesis focuses on one particular category of kinetic architecture. This category involves a specific type of kinetic architecture, related to the physical movement of structural building elements that can result to the spatial movement of a structure as an entirety or just part of it. More particularly, this kind of architecture can be defined as: “Buildings and/or building components with variable mobility, location and/or geometry. Structural solutions must be considered in parallel both the *ways* and *means* for kinetic operability. The *ways* in which a kinetic structural solution performs may include among others, folding, sliding, expanding, and transforming in both size and shape. *The means* by which a kinetic structural solution performs may be, among others, pneumatic, chemical, magnetic, natural or mechanical.” (Michael A. Fox)

Summarizing the above definition in two single words, the main concerning issues of this thesis are simply presented: “folding” and “mechanical”. Our interest is drawn to structures which “fold” by means of different structural elements moving with the help of mechanisms. Buildings like the one mentioned before (Planetarium) and other as Milwaukee Art Museum show this kind of mobility. **Figure 1-2, 1-3, 1-4, 1-5.** Despite the different kind of movement, the Planetarium by bending its articulated elements around specific axis to imitate the move of the eyelid and to accommodate entrance-exit and Milwaukee Art Museum by moving its elements to produce different shadowing effects, they both follow the same functional rules.

A better explanation can be driven by examining the categories into which kinetic structures can be divided. Generally, kinetic structures in architecture can be classified into three general categorical areas:

Embedded Kinetic Structures

Embedded Kinetic Structures are systems that exist within a larger architectural whole in a fixed location. The primary function is to control the larger architectural system or building, in response to changing factors.

Deployable Kinetic Structures

Deployable Kinetic structures typically exist in a temporary location and are easily transportable. Such systems possess the inherent capability to be constructed and deconstructed in reverse.

Dynamic Kinetic Structures

Dynamic kinetic structures also exist within a larger architectural whole but act independently with respect to control of the larger context. Such can be subcategorized as Mobile, Transformable and Incremental kinetic systems.

Out of the types of kinetic architecture mentioned above, the embedded structures are those that are going to be studied in this paper.



Figure 1-2. Milwaukee Art Museum (A. Tzonis)

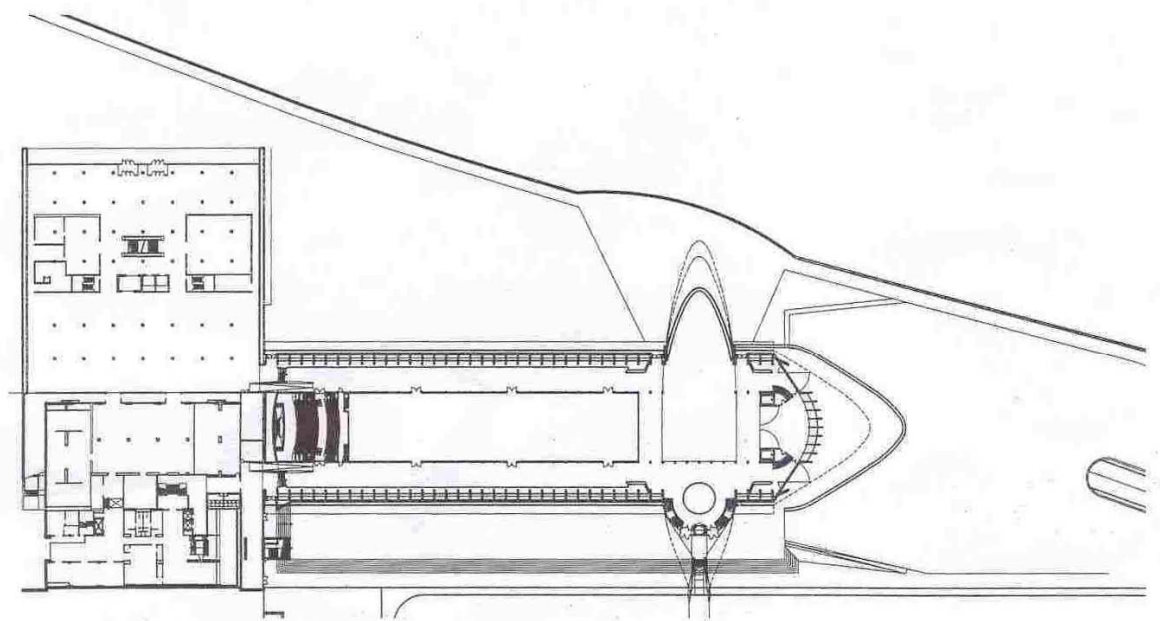


Figure 1-3. Drawing of Milwaukee Art Museum (A. Tzonis)



Figure 1-4. Planetarium in Valencia (A. Tzonis)

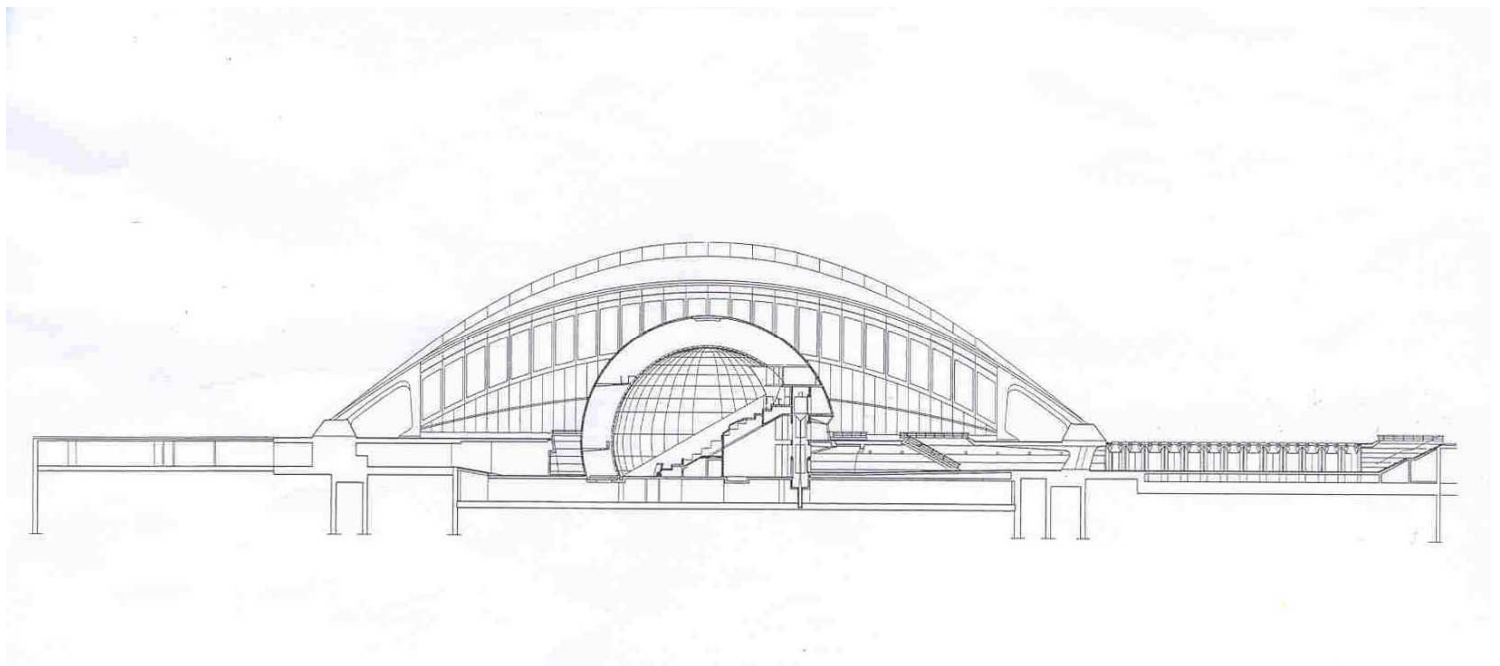


Figure 1-5. Drawing of the Planetarium in Valencia (A. Tzonis)

1.3.2 Examples of kinetic structures

A number of constructions has been chosen as a representative sample of the architectural kinetism. These structures are conceptions that have been realised throughout the years and which, with the mobility that they prove, can be included into the architectural part that the thesis is researching. The movement they show appears to a part of their structure and not their entirety and it serves different purposes like opening and closing entrances or opening and closing dome. In order these structures to be evaluated, certain criteria need to exist, based on which the category on the constructions in different sections will be performed.

In this case, the criteria for this categorisation can be the motion mechanisms of the smallest in number set of elements that forms a kinetic part. The structures are decomposed to the smallest parts from which they have been formed, like beams, columns etc. However, in this case, the composition does not reach the primitive elements of a simple construction but it stops to the smallest individual parts that can form, when multiplied, the kinetic part of the structure. This means that by using the smallest number needed of primitive structural elements and by preserving their relationships over the kinetic movement, the result is the primitive set of elements needed for the description of this kind of structures.

The result of this decomposition was five basic motion mechanisms which can be generalised to refer, correspond and include all the kinetic constructions. The results can be seen in **Table 1-1**, where the relationship between the buildings and their motion mechanisms is being shown. The first mechanism is been formed by two elements and an articulation between them, with the first element fixed in a certain position but the possibility of rotating and the second moving along a specific linear path. The second case looks like the first. However the first element is stably fixed in a position and the second element is rotating along a hinge not at the end of the elements body where there can be a joint, like the fourth case, but inside a point of the second elements body. The third is a mix of hinge and joints and the fifth is a simpler form of the third including just the joints. The first four introduce one degree of freedom, whereas the last one shows two degrees of freedom. Cases and construction which use mechanisms with three degrees of freedom are not included to this research, as it is even more complicated this kind of mechanisms to be animated, where already those with one or two degrees of freedom seem to face obstacles in their representation.

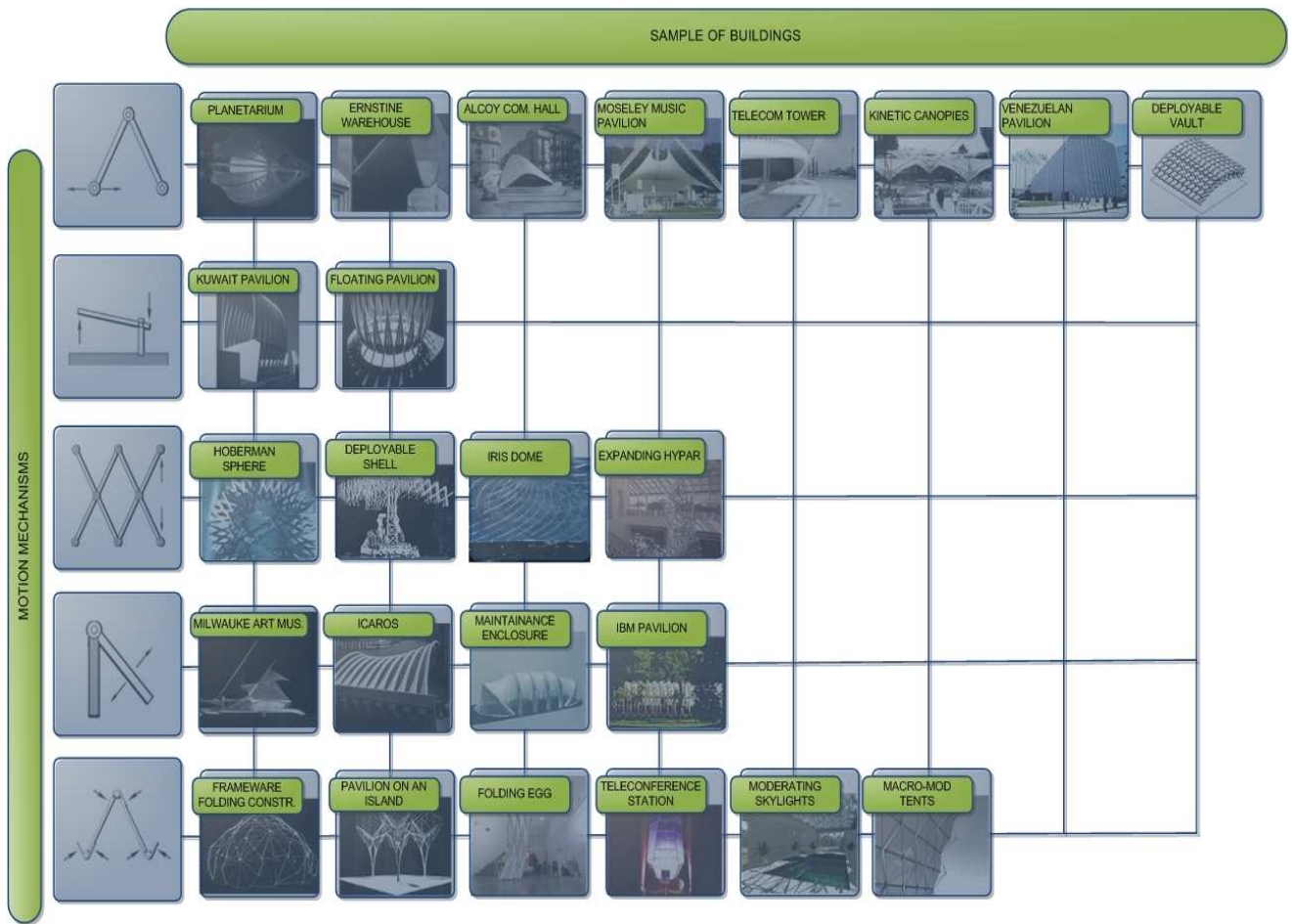


Table 1-1. Buildings categorisation to motion mechanisms

2 METHODOLOGY

Knowing already from the literature and internet research that there doesn't exist an animation software dedicated to kinetic structures, the research will be targeted and based on the use and the interpretation of an already existing software into an appropriate application for the animation of kinetic structures. In detail, the steps to be followed for the realisation of the thesis are:

- [1] Evaluation of the existing animation software
- [2] Selection of an appropriate application for the conduction of the thesis based on criteria
- [3] Usage of the selected software to model several case studies taken as representatives for the different types of kinetic structures
- [4] Identification of the limitations of the software for kinetic structure design support
- [5] Possible opportunities for improvement of the software with the usage of scripting language

A comparison of animation and 3D prototyping programs will be performed, in order the most appropriate software to be chosen for the accomplishment of the research. The criteria for the choice of the software will be their ability to customise, to produce geometric modelling and animation regarding time (video animation). Of course, in order to understand better the different parameters that are needed for the evaluation, it is necessary first a correspondence between the language already used by the software and the actual construction and elements terminology to be performed. By realizing how the parts of a structure can be interpreted into the software, it will be easier to recognise the functions and the different structural parts that are missing but they are necessary for the representation of the kinetic structures.

After the choice of the program, different case studies will be assigned into the application in order to recognize in real time its possibilities and weaknesses. The interaction with the program over the delineament of existing kinetic buildings will stretch out the lacks and the helpful commands that are included in the program regarding the representation of kinetic architecture. The case studies will be chosen according to their ability to represent a mass of equivalent structures, but also based on the presentation of an increasing difficulty towards the steps of the creation of the structures and its animation.

Lastly, after summarizing and categorisng the results from the case studies, with the help of the computer and the scripting language, an attempt and examples will be given towards the creation and overcome of the malfunctions. Future possibilities and usage of the program will also be described.

3 EVALUATION OF EXISTING ANIMATION SOFTWARE

3.1 Concept

In order to be able to evaluate and choose the required software for the conduction of this thesis, it is appropriate to perform a research for the kind of existing software in the area of architectural design and animation nowadays, their special features and their ability in a structure animation. As aforementioned, big variety of programs exists today in the market, which can relatively easily animate a kinetic structure. A research on internet or in proper books can provide the reader with necessary information regarding the special features for each software, as well as with comparison tables. However, the common characteristic of the existing software is that they are basically created for the performance of character animation and not architectural constructions. Therefore, in order to create the kinetic structure animation into this software, the purposes for which the functions are used in the character animation must be interpreted in correspondence to serve the animation of structure.

However, the software existing today show between them a common way of operating, whether speaking for character or structure animation. Usually, the basic steps for creating an animation are the same. The design process, the creation of the necessary *joins* and *articulations* together with the *bones* to create a *skeleton* and lastly, the animation are briefly some of these common operations. All the software follow approximately the same procedure, differing in the terms that they use to define the different elements and functions. Of course, differences seem to result as well, despite the similarities between them, regarding the existence or lack of certain functions. Some demonstrate weaknesses in the design process, an important factor when an architectural structure is been modelled, others seem to be lacking of important basic functions such as copying and pasting.

In order to understand the differences between character and structure's animation, it would be useful to mention the meaning of different terms. By the term "character animation" it is meant the animation which is implied on objects or models designed to resemble as creatures, such as humans, monsters, animals etc, resulting with their 3d natural or a human like representation of movement. These models are named in the software language as "characters" and can be described as *skeleton* construction, with *bones* and *joints*, which can be outer covered by a mesh, named as "*skin*". Basically this whole structure has been named as character for it resembles and follows the system that a human body is created. More particularly, "skeletons" in the software are "hierarchical, articulated structures that let you pose and animate bound models. A skeleton provides a deformable model with the same underlying structure as the human skeleton gives the human body. Just like in the human body, the location of joints and the number of joints you add to a skeleton determine how the skeleton's bound model or 'body' moves. When you bind a model to a skeleton, it is called skinning" (Maya User Help). Most commonly all these models are used in animation movies or in video games where different creatures are needed.

Basically, all the terms used into the software language for character animation, correspond exactly to their real meaning in a human body.

Therefore the joints are “the building blocks of skeletons and their points of articulation. Articulations are the state of a skeleton’s joints, including position, rotation, and scaling” (Maya User Help). Joints are behaving in the software program exactly as a wrist or the elbow of a human body, it can have different degrees of freedom in the movement, according to the rotation constrictions that can be given, for the achievement of the desirable pose. Joints let you transform a skeleton when posing and animating a model “Each joint can have one or more bone attached to it, and more than one child joint. Bones are only visual cues that illustrate the relationships between joints” (Maya User Help). When speaking about children joint, the term refers to the hierarchy of the structure depending on the bones, where the first bone is the parent and the following is the child. The joints are always attached to the ends of the bones and they are virtually represented. A bone can have maximum 2 joints at its ends as shown in **Figure 3-1**. Therefore, the child joint is connected as a meaning with the child bone. The relationship between the parent and the children can be one to one or one to many.

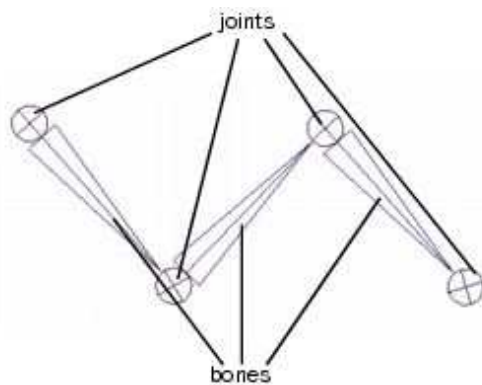


Figure 3-1. The skeleton structure showing the bones and joints system (Maya Help)

Having in mind everything mentioned before, it is easily understandable that the comparison of the software depends on their ability to relate different functions with various animating purposes. The adaptability of the software to fulfil a number of demands over animating will be the criteria of selection. The more capable the program will be able, while designed to create animation of characters, to adjust the meaning of its function to serve the animating needs of a kinetic structure, the more appropriate will be among the other software. Of course, that depends on different factors that should be investigated while the evaluation. However, first of all, it is important to give the correspondence between the software and the structural language for kinetic architecture. Knowing the relationships among these parts, it is possible to understand in further detail the specific features that the software should prove in order the desirable research to be performed. The right choice of the program not only will provide with best possible results for this thesis but will also facilitate the interaction and make possible the further development of the software.

3.2 Modelling kinetic structures in animation applications

By using and interacting with the existing animating program, it is easily

understandable that the applications already form a link between nature’s terms such as “skin”, “bones” etc, with software’s terms. By importing in their language exactly the same names of different elements or terms existing in nature to correspond to elements or functions created into their environment, they help the user understand the way of the function operation and make easier the interaction. Since these software refer to character and not to structure animation, it would be useful for the thesis to create a further connection between the terms of these software to construction elements.

The **Table 3-1** shows this relationship between structural elements and software elements. As proved from the case studies, the implementation of the animation was not always done in a direct way. Though it would be expected the elements of the structure, their connections and their parameters to be sufficient for the creation of the movement of the kinetic construction, in some cases that was not enough. Other methods of implying the animation were used, those that were providing with more liberties and possibilities that the typical way could. The meaning of usage of the skeleton with the skin isn’t directly obvious into a structure from beams and columns, for example. However, this indirect way, by implying a skeleton inside the skin, which in this case resembles the columns and the beams, gives more possibilities of creating the representation of the movement of the kinetic structure. This, however, will be easier to understand into the presentation of the case studies. The analysis and the correspondence of the elements from physical to virtual, has been performed by decomposing different kinetic structures to their primitive elements.

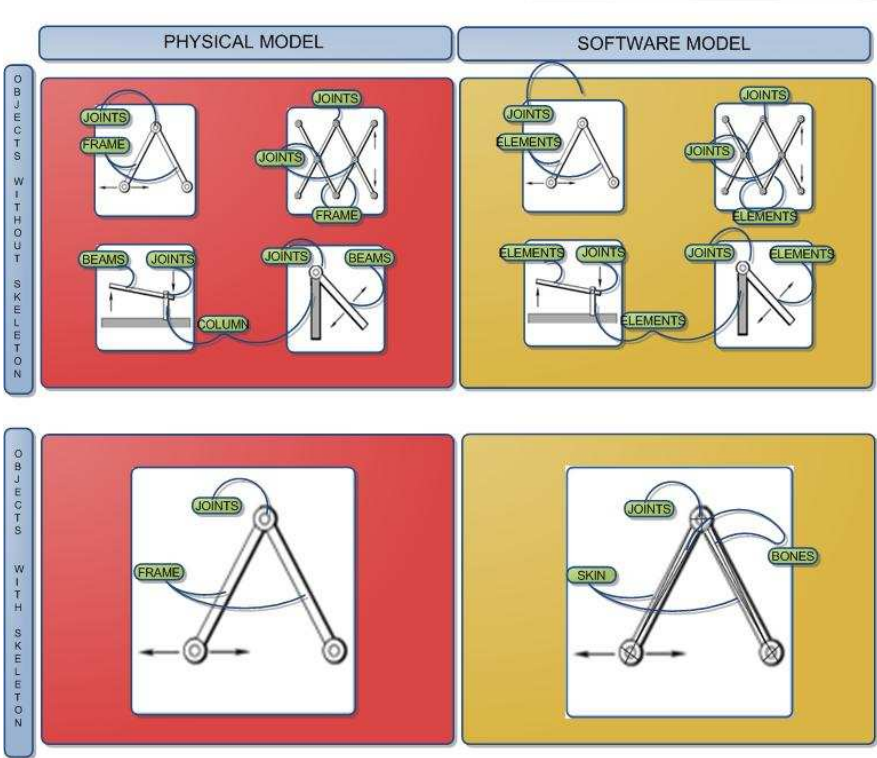


Table 3-1. Correspondence between mechanical model and model implemented into an animation software

3.3 Evaluation criteria

By recognizing the basic structural parts required for the creation of a kinetic structure and by investigating their cooperation between them in terms of movement, it is easier not only to choose the functions and different parameters inside the software that are useful for the research but also generally to form a list of criteria on which the software can be evaluated. These criteria will provide, at the end, with an objective aspect of the most suitable software for this thesis, by estimating each one of the 3d design software by themselves, but also in comparison to one another. The criteria regard the software's necessities for each level that will be investigated in the thesis, but also the general basic knowledge of the required software operations and architectural design procedure. By classifying these criteria, we could say that these are the following:

- Possibility of creating proportional design
- Modification of spatial arrangement
- Possibility of creating parametric elements
- Elements/Objects Definition
- Joints Definition
- Control time function
- Existence of scripting language
- Types of import /export files

The first three criteria refer to the design process into a 3d modelling software. As the field of research regards to the architectural design, the existence of functions for creating a real metric representation of a building or a structure is absolutely necessary. The approximate delineation of an architectural drawing would not respond to the architects original creation idea and the software would not provide after all to the user realistic results. Under the same sense, for the construction of a 3d drawing or better a structure, the possibility of the modification of the spatial arrangement of its elements is important. When the creation of the same elements in certain distances is needed, something usual in architectural design, the existence of functions working towards that direction is required. In the software language, it can be executed with a combination of different functions such as "copy", "paste", "offset", and "mirror" and therefore their constitution is appropriate. Regarding the parametric elements, the ability of changing the geometric forms of the objects is easily understandable, as for each structure and regarding its original drawing, elements of different proportions and use are going to exist.

In some cases, there might be a need of importing into the modified software that the whole thesis is going to be formed, files or better drawings and 3d objects that have been designed in other programs. For this reason, the possibility of importing different kind of files in the software is necessary. The more files the chosen program supports, the best will be the cooperation with the other software. At the same time, while importing the files, it is necessary that the program will have the ability to recognize, accept and define the objects or drawings imported in its own environment, and, for this reason, the existence of elements definition is required.

A very important factor is the ability of the software to define the joints and

the connections between the objects. As we are talking about kinetic architecture, movement will be implied in the elements of the construction. The joints are the key to the movement, as they are the ones who can constrict or permit the direction of orientation of every element. Therefore, the importance of defining their rotation angle, their degrees of freedom but also their constraints is easily understandable.

Furthermore, through the animation or simulation of this movement described as above, the connection of every pose of the object with a certain chronicle moments is essential. During the study of the movement behaviour of the construction, which will be conducted with the help of the software animation, it is useful to know the exact chronicle moments in which the result of the animated movement of the object is not the desired one. By this way, we can easily afterwards alter the behaviour of the structure for this certain moment.

Lastly, the possibility of using a scripting language in the software, is considered as a required factor, only by this way it will be possible to modify the chosen software and create new functions that there might be needed.

3.4 Comparison and choice of the software

Searching throughout internet, a lot of information and comparisons for 3d design software can be found. Some of these software seem to be more “famous” in the area of 3d representation and animation, as their names frequently appear while searching. At the same time, in architectural circles, some of these seem to be appropriate and known for architectural 3d design. Having in mind those most well know programs involving animation and 3d architectural design, but also the necessity of fulfilment of important factors that has been aforementioned, a selection of 7 programs has been performed. These programs are the following:

- 3DS Max
- Blender
- Cinema 4D
- Houdini
- LightWave 3D
- Maya
- XSI

A first approach, with the help of information found over internet regarding the qualifications of the software, showed that, in fact, the differences between them focus more on the scripting language and the types of files that can be imported or exported. This can be easily understood with a quick overview on the following **Table 3-2**. In this table, some of the main features are included, whose existence inside the programs possibilities is necessary for the choice of the software. In particular, regarding the animation and simulation ability of the software, the creation of objects and characters is thought to be the most important.

By using the term “*objects*”, we refer to the *primitive objects*, which can be modified and constitute the separate basic parts of the whole construction, whereas with the term “*characters*”, we refer to the number of objects, articulated between them, which form a whole structure. Therefore the use of the terms is been done with a respect to the creation of a kinetic structure inside the software’s environment and not its original terminology

corresponding to character animation. With the term “physics”, we refer to the number of powers, such as gravity, that the program proves as functions and that can be implied to the character or the object in order through the animation procedure to have a more natural result. The types of the importing and exporting files refer to the extensions of the files which include 3d objects.

It is easily understandable that the information provided through the table as a first approach, are not sufficient to form a good aspect over the abilities of the software. All the programs seem to have the same capabilities with almost no significant differences. Therefore, the programs can only be better evaluated when seen more closely and in detail towards which are their specific differences and what uniquely is been offered by each one of them.

With a more in-depth research it has become evident that some of them seem to be more appropriate for rendering purposes, like Cinema 4d, whereas others seemed to have more abilities into animating like Maya and 3DMax. But still, whether a program fulfils the aforementioned criteria, it is difficult to be estimated. The information that

COMPARISON OF ANIMATION SOFTWARE								
Package	Animation and Simulation		Extension Language	Import/Export type of Files				
	Object	Character		DWG	OBJ	3DS	DXF	FBX
3ds Max	Yes	Yes	MAXScript	Yes	Yes	Yes	Yes	Yes
Blender	Yes	Yes	C, Python	No	Yes	Yes	Yes	No
Cinema4D	Yes	Yes	COFFEE	Via Plugin	Yes	Yes	Yes	Yes
Houdini	Yes	Yes	HScript, VEX, Python	No	?	?	?	?
LightWave 3D	Yes	Yes	LScript	No	Yes	Yes	Yes	Via Plugin
Maya	Yes	Yes	MEL	Via Plugin	Yes	Via Plugin	Yes	Yes
XSI	Yes	Yes	VB Script, JScript, Python	Via Plugin	Yes	Via Plugin	Via Plugin	Via Plugin

Table 3-2. Comparison of Design Software

is given for every program does not have the same quality and quantity as it derives from various sources. In cases where specific details needed to be found, such as whether it is possible to create objects with certain parameters

like width or length which can be later modified or whether it is possible to use the functions “copy”, “paste” or “mirror”, they can be found out only by interacting with the programs itself.

However, with the information found in the previous steps, the number of software has been limited into two, Maya and 3DMax. These two seem to provide with more possibilities over the animation part, to be the most well-known and used software, with a wide field of tutorials and user instructions. More information about the two programs but also for the rest can be found in Appendix that presents in detail each program and the scripting language of use.

In order to underline which of the two programs is more qualified for the conduction of the thesis, a creation of a simple kinetic structure into both software has been performed. Already since the beginning process of designing the structure Maya has shown to be more complicated. Though it has more capabilities than 3DMax, this abundance of functions seems to be working in the contrary way than expected. For a new user, the environment and the documentation in the user help is difficult to understand and handle and functions seem to be differently organized in comparison to usual programs. Therefore, for mirroring a copy or changing the proportions of an object, the tabs and the appropriate panels seem to be difficult to find. Moreover, Maya though it supports the .dxf file, it doesn't do the same for a .dwg, an important factor when someone wants to import a drawing. On the other hand, 3DMax has the possibility to import a .dwg file and it is easier to use. Therefore, 3DMax is chosen as the more appropriate software that would be used for the study of kinetic structures.

4 CASE STUDIES

4.1 Approach

As mentioned in the methodology, in order to recognize the needs, the possibilities and the lacks of the software to create a simulation of a kinetic structure, the conduction of a number of cases studies is needed. These case studies will relate to the creation of different already existing kinetic structures and their animation inside the software. Through them, concerning different kinetic structures with various levels of complexity, the existing problems between the animation of kinetics and the modern software will be stretched out and a wide number of these analogical constructions will be covered, giving more validate results for the thesis.

By the matrix presented in the chapter 1.3 “Clarification of kinetic structures”, three constructions have been selected to form an equal number of case studies, the Kuwait Pavilion, the Alcoy Community Center and the Folding Egg. The mechanisms and, therefore, the buildings, presented in the matrix seem to have an increasing difficulty concerning the creation of the movement animation into the software. The more elements and freedom in movement that the kinetic elements seem to have, the more complex and difficult is their representation over the software. The case studies appear to have, as expected, an increasing level of complexity, which corresponds to the order of their study. Therefore, the Kuwait Pavilion shows less complexity than the Alcoy Center and the Folding Egg is thought to be the most difficult of all. The three case studies include also the other two mechanisms that are not going to be studied individually, as the Kuwait Pavilion already covers the second mechanism and the Folding Egg the third.

4.2 Generalization of steps needed for the creation of a kinetic structure

All animation software follow the same procedure for the creation of a kinetic representation, as aforementioned, either this is a character or a structure animation. Therefore, it would be wise to introduce this general way of creating the different structures into an animation software and implying on them the necessary functions for the simulation of their movement, before describing each case study individually. This overall procedure can be graphically described as shown in **Table 4 -1**.

As shown by the graph, the whole process can be divided into three major categories, the setup environment, the definition of the elements and the animation. The first part covers almost 20% of the overall time, the second, which is the most time consuming, 70%, whereas the third part, though expected to be of longer duration, only the 10%. Of course, according to the model and its demands these percentages may differ. However, in a general approximation it can be said that typically the time for the definition of the structures and their animation follow the given graph.

The whole approach starts with the question whether there exists an electronic form of the 3d model of the structure. If it does, the file should be imported into the application in which the project will take part and eventually all the objects that form the structure should be recognized into the new software. By “recognizing”, it is implied that the program should be in

position to “understand” the different objects as if they were created in its environment, providing them with all the possibilities of modification and functioning. That way, the application will be able to manipulate, during the development of the project, the elements and the objects of the construction imported. If there doesn’t exist the 3d model, then the procedure should start from the very beginning, by designing all the appropriate elements based on a drawing under scale which could be imported as a background and used as a template.

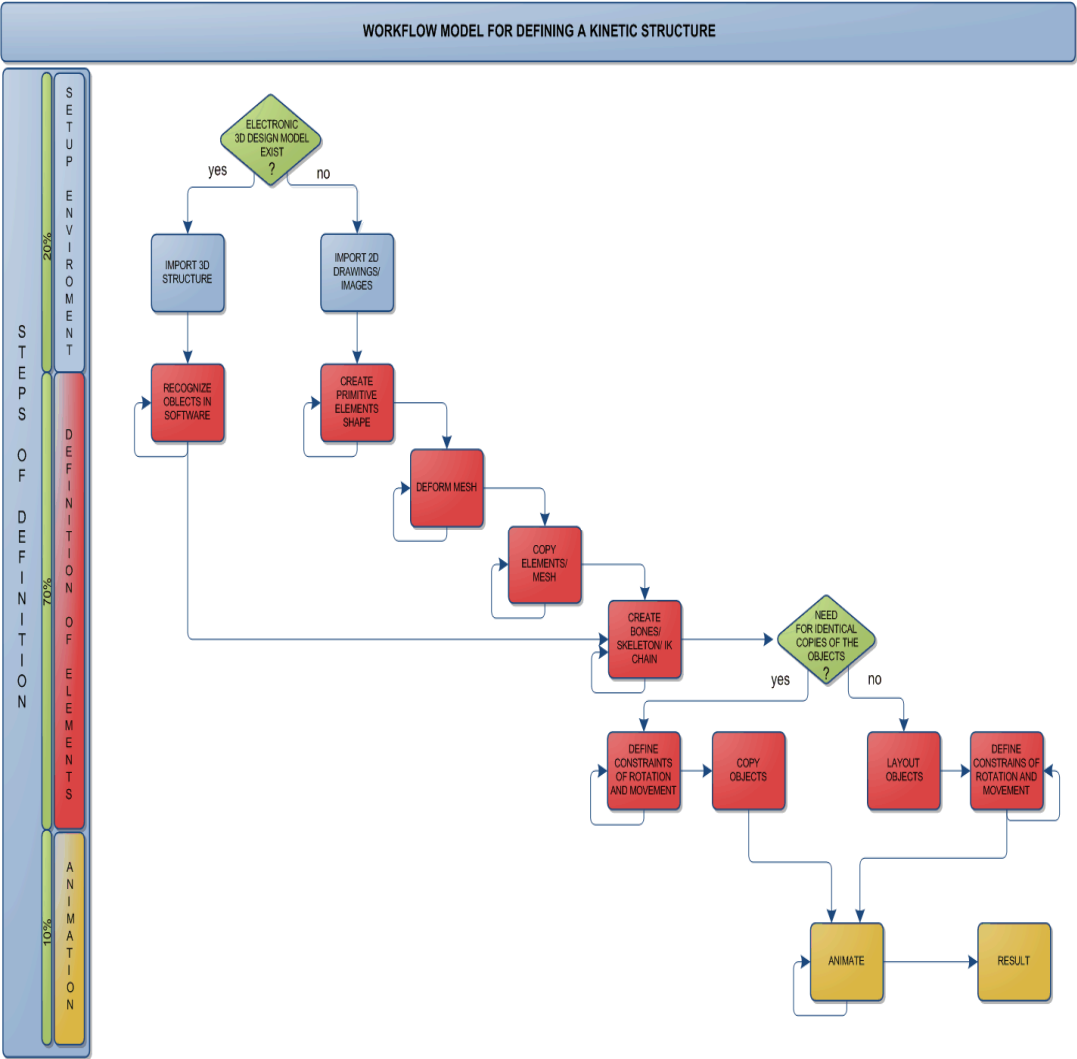


Table 4-1. Graphic representation of the definition of a kinetic structure into an animation software

Firstly, the elements that construct the objects of the structure (as defined as above and shown in **Figure 4-1**) should be created, but in a draft way, using an appropriate approximate primitive shape like a sphere, sphere, pyramid, etc. which can contain the overall desirable shape of the element. **Figure 4-2.** This basic shape is the one whose mesh will be deformed in order to give the shape of the element of the structure **Figure 4-3**. It should be mentioned that for structures that consist of many identical or analogical proportional objects, the elements and therefore the basic object should be created once, as the special arrangement and the coping for the overall definition of the structure will be performed after the definition of its movement. In cases where the elements of the basic object are identical can be created by coping the first element after its mesh deformation, whereas for cases like the example shown in **Figure 4-1**, the elements are not identical and should be created separately.

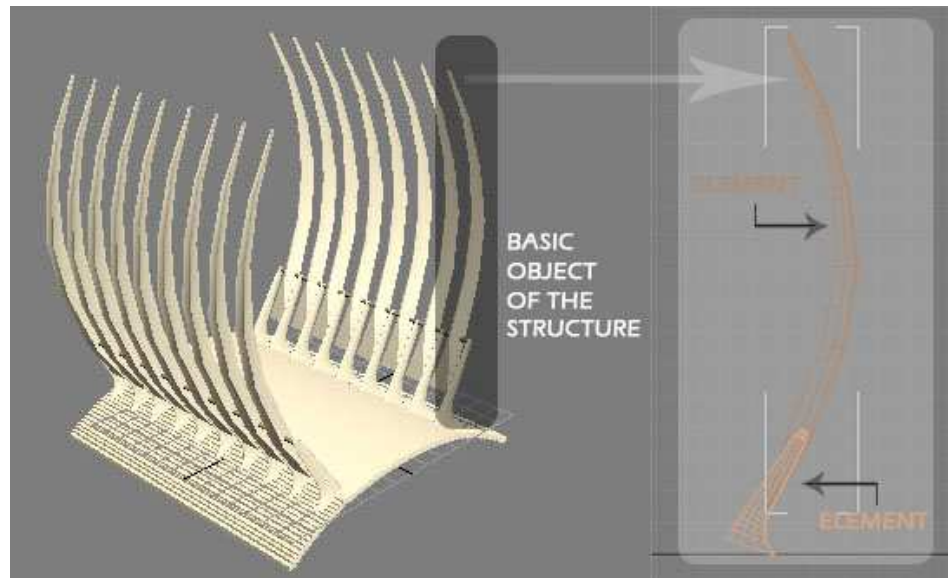


Figure 4-1. Elements and basic objects of structure.

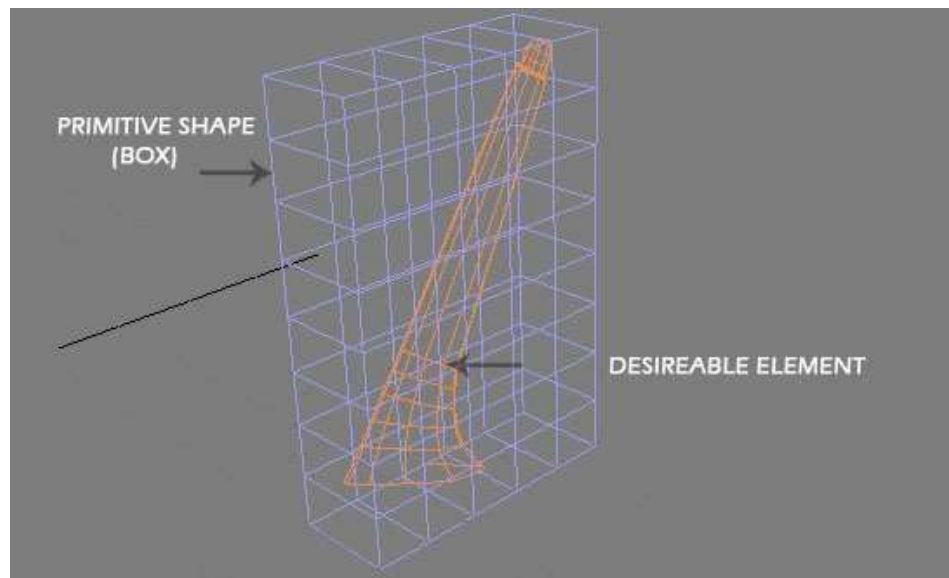


Figure 4-2. Primitive shape for an element's creation

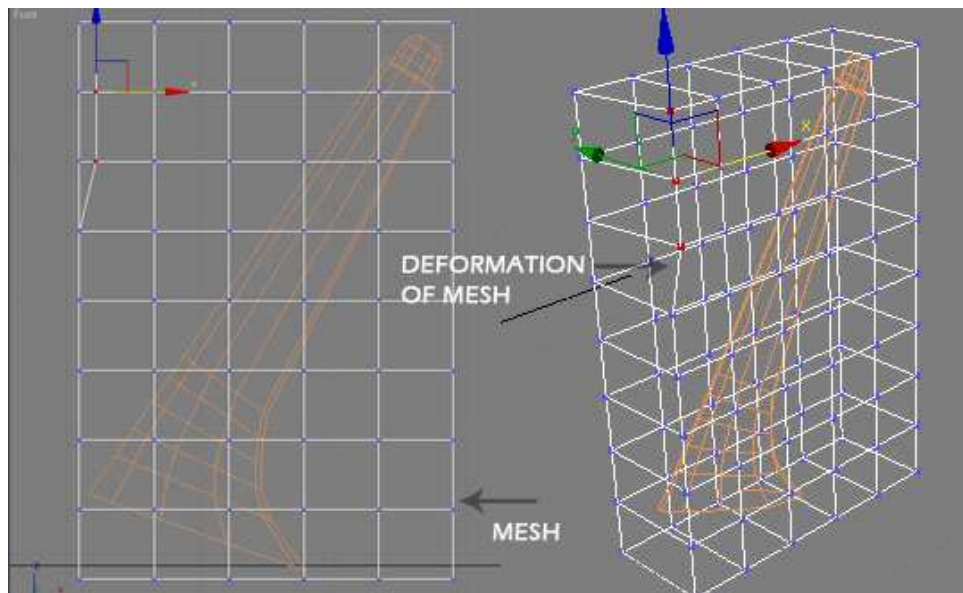


Figure 4-3. Deformation of the primitive element

At this point, the former existing 3d model which is imported into the software meets with the model created into the working application. Henceforth, the steps are the same for both cases. After the final set of the basic object's shape, the bones and the skeleton must be created. This can be done either by setting the deformed meshes as bones and giving the hierarchical relationships between the elements or by using the functions and setting virtual bones inside the structure, identifying, at the same time, that the deformed mesh is actually the skin of the structure. **Figure 4-4.** In the second case, an interpretation of the character design has been performed to adjust with the needs of the structure. Which of the two methods is going to be used during the creation of a construction into the software depends on the complication and the needs of function for the implementation of movement in each case.

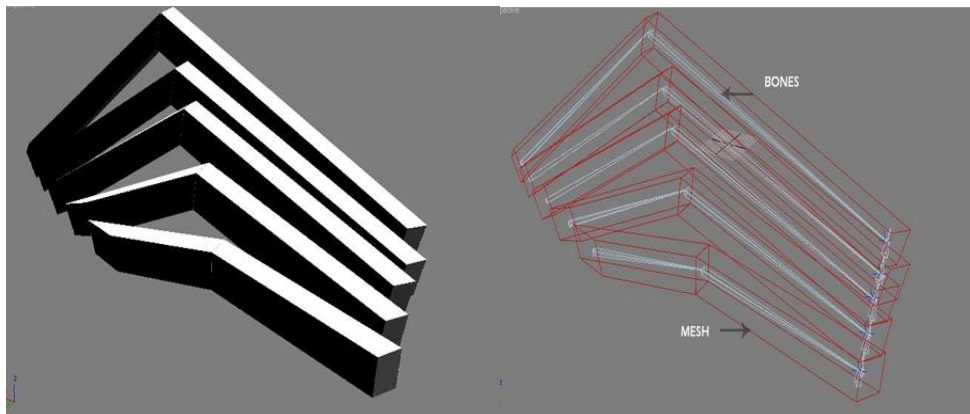


Figure 4-4. The bone structure and skin(mesh) into an kinetic construction

For one more time, a distinction should be performed on whether there is a need for identical copies of the basic object or not. Depending once more on the structure, there are two ways to finally set the special arrangement of the whole construction. **Figure 4-5.** If there is need for identical objects, then first the constrains of the movement are being implied on the basic object and

then this is being multiplied to give the complete structure. This is done because once the constraints are set, there is no possibility of modification of the objects and their elements. In any other case where the object should be even resized, first the spatial arrangement should be done and then the constraints, individually in every object, should be set.

Lastly, after the settlement of the whole structure, the animation parameters have to be adjusted for the representation of the kinetic movement. Depending on the automations that have been created with the help of different functions for the movement of the structure, the animation can be easily or difficultly achieved. By setting, for example,

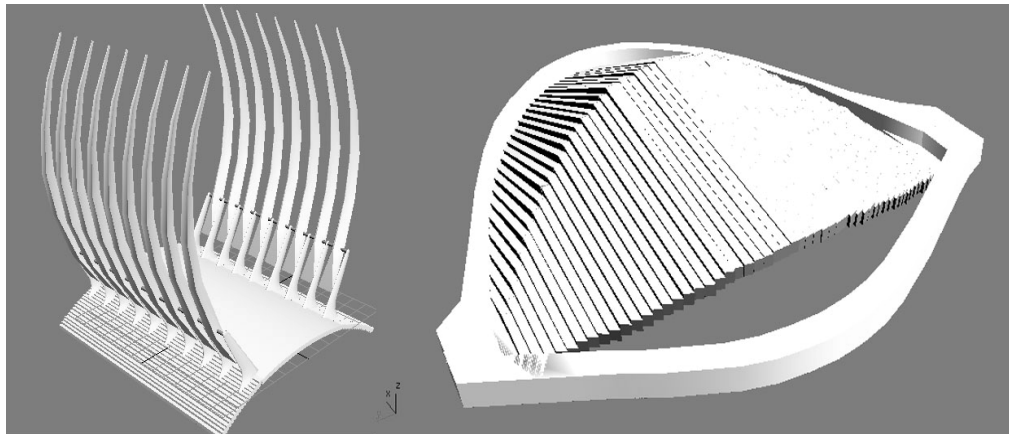


Figure 4-5. a) Structure with identical objects, b) structure with different proportional objects

the motion of the structure to be linked and follow the position of one single object, it is easy to move the actual construction just by moving this object into eligible locations and by setting in the timeline, keyframes to represent these locations. In other cases, where no automation exists, all the objects should be arranged individually into different locations and poses for every different keyframe that will be created, in such a way so that the whole movement will have a logical progress.

In order to be understandable, the way of creating the animation appears similar with the different frames in a film of a movie. However, in this case, the frames of the animation are been created by the user, by giving into fixed chronicle times the position of the objects of the structure. The software itself creates afterwards the middle steps of the movement, giving as a result a complete movie of the whole kinesis.

4.3 Case study 1: Kuwait Pavilion

4.3.1 Introduction

Kuwait Pavilion was built in 1991-1992 and it is a symbolic structure which was commissioned at the behest of the emirate for the 1992 World's Fair in Seville, Spain. The roof structure is made up by mobile elements which come together as tapered fingers of two hands, or better in this case, as the leaves of two palm branches. Seventeen 25 meter long finger elements, constructed in timber and supported by hydraulically operated reinforced-concrete columns, interlock to form an impenetrable vault and then unfold to reveal the gentle barrel-shape of the 525 square meter piazza. The piazza is glazed with panels of laminated structural glass and translucent marble that illuminate the gallery bellow during the day and grow with an enigmatic inner light at night when uncovered. When half closed, the roof casts light and shadow on the piazza bellow, providing protection from the sun, like a palm shaded oasis. (A. Tzonis)



Figure 4-6.
Kuwait pavilion
(A. Tzonis)

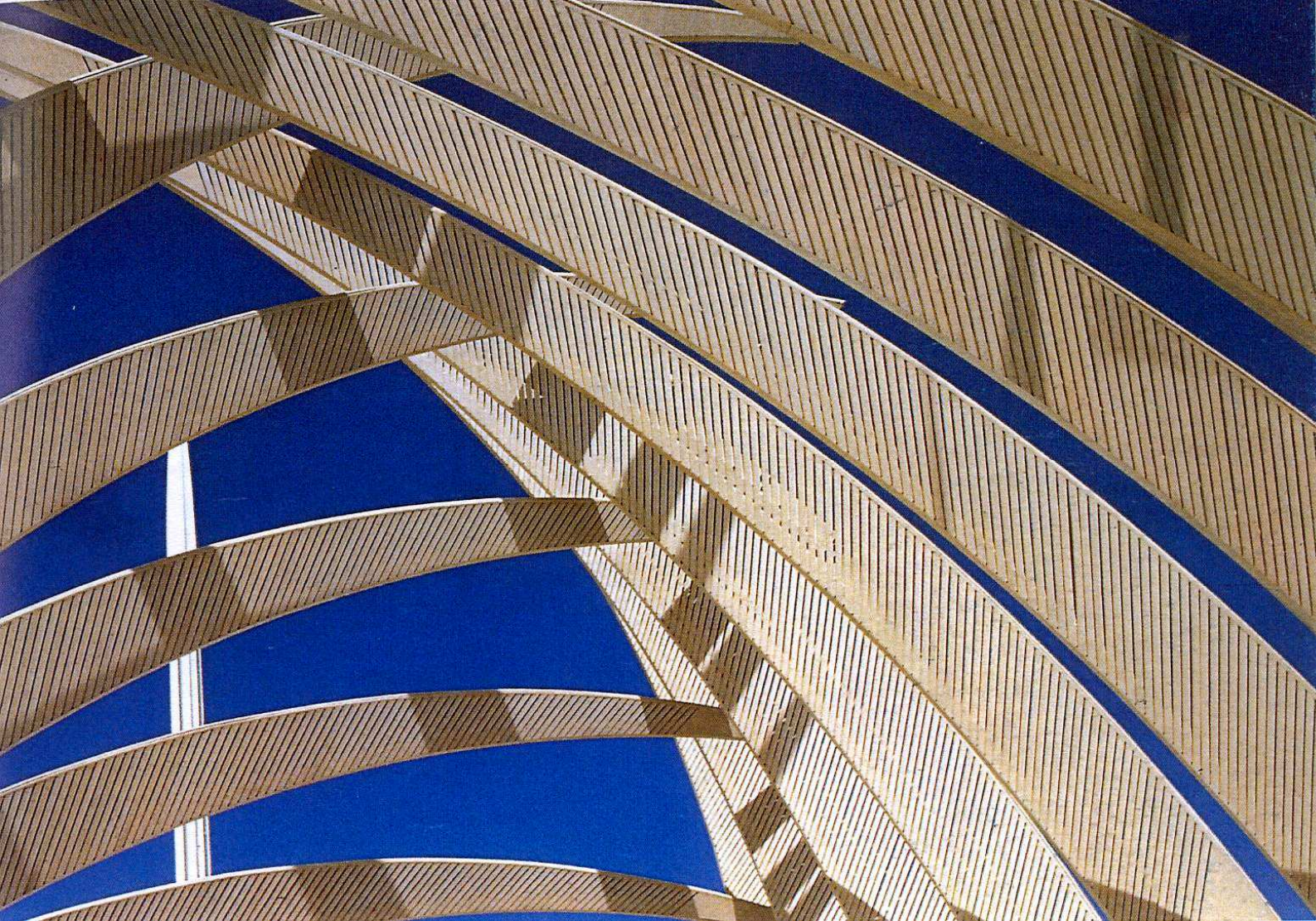


Figure 4-7. Detail of the “fingers”. (A. Tzonis)

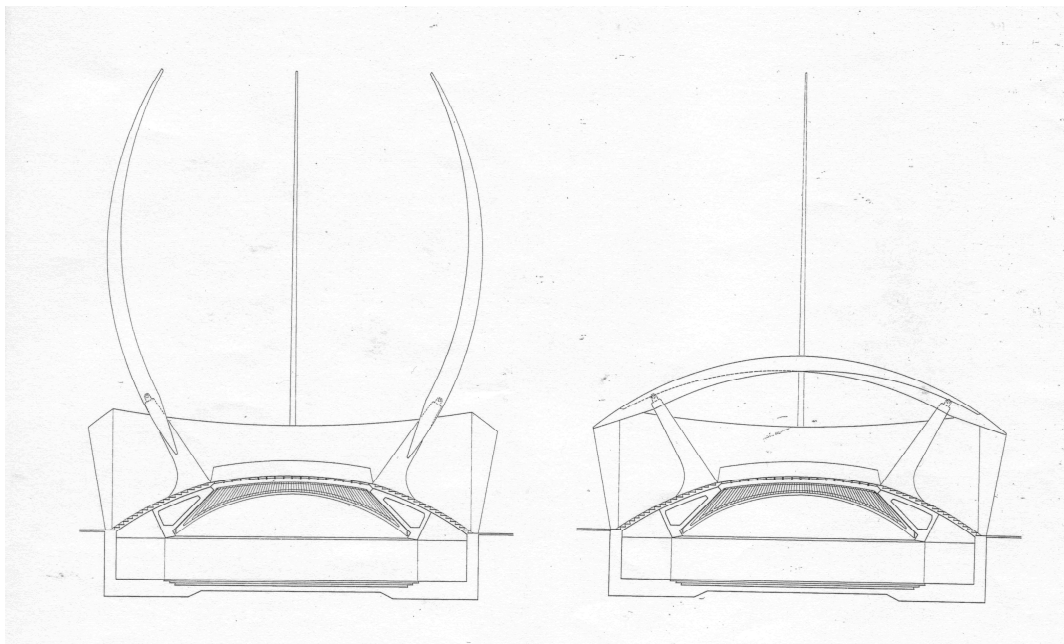


Figure 4-8. Drawings from Kuwait Pavilion (A. Tzonis)

4.3.2 Construction of Kuwait Pavilion

This first case study, as aforementioned, is the simplest in complexity of the three cases. The software was able, without further “trick”, to create the animation with the objects that were designed to represent the elements of the structure. The correspondence of the terminology between the real structure and the one implied in 3dMax is shown in **Figure 4-9**. Not many differences seem to exist in this case regarding the terms used, except of the column which in the software is called as element (base element).

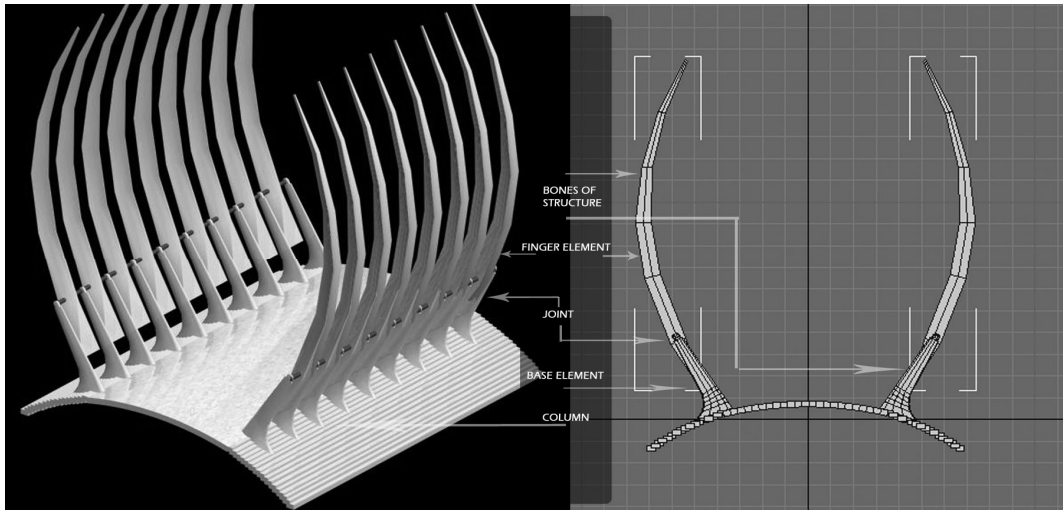


Figure 4-9. Terminology of real and virtual structure

Following the general procedure described before, the basic elements and object was created based on a drawing which was used as a template. In this case, the primitive kinetic object, which consists of the base-column and the finger element, was not set to be recognised by the program as a skeleton structure, as there was no need for complex animation; only the finger element is rotating around an axis. Therefore, there was no relationship between the elements and the hierarchy is independent. **Figure 4-10**. The whole kinetic object was copied and mirrored in order to have the realization of the structure. Lately the animation was created by posing each one of the elements in sequenced chronicle periods in order to form the desirable animation. The posing for every period was done with respect of the final animation and by understanding the possible position that each of the finger element should have in the different chronicle moments.

The time consume for the creation of this case study can be seen in the graphic representation shown below by thinking as analogical with the difficulty of creation. By using, from **Table 4-1**, each horizontal line as a stage for the definition of the structure into the animation software and comparing it with the level of difficulty that it proved during its manipulation, the graph shows how the relation of difficulty with each stage has occurred. **Table 4-2**. As visible, most of the time was spent on the definition of the constrains, the spatial arrangement of the objects and on fixing the animation.

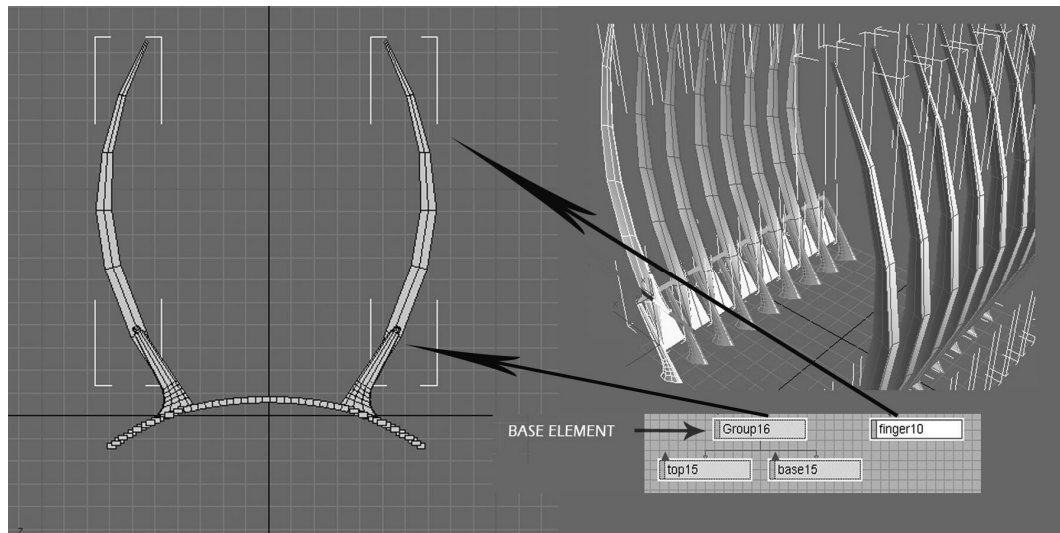


Figure 4-10. The hierarchy of the basic object. The selected elements are presented with different colour both in hierarchy graph and in scene

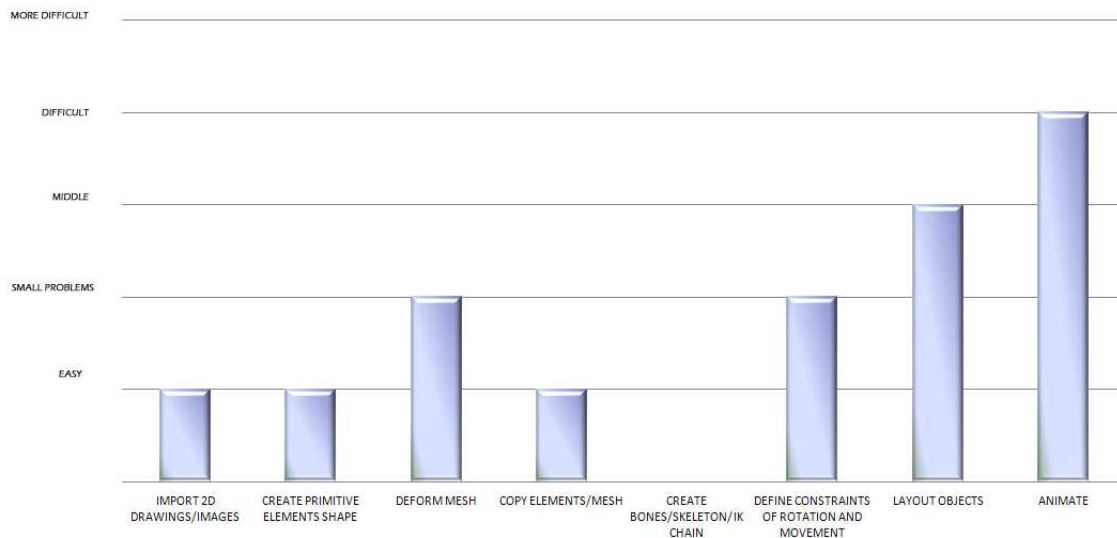
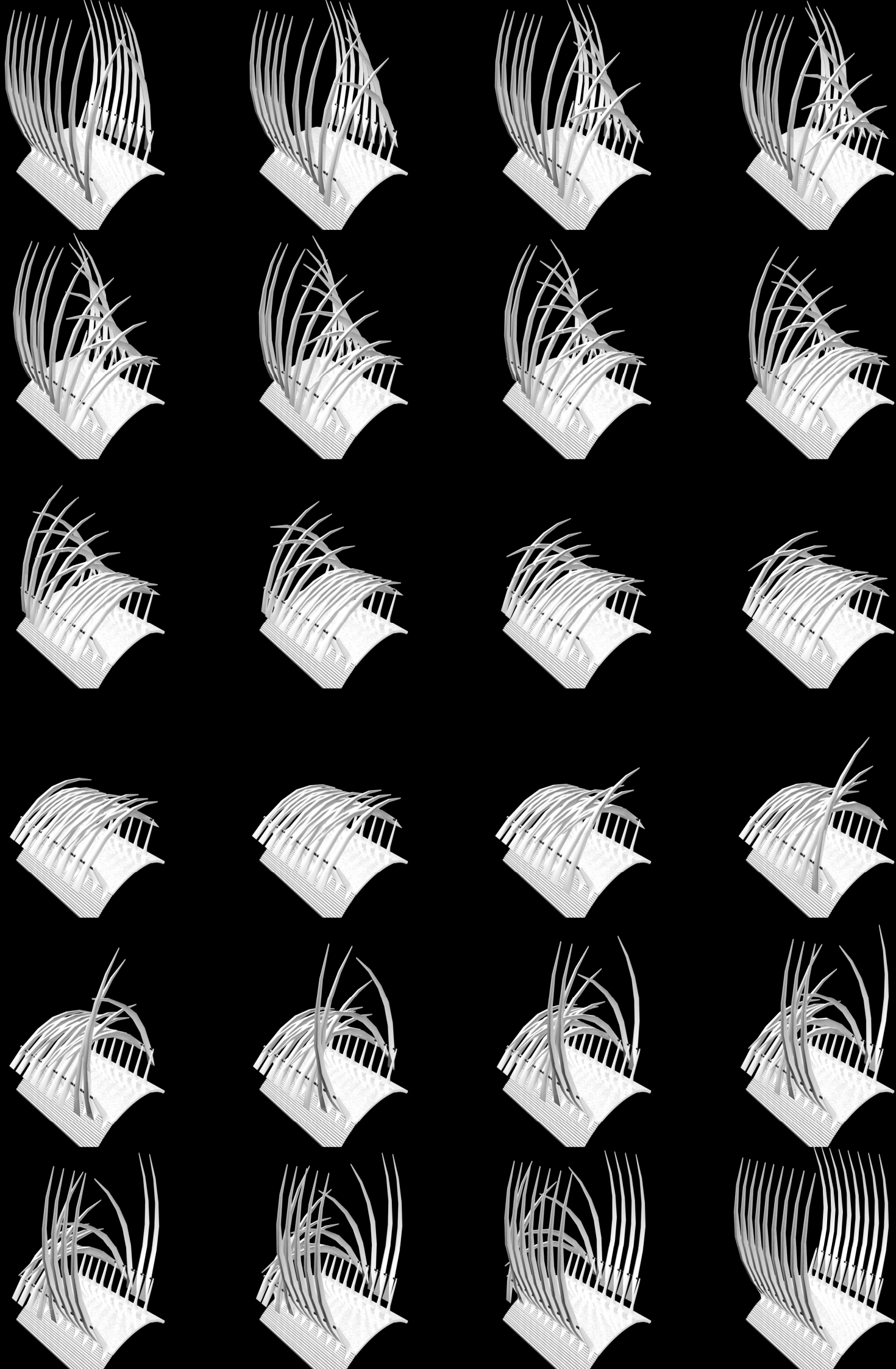


Table 4-2. Evaluation of each stage of the Kuwait's Pavilion structure definition according to the level of settlement's difficulty

The existence of many constraining functions but also the various possibilities of applying one, created confusion over the choice of the most appropriate for the needs of this construction. This caused the constant repeat of the stage until the proper constrain was found, leading to time lose. As after the coping of the basic elements the names were given to the objects in a

random way, reorganizing them in a logical way so that to be easily selected was necessary to be done. Regarding the animation, as aforementioned, the posing of the elements was done individually for each one of the 17 and therefore more time was spent in settling their angle of rotation for different moment instances.





4.4 Case study 2: Alcoy Community Hall

4.4.1 Introduction

Alcoy Community center was built in Spain in 1995 by Santiago Calatrava. The hall is situated in the heart of the town's Plaza Espana, site of festival of St. George and other community events. To preserve historical nature site, Calatrava created a subterranean multiuse civic hall with capacity of six hundred, ideal for civil weddings and exhibitions. Translucent glass floor panels attached to stain less-steel frames let light into the hall by day and emit a diffuse, mind glow by night that illuminates the plaza above. Above ground, a fountain, lights and an enclosed entrance break the continuity of the plaza surface to announce the presence of the underground space. Both the fountain and the entrance employ moving rod and joint mechanisms that produce the effect of veil or drape, a variation of Ernsting Warehouse and on the "eyelid" of the planetarium in Valencia. Building on the previous cases, the movement produced here is surprising, strange and in this fountain even eerie. As with the doors of the Ernsting Warehouse, Calatrava draws on both the technological principles of his Ph.D. dissertation and on the motor-spatial investigations of his sculptural experiments. The Hypnotic movement of stainless-steel door, which reveals an entrance cavity and stairs, as well as the folding cover of the spring-fed circular pool, at the opposite end of the plaza, offer a provocative prelude to the puzzles and marvels under the pavement of the plaza. (A. Tzonis)



Figure 4-11. Phases of the movement (A. Tzonis)

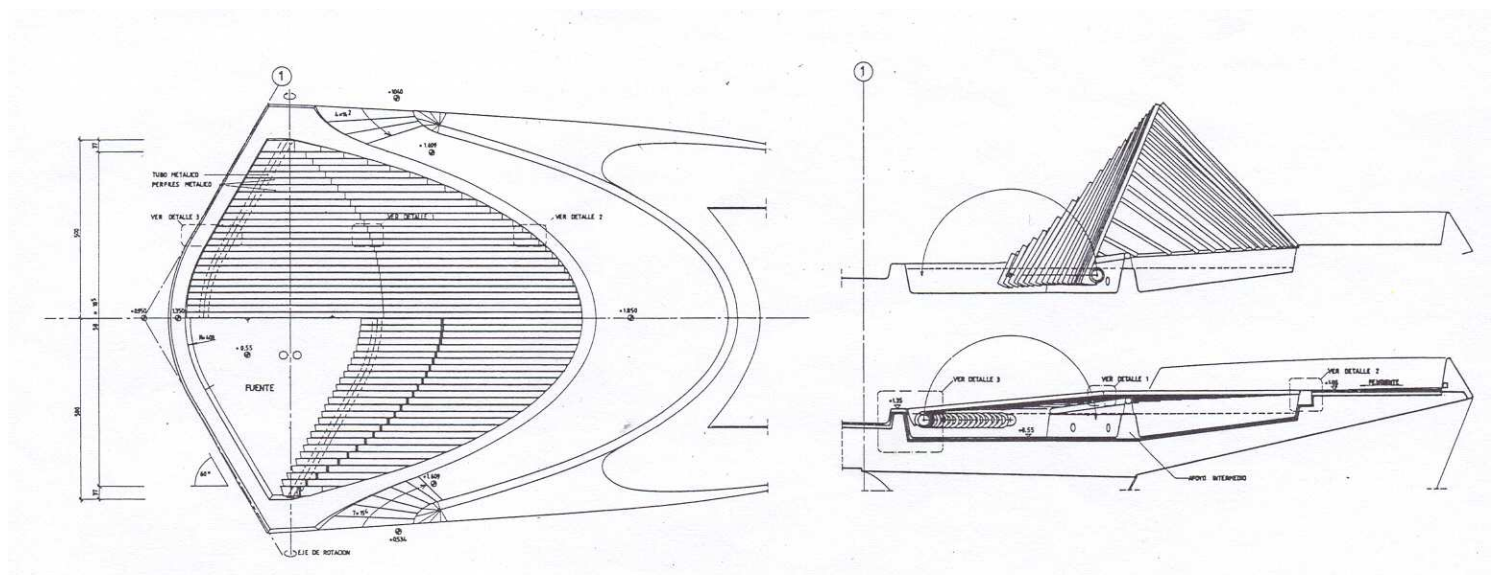


Figure 4-12. Drawings of the entrance(A. Tzonis)

4.4.2 Construction of Alcoy Community Hall

Alcoy community Center, as the second in complexity case study, has shown to be more difficult in its creation into the animation software than Kuwait pavilion. Here, as well, the general procedure of defining a structure was used to represent the structure into 3dMax software.

The first steps, associated with setup environment and the definition of the element, were performed easily as the basic elements of the structure have a simple primitive shape (oblong). In this case the steel frames are called box elements into software language whereas the rest use the same terminology. The correspondences between the constructive language and the software can be seen in **Picture 4-13**.

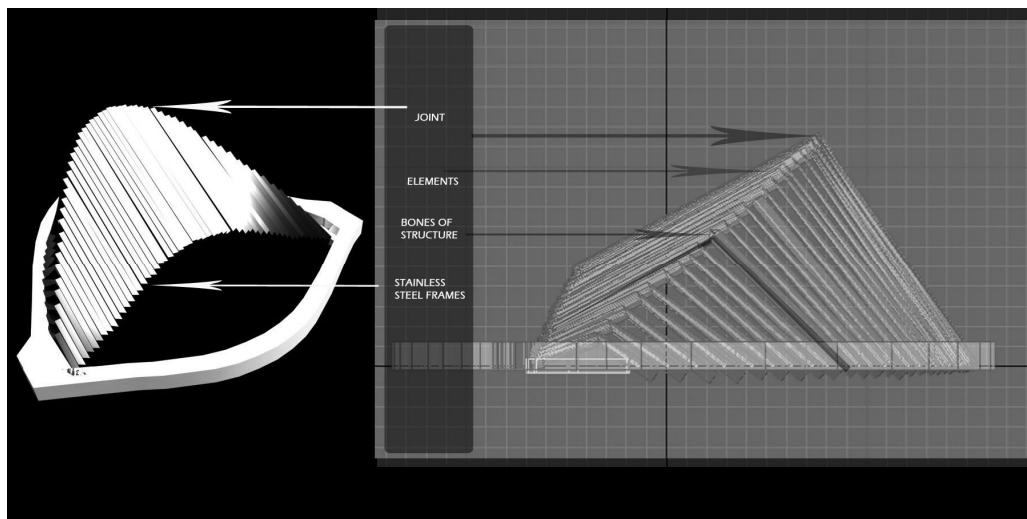


Figure 4-13. Terminology of real and virtual structure

However this case differs from Kuwait pavilion in the fact that its objects are not of the same proportions, they are not analogical and have different length. Therefore, the elements and eventually all the objects were created from the beginning individually. For this reason, as seen on the **Table 4-3**, the fourth stage was thought to be demanding whereas the seventh does not exist as it is included in the fourth.

In addition, the animation of Alcoy Center was difficult to be performed. With the existing constrain parameters and hierarchy connections, the representation of the real movement of the opening entrance could not be animated. The connection between child to child into independent hierarchies is not supported by the software and can not be overcome with appropriate constrains. When an attempt was made to restrict in position each one of the last in hierarchy elements of the moving frame objects in respect to their neighbour elements, the outcome of the structure's movement was not the anticipated one.

In the real construction, all the moving frames are articulated on a long bent rod in specific equal positions. (**Figure 4-14**) However, this could not be done into the software as while creating an element, this is considered to have only one point of connection, at the pivot point. Under this consideration all the objects of the construction when tried to be connected with an element representing the bent rod, where joined in the same position and the result

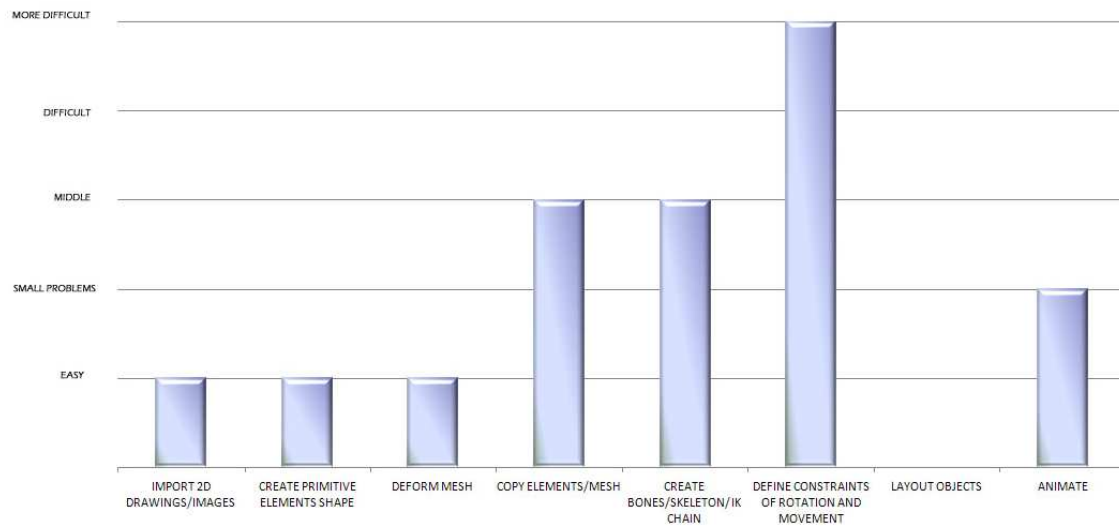


Table 4-3. Evaluation of each stage of the Alcoy's Community Center structure definition according to the level of settlement's difficulty

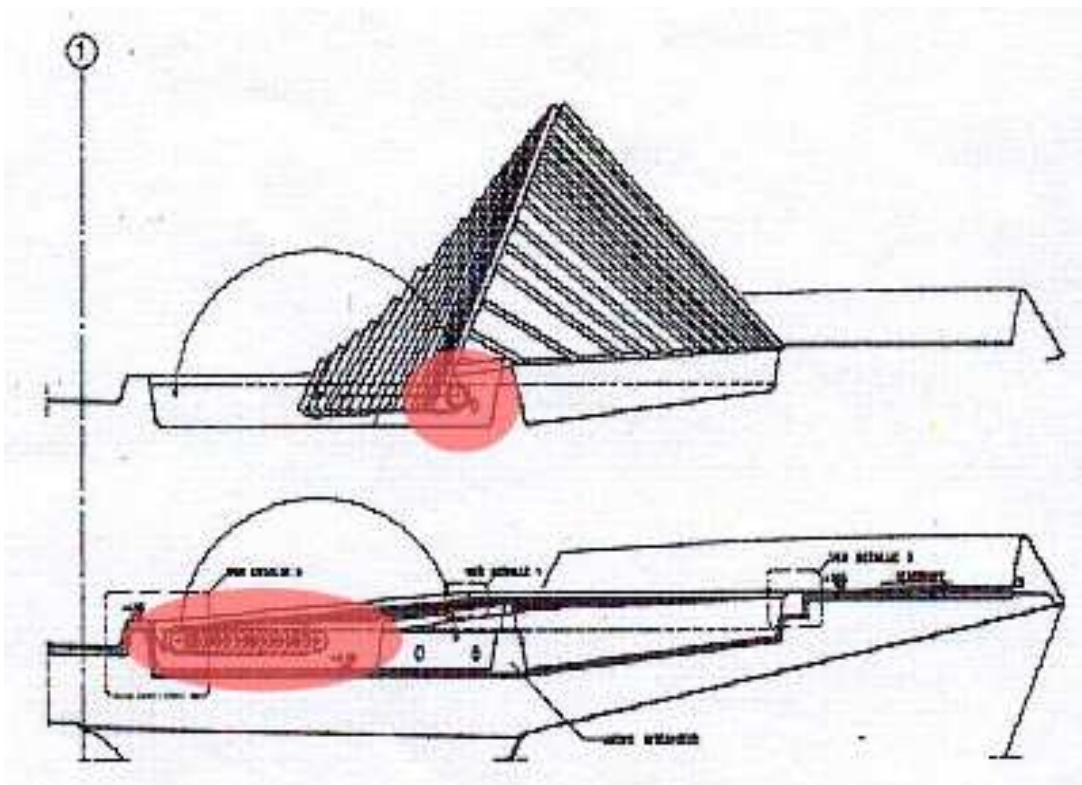


Figure 4-14. Position of the bent rod where all the objects are articulated

was that they were moving along the same semi-circle whereas in reality each part of the structure is moving along a different, with analogical increasing or decreasing radius, semi circle. Therefore an indirect way had to be used in order to achieve the animation .

This was done with the creation of a cluster of different skeleton structures. The first type of skeleton construction regarded the moving metal frames and the connections between them. Two bones were created to be positioned inside the two metal frames, which in the software were represented by two boxes with similar proportions with the real structure, that form one object with a hierarchy of child to parent shown as in the **Figure 4-15**. The boxes were attached afterwards to the bones as skin to the skeleton. For their movement, in order to be restricted in the angle of rotation and to resemble to the physical movement of a human leg (including the thigh, knee, shin), the IK Solver, a function was used, which has as a result the appearance of a rope in scene and it is not rentable(IK Chain). The same procedure was done for every pair of metal frames.

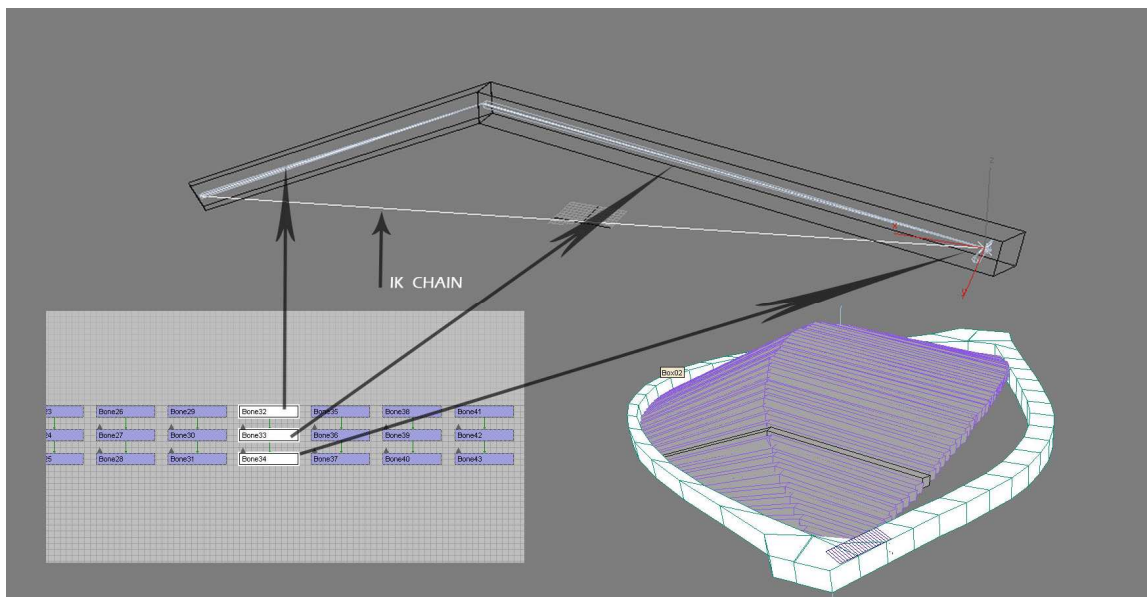


Figure 4-15. Hierarchy of the basic object

The second type of skeleton structure was created to stand for the bent rod. In order to achieve the curvature but also the possibility of different points of connection, the rod was formed from bones in a continuous hierarchy as shown in **Figure 4-16**. Each bone has a dimension equal to the vertical distance of one IK Chain to the other between two objects and follows at the same time the shape of the bent rod. (**Figure 4-17**) In a skeleton structure, when no constrain or movement restriction is given, all the bones follow the position or rotation of the main parent bone acting as a massive articulated object when this parent bone is moved or rotated. On the contrary, when a child is moved, this doesn't affect the parents of the hierarchy before it. Lastly, all the IK Chains, which manipulate the movement of the joined frames, were constrained in position to the corresponding bone of the rod skeleton structure (**Figure 4-18**) enabling, during the rotation of the parent root of the rod, the whole structure to simulate the real movement of the Alcoy entrance.

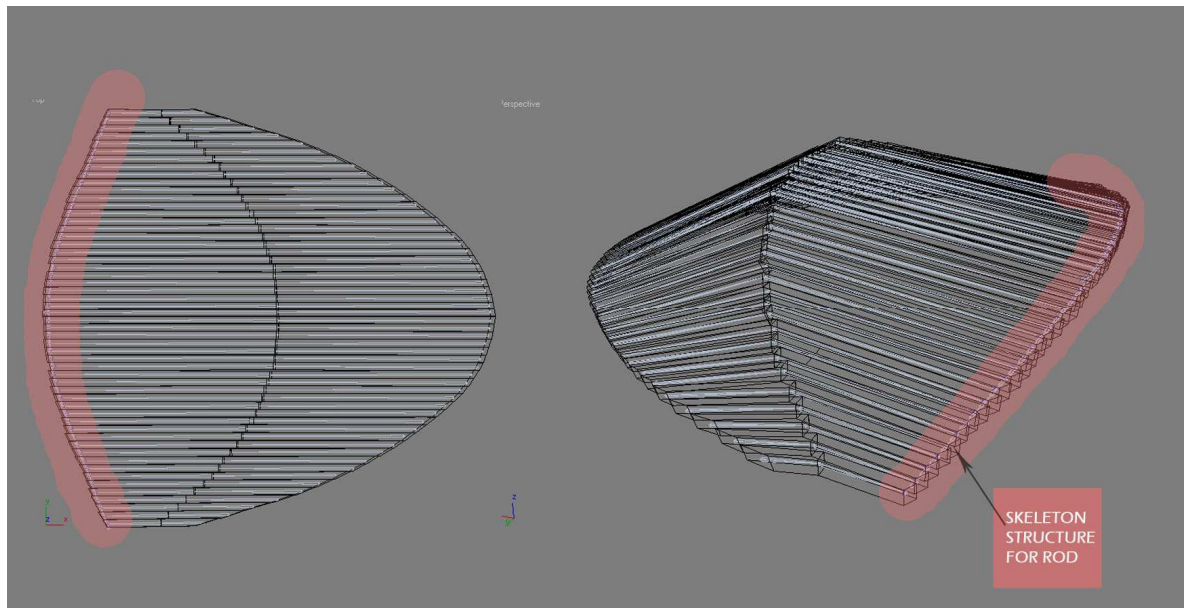


Figure 4-16. Position of the skeleton rod

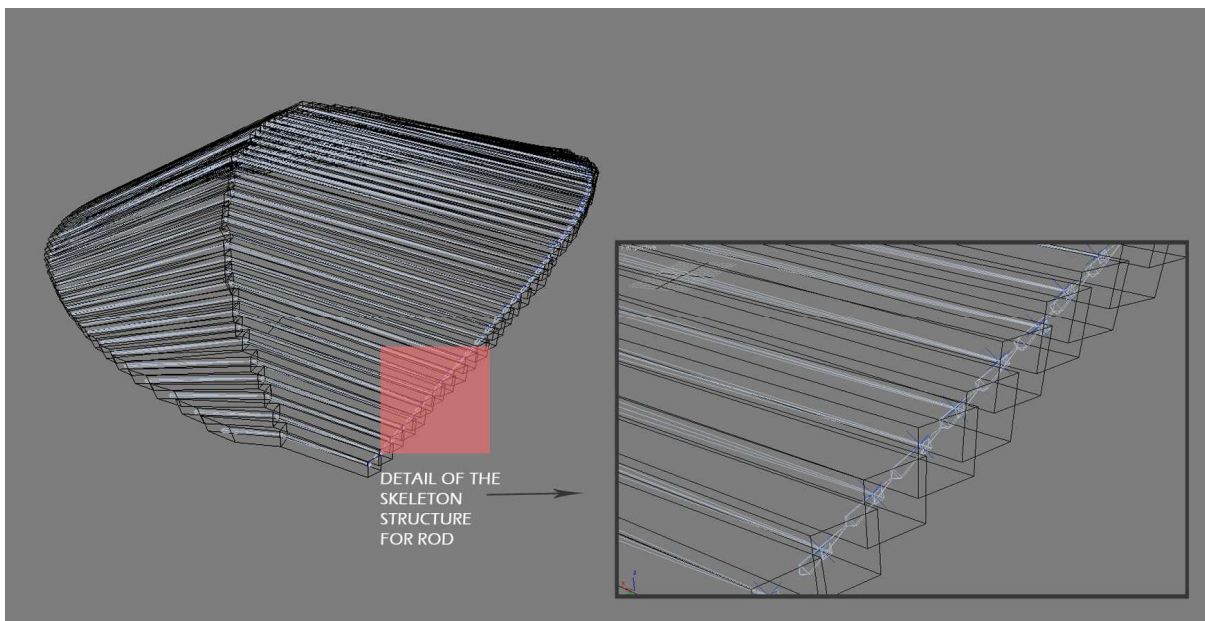


Figure 4-17. Detail of the connection between the rod and frame skeleton structure

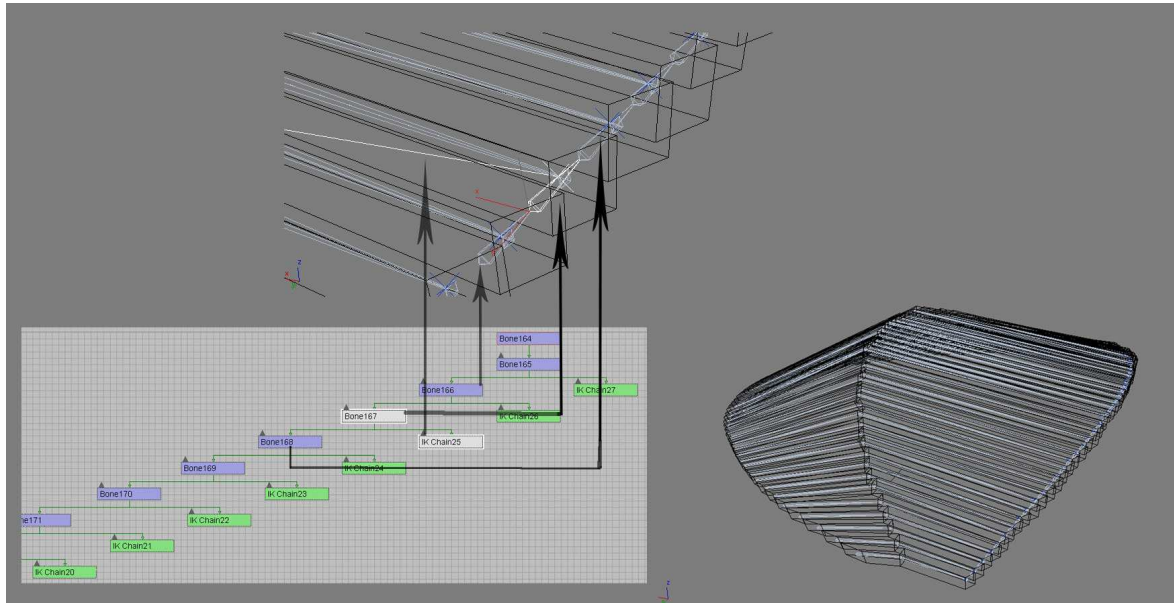


Figure 4-18. The hierarchy of the rod. The selected elements are presented with different colour



4.5 Case study 3: Folding Egg

4.5.1 Introduction

The Folding Egg is a prototype kinetic folding sheet created by the Kinetic Design Group of Massachusetts Institute of Technology. This structure is made by low-cost recyclable material and is essentially a collapsible tree-dimensional truss structure. It has a structural stability and many possible configurations.(**Figure 4-19**)

The basic principles and shape that this construction uses are not newly formed but have been explored in previous years with different constructions. Examples can be seen in **Figure 4-20, 4-21** where a church and a model have been built over the same principles. Generally it can be said this kind of constructions belong to the optimized in efficiency constructions. However, the Folding Egg differs from these structures as additionally it has the ability to reconfigure itself through kinetic movement.



Figure 4-19. Different views of Folding Egg(Internet)

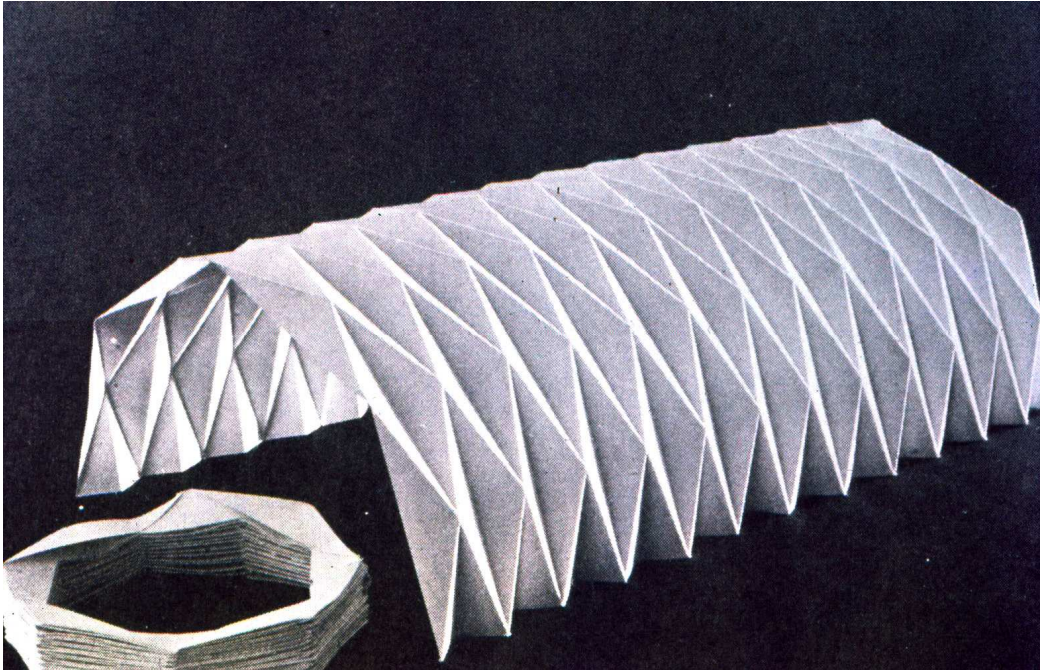


Figure 4-20. Paper model of a folding construction (V.Sedlak)



Figure 4-21. Inner space of a church (V.Sedlak)

4.5.2 Construction of Folding Egg

This case has proved to be the most difficult study in realization from all the three. The same procedure as for the rest studies have been followed in this one as well, but from the early steps, obstacles appeared. The difficulties in each stage can be seen on **Table 4-4**. In order to be designed, the structure had been simplified to the representation inside the software only of the main truss that supports the whole construction without including the panels and only for a segment, the one which could be considered as the

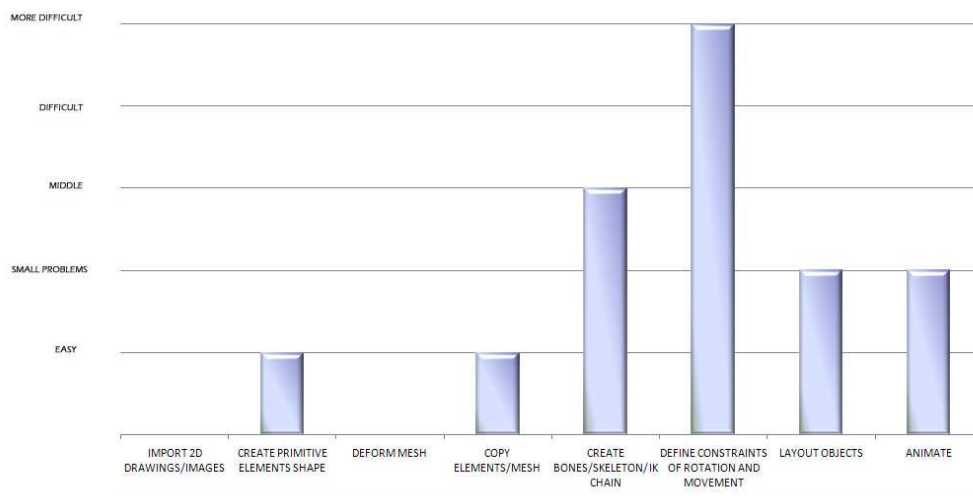


Table 4-4. Evaluation of each stage of the Folding's Egg structure definition according to the level of settlement's difficulty

basic object of the structure **Figure 4-22**. The important thing for the representation of this construction into 3dMax is the existence of the main truss, on which the whole kinetism is depending, and the restriction of its elements to a curtain length which will not transform during the animation.

This basic object was as well divided into three different parts, each one corresponding to the three different directions of elements which result to the curvature of the basic object **Figure 4-23**. Firstly, the lower part of the three sub-objects was created by a number of primitive elements (cylinders), properly spatial arranged to form the truss and the skin of the skeleton system attached inside. The correspondence between the actual structure and the virtual can be seen in **Figure 4-24**

The first attempt was each part of the basic object to be designed in such a way that the deformation and the movement will be done due to an horizontal set of forces. The creation of the skeleton was difficult to be formed as it had to be created with a consideration of the direction of the movement, the hierarchy and the different constraints that would be eventually implied; all of them would affect the mobility and the result of the animation. Therefore a lot of attempts have been performed until the proper positioning to be found. At the same time the restrictions in rotation and in position were difficult to be performed; every set of elements had to be constrained in position and rotation and then again, as one object, to be restricted with the rest of the analogical objects.

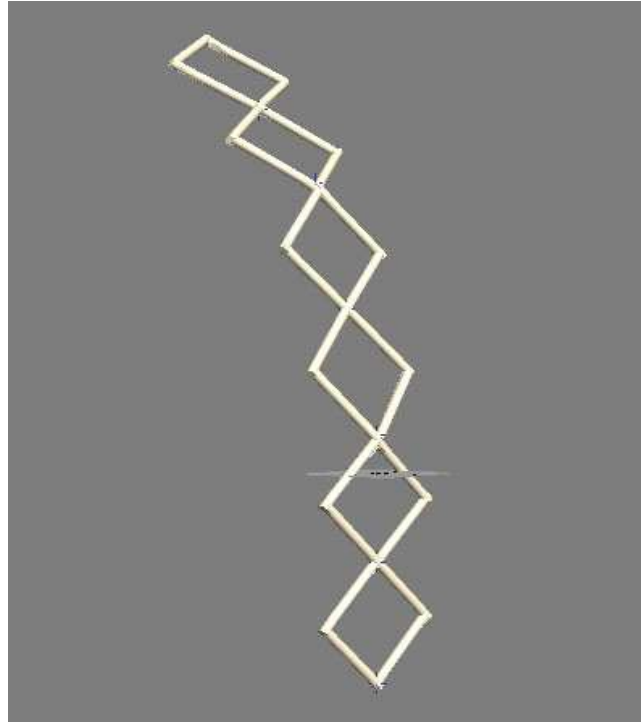


Figure 4-22. Basic Object

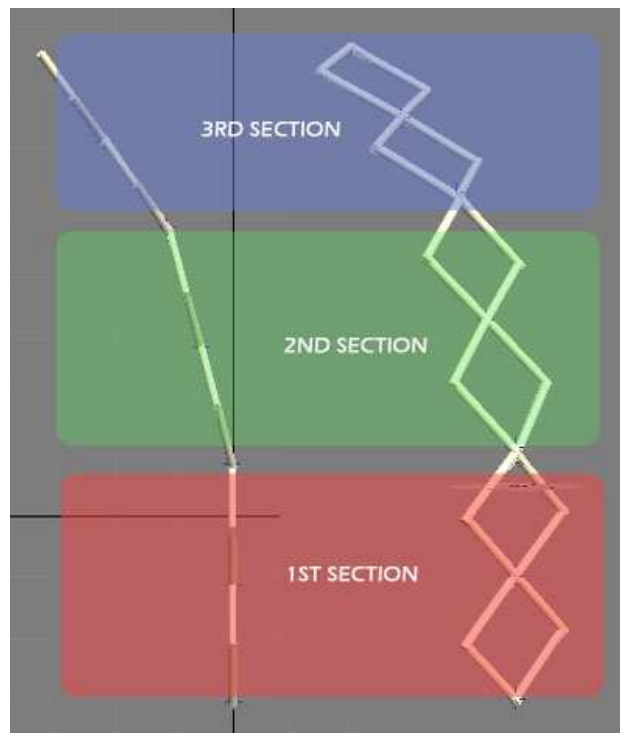


Figure 4-23. The three parts

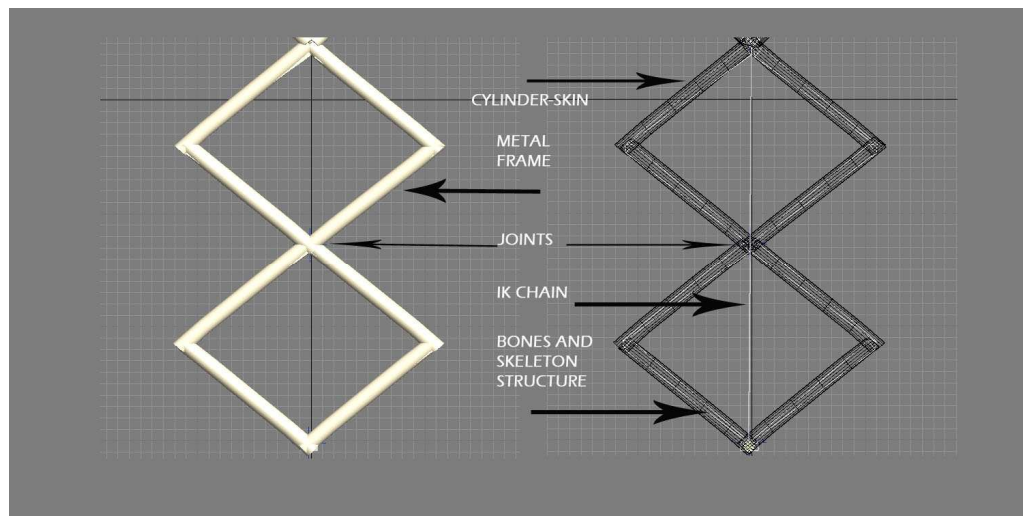


Figure 4-24. Correspondence of terminology

Nevertheless, the creation of the construction to this axis didn't work. While testing the movement of the basic object, the three different parts, during the stretching of the object, were keeping distances as seen on **Figure 4-25**, something which couldn't be overcome. The animation into two directions could not be performed with the existing functions, without the interference of the user to transfer the elements in the right position for the different keyframes of the animation during the stretching of the object. Therefore, the whole construction was created from scratch with respect to the vertical axis.

In this case, the three parts could move, with a small declination to their analogical distances and the real movement, without separating, into the three different directions **Figure 4-26, 4-28**. However, when an attempt was made to multiply the basic object, it had the same result as with the previous case, the objects were keeping distances between them **Figure 4-27**. Nevertheless also in this case, the real movement of the structure could not be performed.

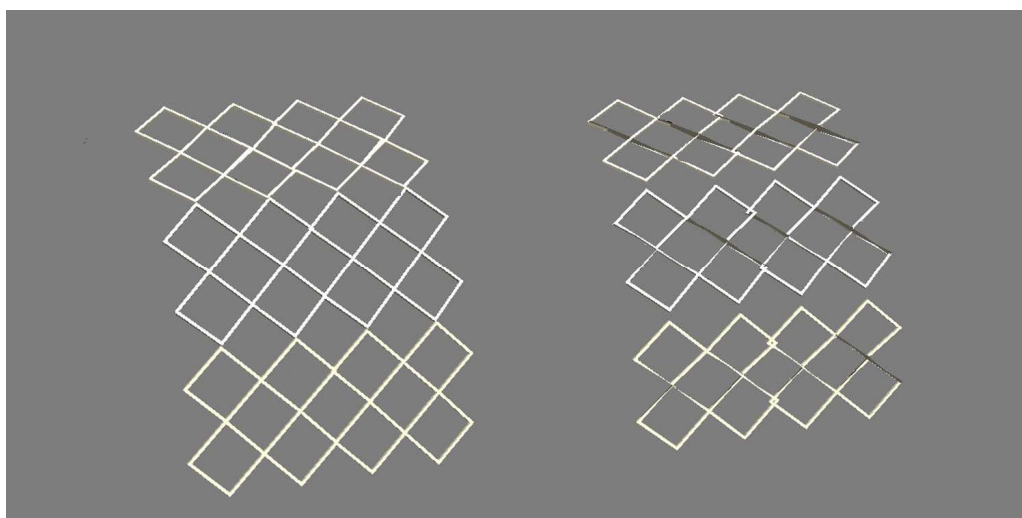


Figure 4-25. Distances between the three parts

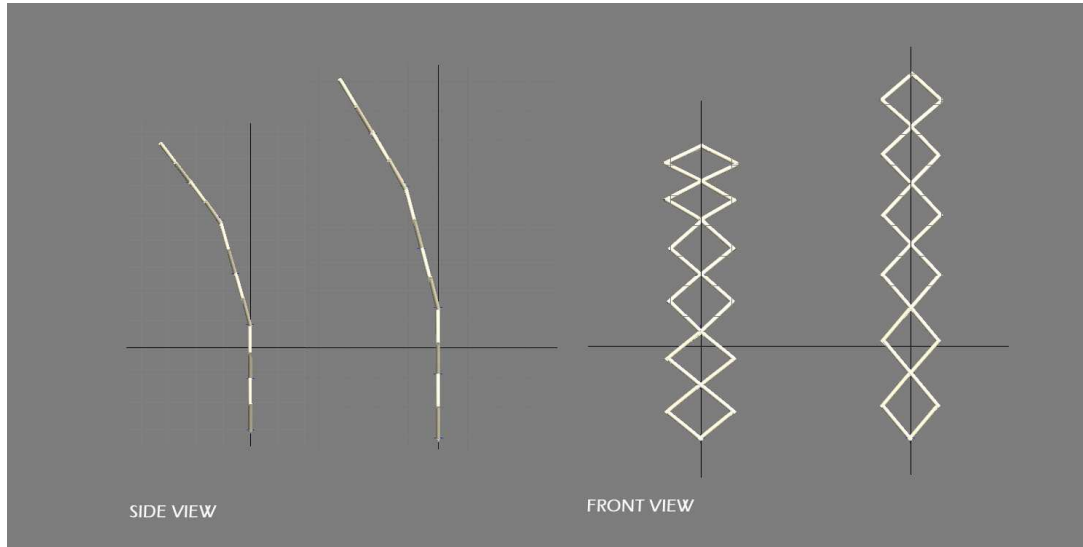


Figure 4-26. Movement of the basic object

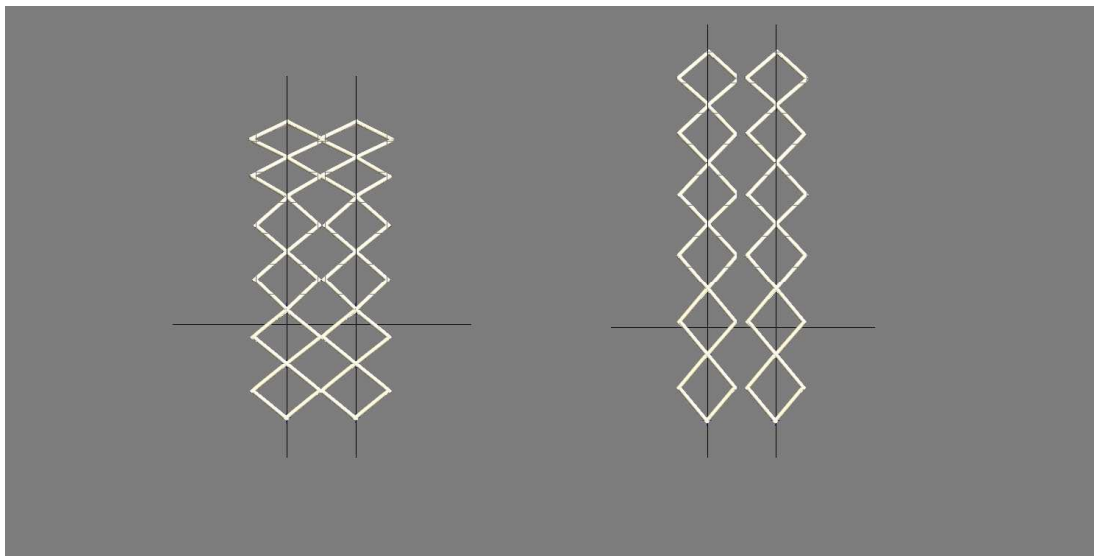


Figure 4-27. Distances between the two parts

As seen on **Figure 4-28** the elements and the different sections are keeping the angle between them almost the same throughout the whole animation and are changing their proportions as expected due to the expending truss. However, the real structure keeps the average height of the construction stable and is adjusting the expansion of the elements by changing the angle between the three sections. But this could not be simulated into the software with automation but with the interference of the user for each keyframe, as again the movement of the structure in two different axes at the same time had to be represented.

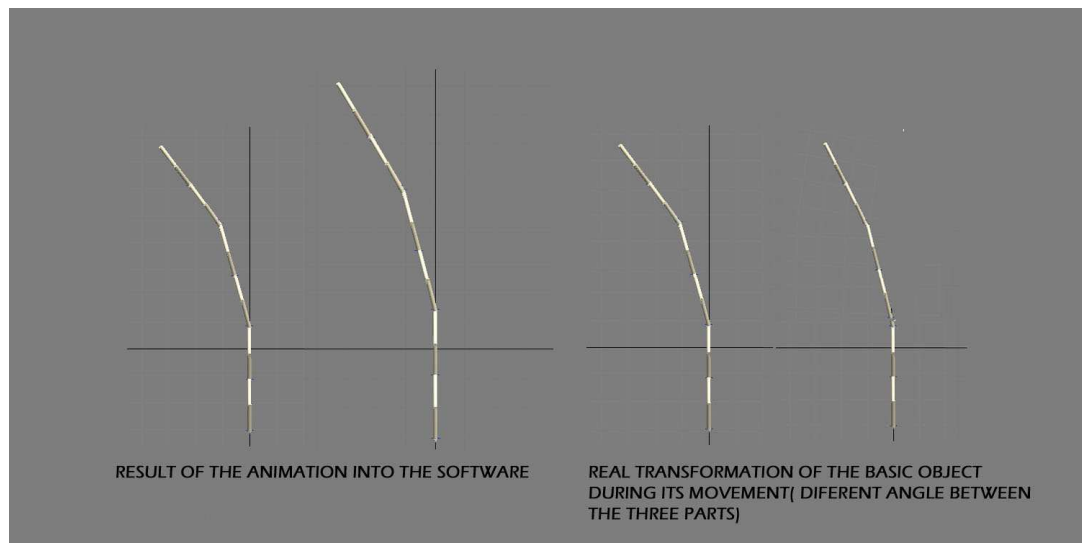


Figure 4-28. Movement of the basic object in animation and in real construction

5 ASSESSMENT

5.1 Benefits and limitations

Seeing throughout each case study has been evident that in various circumstances the software was proving to have some limitations in function, regarding the definition of the structures. These can be identified in the **Table 4-1, 4-2, 4-3** as the stages in which the complexity of settling the functions and the structures was arising incredibly and which are different for each case study. Truly, during the analysis, various points have been established as vulnerabilities of the software. Lack of useful functions have appeared, forcing indirect ways or patents to be used in order to overcome the difficulties, or, even in cases where functions existed, the lack of automation made a constantly repeating procedure to be performed for many similar objects where the same definitions were needed, consuming this way useful time.

By comparing the three case studies and their graphs regarding the level of difficulties throughout the definition of the structures, and in particular by recognising the stages, in which the software is malfunctioning, can be the first step towards the solution for the problem statement of this thesis. The knowledge of which kind of limitations the software has towards the creation of kinetics structures into its environment, can be the base of the further development of the software through the overcome of the malfunctions. The **Table 5-1** shows this comparison of the three case studies.

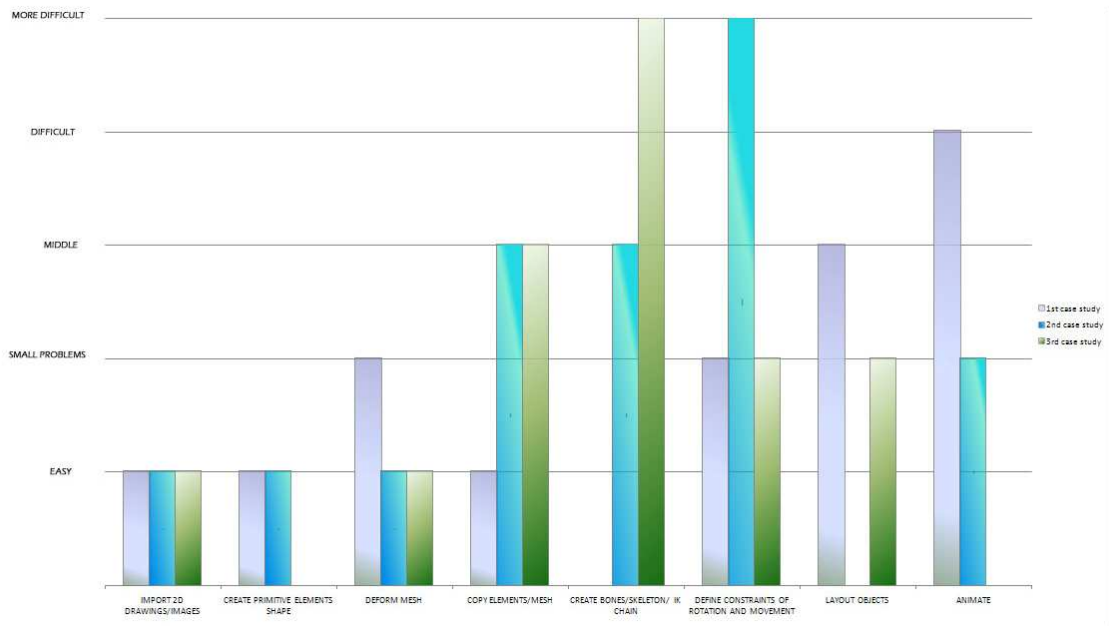


Table 5-1. Comparison and evaluation of the case studies of each stage of the structures definition according to the level of settlement's difficulty

Truly it is evident that in some particular steps the difficulty is increasing, sometimes regarding not only one case study but two or more. Specifically, apart from the first case study which, due to the simple use of the elements without any skeleton construction and therefore no automation, can surely be considered as an individual one, the other two cases seem to have the same difficulties during their settlement. In the first steps, where the simple set up of the software's environment and the creation of the primitive elements is performed, none of the case studies seem to be facing a problem apart from the deformation of the mesh and the coping of the elements that case study one and two equally show, but due to the shape of their construction. However when the creation of the skeleton structure and the implementation of constraints are needed, the procedure for Alcoy Center and Folding Egg becomes really troublesome.

By generalizing and categorizing the findings of this comparison but also by analysing every case separately, can be said that the main problems that 3dMax shows toward the representation of kinetic structures are:

- Spatial arrangement in combination with the naming that the software provides
- Difficulty in changing common parameters for a selection of many elements of the same category
- Lack of automation in various functions
- Problem in attaching elements or primitive objects between them - connection possible only in the pivot
- Problem in the hierarchy in child to child connection
- Difficult in arranging the constraints in complex structures - complicated connections must be formed
- Difficult in manipulating the animation into 2 or more axes at the same time with automation- can be performed only manually
- Difficult in manipulating the animation into 2 or more axes at the same time with automation or not when also the complex system of elements belong to different orientation and axes

From the first case study it has been evident that the ability of the software to automatic name the new copies of an element or an object, can be very useful but at the same time can create problems. In cases where more than one element was selected from the scene with different position, e.g. in the case of Kuwait Pavilion the first basic objects positioned in the beginning of each row which would form the whole structure, the names that were given in the copies did not follow a sequence but seemed to be given randomly. This, in structures where a lot of similar objects exist, can cause a time consume in working and manipulating with them as a lot of time will be spend in the recognition and selection of each desirable element.

In addition to that, though the selection of all the elements belonging to a category, such as "finger" elements from Kuwait Pavilion case study, could be chosen at the same time, their modification is not possible. When the same transformation or implementation of function is needed for the whole category of elements inside a structure, the software does not give the possibility of this group modification and therefore it has to be done separately for every element. This, in combination with the naming issue, can waste valuable time in the whole procedure.

This lack of automation can be found in various functions in 3dMax software. Not only in group modification but also in other cases 3dMax

seems to be unable to present a functionalism in order each assignment to be done easily and fast. It is understandable that the software can not predict in which cases this automation is needed and can more easily understand, handle and avoid mistakes in functions when these are assigned to the elements individually. However, for a fast construction of structure where similar elements exist the existence of cybernation in same cases could be useful.

In the second case study more specific problems have appeared regarding the connections and relationships between the elements. Firstly it has been evident that the connection between the children of different hierarchies can not be performed. Regarding the matter of the bent rod in real construction and its representation into the software, the first attempt was to link hierarchically the latest in hierarchy elements that formed the moving frames. However, that possibility was not given by the software. The second attempt was to restrict them to follow the position of their neighbour element something that though it gave a movement outcome was not the anticipated one. The elements were moving according to their neighbours' position but didn't keep equal distances between the maximum and the minimum path of movement, giving by this way another result. When finally an element to resemble the bent rod was designed and connected with the moving frames, the result was even worse than in the previous case, as all were connected and followed the movement of the pivot point of the rod throughout a certain arch. Therefore the skeleton structure was created in order the rod to be represented and each moving frame to be attached in a certain point which followed a unique arch path.

Finally, one of the main weaknesses of 3dMax, which proved to be the most difficult to be overcome, was the arranging of the constraints in complex structures and their afterwards animation. The third case study was proved to be a good example of this problem as, in order to assign the movement and the relationships between the elements in position, many complicated connections had to be formed with the help of constraints. For each one part of them, as described before, constraints over existing constraints had to be implied in order to achieve the desirable movement. However, even after all these, the case study did not succeed, as the animation didn't resemble the real kinesis. The difficulty in manipulating the animation into 2 or more axes at the same time with the automation which was offered by the constraints and in addition to that when also the complex system of elements belonged to different orientation and axes, could not be overcome and the final animation could not be considered as successful.

One can say that until now only the limitations and none of the benefits of the software have been presented. However, in this kind of research, it can be said that already the recognition of these limitations is at the same time the benefit regarding the thesis. By understanding the problems that the animation software has towards the implementation of the movement it can be the beginning of the solution. Anyway, the useful functions that have been noticed inside the software have been the anticipated ones while evaluating and choosing the software. The only non expected function, which proved to be very useful and fundamental in the creation of the animation, was the implementation of the IK solvers and consequently the creation of the IK chains..

5.2 Enhancements

It is undoubted that 3dMax needs further investigation which cannot be covered by 3 case studies. It might be possible that solutions to the problems mentioned before already exist, but in a very complex way. The program itself covers a very wide field and includes a great deal of functions. However it seems difficult to a new user for these solutions to be discovered. Therefore an attempt has been made to solve some problems of the ones mentioned before or to create a start point, as a sample, for the further development of the software.

One of these attempts regarded the spatial arrangement of elements or objects in combination with the naming order. With the help of the scripting language, two new toolbars have been created, one for an axial arrangement of the objects and the second for a spatial array. Based on an example that is already included into the 3dMax script, by making modification into the functions needed but as well as in the equalizations that arrange the objects in space, a desirable result has occurred. **Figure 5-1**

Both of the toolbars have the possibility by selecting an object already created into the scene, regardless the shape that it has, to create the number of copies, instances or references by selecting the type of

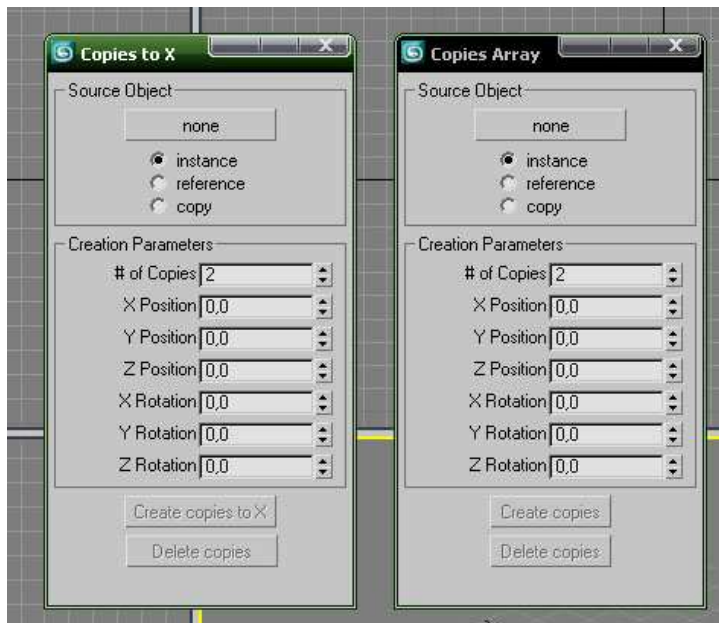
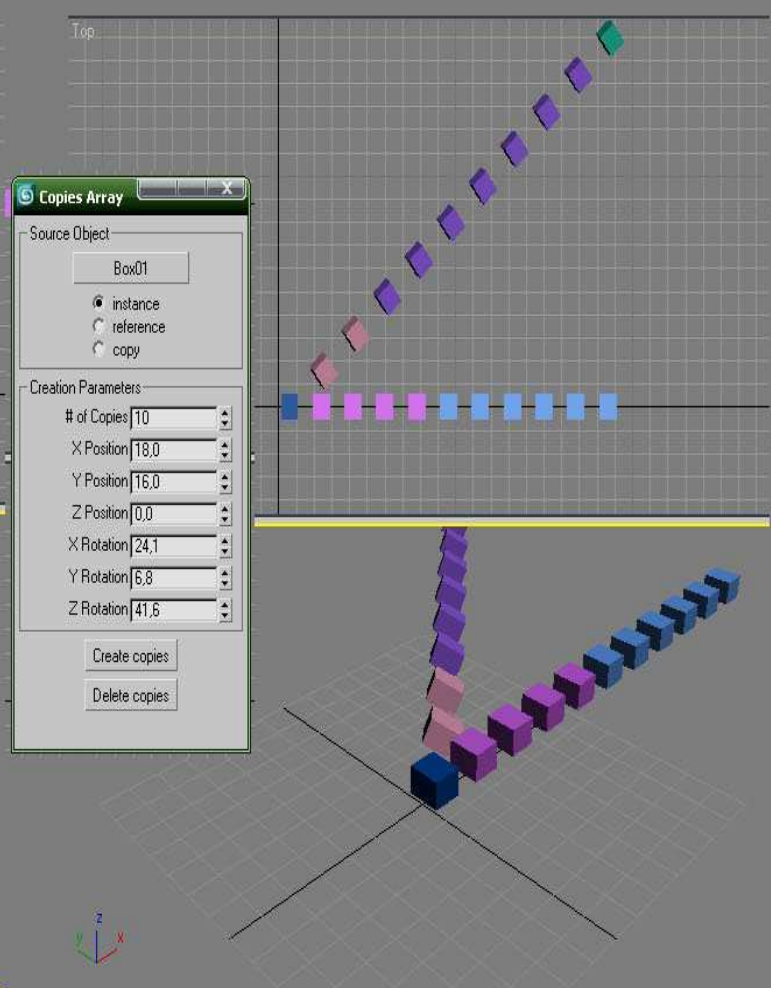
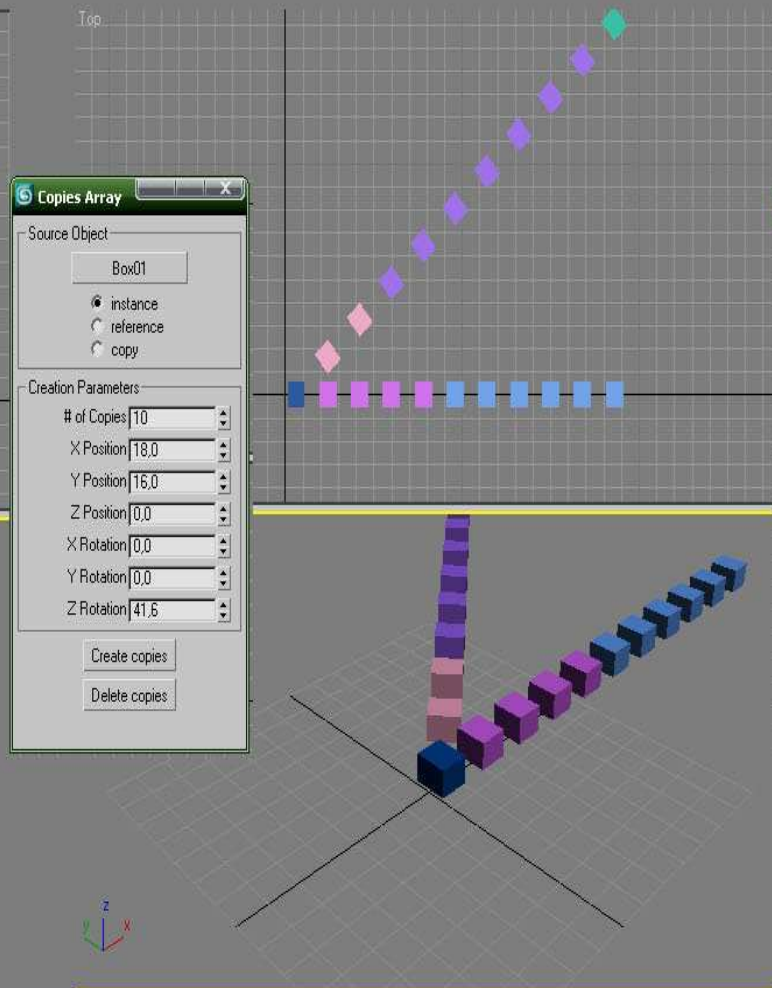
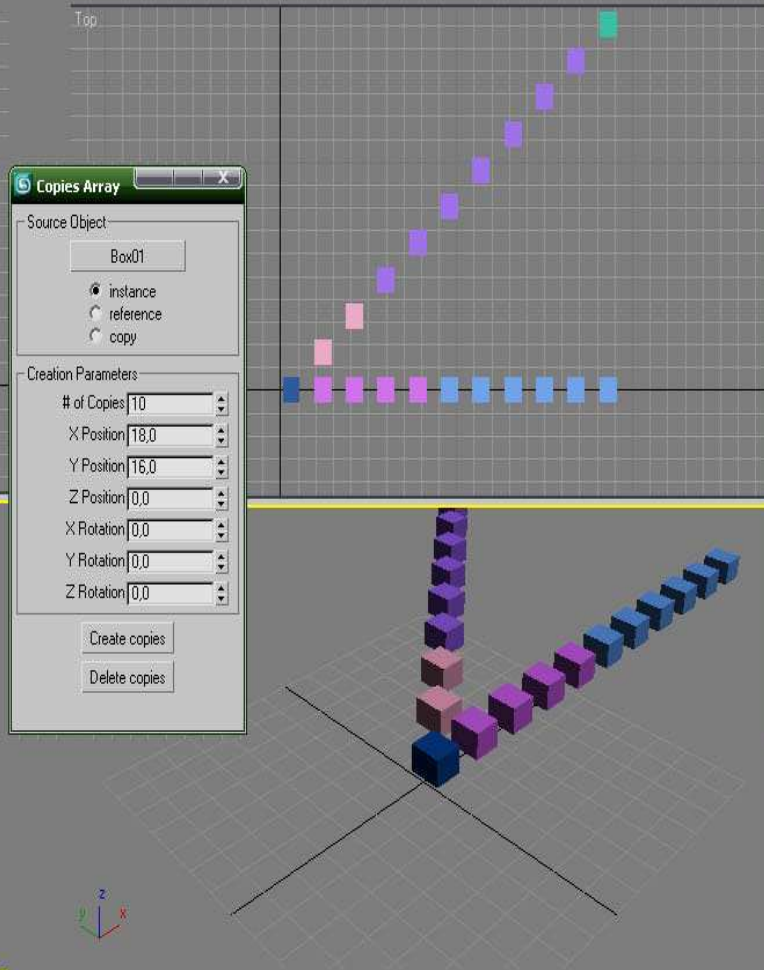
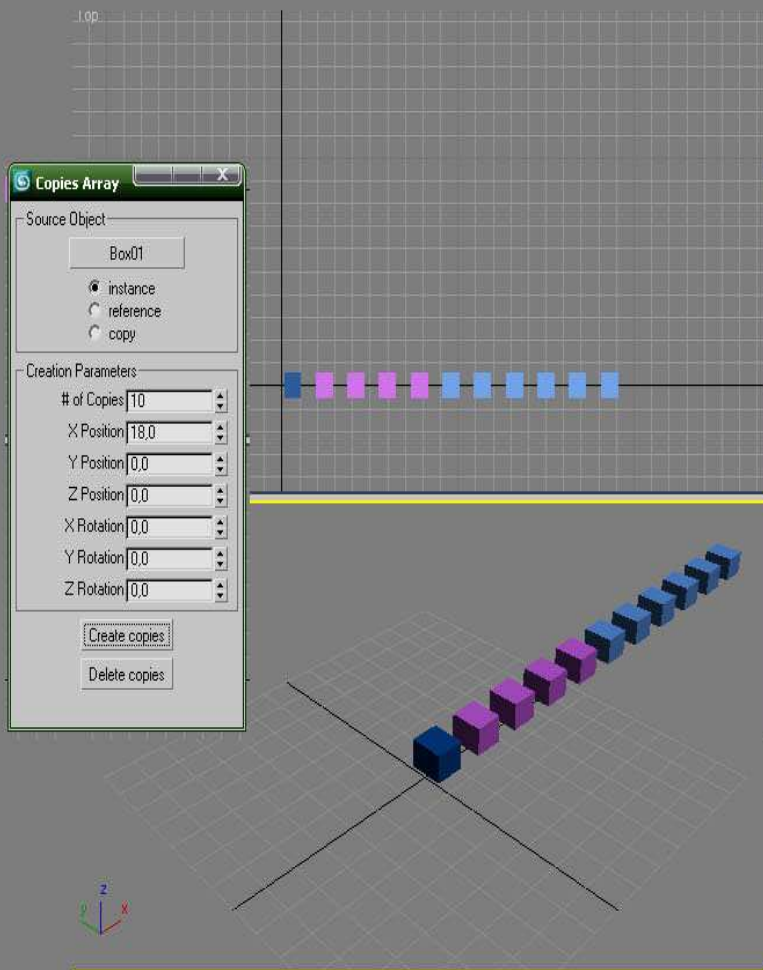


Figure 5-1. The two toolbars

copies of the three mentioned and by giving the number with a spinner button. At the same time, always with the use of a spinner button, the user can select the distances for the basic object of the copies in X, Y and Z axis but also the angle of rotation in each one of these. Lastly, the copies are created with just a push of the button “Create Copies”, and if no other function has been performed in the meantime, they can still be modified in every position parameter included in the toolbar or even deleted. The most important is that all the copies preserve an order during their creation, no matter how many in number, by facilitating their selection, for further modification, for the user. The following Figures give examples of the toolbar



Top

Copies to X

Source Object

Box01

☒ instance
☐ reference
☐ copy

Creation Parameters

of Copies 10

X Position 18.0

Y Position 0.0

Z Position 0.0

X Rotation 0.0

Y Rotation 0.0

Z Rotation 0.0

Create copies to X

Delete copies



Top

Copies to X

Source Object

Box01

☒ instance
☐ reference
☐ copy

Creation Parameters

of Copies 10

X Position 18.0

Y Position 23.0

Z Position 0.0

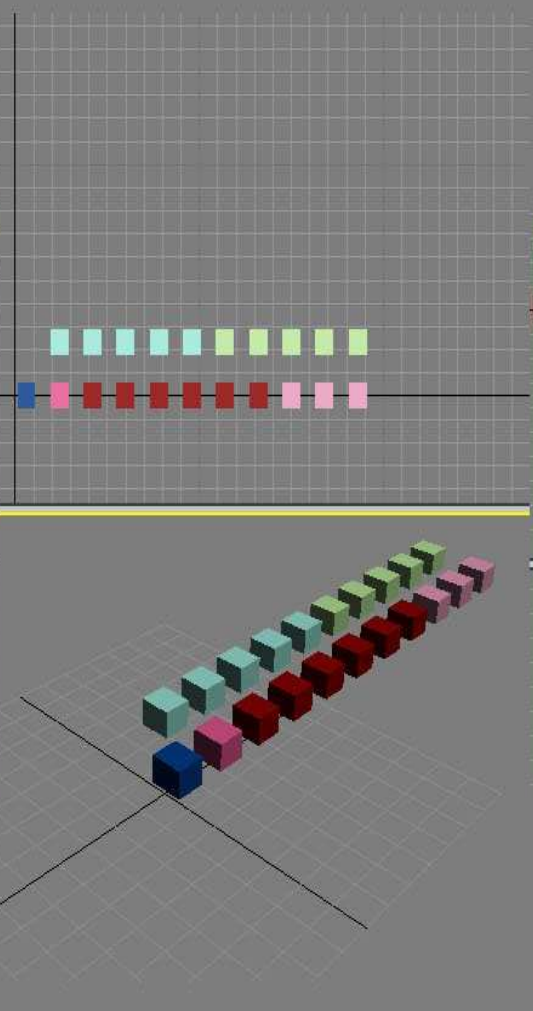
X Rotation 0.0

Y Rotation 0.0

Z Rotation 0.0

Create copies to X

Delete copies



Top

Copies to X

Source Object

Box01

☒ instance
☐ reference
☐ copy

Creation Parameters

of Copies 10

X Position 18.0

Y Position 23.0

Z Position 0.0

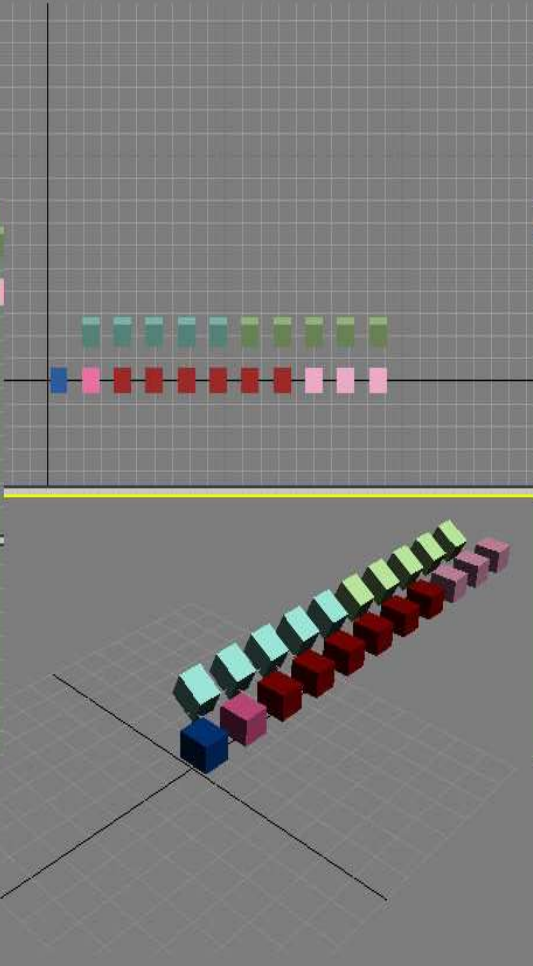
X Rotation 27.8

Y Rotation 0.0

Z Rotation 0.0

Create copies to X

Delete copies



Top

Copies to X

Source Object

Box01

☒ instance
☐ reference
☐ copy

Creation Parameters

of Copies 10

X Position 18.0

Y Position 23.0

Z Position 0.0

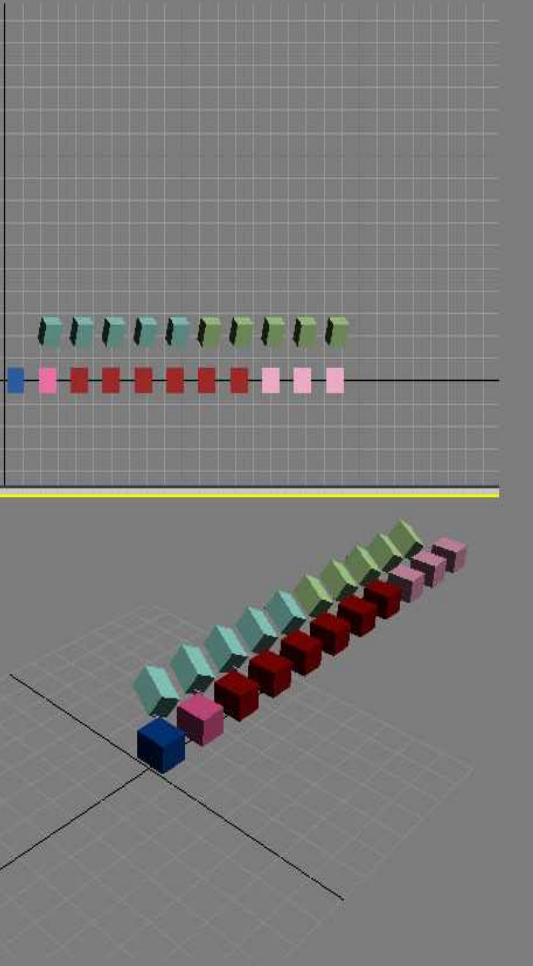
X Rotation 27.8

Y Rotation 24.6

Z Rotation 0.0

Create copies to X

Delete copies



This has been a small sample of the possibilities of modification that 3dMax has, with the use of the scripting language. Many more limitations, as stated before, need to be solved in order 3dMax to become the base where kinetic structures and their movement could be simulated. This seems to be possible to be achieved if this is considered as the start of a future work.

Problems like the automation in different functions or changing common parameters for a selection of many elements of the same category could be as well solved with the use of scripting language. A new toolbar could be created that could imply instantly all the desirable changes or functions on the elements causing the automation in both cases, which in the current state of 3dMax has to be performed separately for every element.

Based on the same theory as the toolbars created for the spatial arrangement of the copies, this also could have the possibility to select the desirable element or category of elements. Then with a selection among its functions or by changing the value of parameters and by pushing a button the transformation should be implied. For example, if the same dimension modification for a set of elements is needed, the parameters that are known and already used for every element as shown in **Figure 5-2** could be included into the toolbar and after their alteration, the result could be implied with the push of a button in the selected elements.

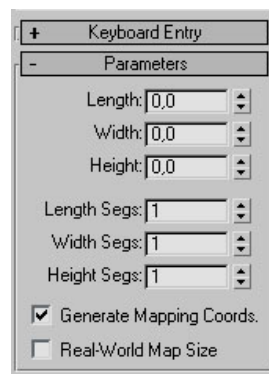


Figure 5-2. The parameters of the elements dimensions

Regarding the connection of elements or primitive objects between them, which is now restricted at the position of the pivot point, again a solution could be found with the help of scripting. In the case study of Alcoy Community Center, where the problem first appeared, the solution was found with a creation of a skeleton structure to resemble the bent rod. Upon this consideration, for similar cases, a function or a toolbar could be created which by giving the path or the shape and the number and dimension of intermediate bones, a series of hierarchical bone structure could be created to substitute a single object or element. This way, many objects will be possible to be attached to a single element into different positions of its surface, according to the needs of the structure.

Of course, the last two suggestions are only speculations of how the limitations recognized in 3dMax could be overcome. In reality, the solution can differ or can be done with another procedure. For some of the problems stated before, it cannot be predicted how the solution could be formed and therefore no suggestions can be made. These are more complex matters (such as manipulating the animation into 2 or more axes at the same time with automation or not, when also the complex system of elements belong to different orientation and axes) that include a variety of functions performed at the same time. However, in both cases, further research and investigation are needed in order to find a real solution..

6 CONCLUSION

Searching and evaluating a subject such as Kinetic Architecture and especially a specific area of it, is not an easy issue. The lack of proper documentation but at the same time the new inventions and researches that are performed and are constantly being presented, make difficult the overall view and the up today level of knowledge of this theme to be kept. However, this means that kinetic architecture is positioned in the middle of the interest and that it is a promising area in the field of construction.

Investing, therefore, in further knowledge over kinetic structures with the creation of a software is nothing but a gain. And that because not only this investment concern a developing issue, which is kinetic architecture, but also relates to a subject which in the future will be included even more than today in our daily life, the use of computer. Understanding, that until now, these two areas have not been combined, an opportunity is being created for the fulfilment of this knowledge gap.

The information that this thesis is presenting can lead to this direction and can be considered as the base of information on which this software could be created. An application which will be used as a tool to provide the architect with the ability to create simulation of the kinetic movement based on the calculation of the thermal, solar etc. analysis, can be thought as a useful step for the further development of kinetic architecture. Using the findings of the thesis over the malfunctions of the software and discovering a way to overcome and even further develop them, can be seen as a future work towards the simulation of the movement of kinetic structures into animation programs.

Seeing in the future and in detail the possibilities of this software it is thought that it could include:

- Necessary tools for the detailed construction of the building (column, beams etc. as AutoCAD, Archicad) in 3D
- Necessary tools for the detailed instructions of movement of the elements that will be used in the kinetic architecture
- Necessary tools for the calculation of the different analysis (i.e. Ecotect)
- Library of materials
- Library of elements
- Necessary tools for the detailed construction of the kinetic mechanism

More analytically this application will have the possibility to represent, with architectural design standards any kind of kinetic structure. It will include all the necessary functions for the implementation of the desirable movement and the possibility of creating a simulation of the movement based on parameters of luminance, temperature, etc for a day or throughout a year that the user will input to the program. The result of the simulation will be associated also with the kinds of material that the user will chose from the existing library, as they will include specific characteristics according to their type, e.g. for a fabric there will be the parameter of elasticity included into

the material definition and therefore recognized by the software. The library of elements will include different structural parts which have resulted after decomposition of the kinetic structures used in the case studies to the smallest parts from which they have been formed, like beams, columns etc. By creating a library with the simplest elements it will be easier for a user to create any kind of kinetic structure from the prefabricated parts of structure.

Truly kinetic architecture cannot be seen as anything else but the future of architecture. The mobility, the transformation and the transportation that it shows cannot be compared with any other kind of architecture where stability and permanency is their characteristic. In the future inhabitancy, where even the scenarios of other planets are included, kinetic architecture can only be in the service of the future human. And in addition to that, a tool that will be used for the expression of this architecture cannot be thought but essential.

REFERENCES

Books

Foley, J. and Van Dam, A., "Computer Graphics", Reading, Mass.: Addison Wesley 1990

García, J. de Jalón and Bayo, E., "Kinematic and Dynamic Simulation of Multibody Systems. The Real-Time Challenge", Springer-Verlag, New-York 1994

Kronenburg, Robert: Houses in motion: the genesis, history and development of the portable building. St. Martin's Press 1995

Kronenburg, Robert: Portable Architecture, Architectural Press, Boston 1996

Kronenburg, Robert, "Transportable environments: theory, context, design, and technology: papers from the International Conference on Portable Architecture", London, 1997, E & FN Spon, London 1998

Topham Sean, "Move house", Prestel publishing, Munich- Berlin-London-New York 2004

Tzonis Alexander, "Santiago Calatrava, The Poetics of Movement", Thames & Hudson Ltd, London 1999

Zuk, William, "Kinetic Architecture", Van Nostrand Reinhold, New York 1970

Web Pages

<http://kdg.mit.edu/Matrix/matrix.html>

<http://www.robotecture.com/kdg/>

<http://www.adifitri.com/kinetic/kine03a.html>

<http://www.jvoegele.com/software/langcomp.html>

http://en.wikipedia.org/wiki/Comparison_of_programming_languages

<http://merd.sourceforge.net/pixel/language-study/scripting-language/>

http://wiki.cgsociety.org/index.php/Comparison_of_3d_tools

http://wiki.cgsociety.org/index.php/3ds_Max

<http://wiki.cgsociety.org/index.php/MAXScript>

<http://wiki.cgsociety.org/index.php/Blender>

http://wiki.cgsociety.org/index.php/Cinema_4d

<http://wiki.cgsociety.org/index.php/Houdini>

<http://wiki.cgsociety.org/index.php/Maya>

<http://wiki.cgsociety.org/index.php/Mel>

<http://csl.sourceforge.net/csl.html>
http://msdn2.microsoft.com/en-us/library/ms974627.aspx#vb_topic4
http://msdn2.microsoft.com/en-us/library/ms974627.aspx#vb_topic4
<http://filext.com/file-extension/obj>
<http://kdg.mit.edu/Projects/p04.html>
<http://contractorsglossary.com/>
http://www.cement.org/tech/faq_joints.asp
<http://www.answers.com/topic/joint?cat=biz-fin>

IMAGE SOURCES

Picture	Sources
Picture 1-1	http://kdg.mit.edu/Projects/p04.html
Picture 1-2	A. Tzonis
Picture 1-3	A. Tzonis
Picture 1-4	A. Tzonis
Picture 1-5	A. Tzonis
Picture 3-1	Maya Help
Picture 4-1	Snapshot of the software
Picture 4-2	Snapshot of the animation software
Picture 4-3	Snapshot of the animation software
Picture 4-4	Snapshot of the animation software
Picture 4-5	Snapshot of the animation software
Picture 4-6	A. Tzonis
Picture 4-7	A. Tzonis
Picture 4-8	A. Tzonis
Picture 4-9	Snapshot of the animation software
Picture 4-10	Snapshot of the animation software
Picture 4-11	A. Tzonis
Picture 4-12	A. Tzonis
Picture 4-13	Snapshot of the animation software
Picture 4-14	Snapshot of the animation software
Picture 4-15	Snapshot of the animation software
Picture 4-16	Snapshot of the animation software
Picture 4-17	Snapshot of the animation software
Picture 4-18	Snapshot of the animation software
Picture 4-19	http://kdg.mit.edu/Projects/p04.html
Picture 4-20	V.Sedlak
Picture 4-21	V.Sedlak
Picture 4-22	Snapshot of the animation software
Picture 4-23	Snapshot of the animation software
Picture 4-24	Snapshot of the animation software
Picture 4-25	Snapshot of the animation software
Picture 4-26	Snapshot of the animation software

Picture 4-27	Snapshot of the animation software
Picture 4-28	Snapshot of the animation software
Picture 5-1	Snapshot of the animation software
Picture 5-2	Snapshot of the animation software
Picture GT-1	Maya Help
Picture GT-2	Maya Help
Picture AB-1	http://upload.wikipedia.org/wiki/en/9/92/3dsmax8Screenshot.jpg
Picture AB-2	http://upload.wikipedia.org/wiki/commons/e/ed/Blender_node_scren_242a.jpg
Picture AB-3	http://upload.wikipedia.org/wiki/en/3/3b/C4d_r10-2.jpg
Picture AB-4	
Picture AB-5	http://upload.wikipedia.org/wiki/en/b/ba/LW9_modeler.jpg
Picture AB-6	http://upload.wikipedia.org/wiki/en/6/60/Autodesk_Maya_8.0_win32.png
Picture AB-7	http://upload.wikipedia.org/wiki/en/8/89/SoftimageXS14_2Screenshot.png

GLOSSARY

Note: the following term definitions are a selection of these sources:

3Ds Max User Manual

Maya Help

<http://www.answers.com/topic/joint?cat=biz-fin>

Articulation

In animation, the state of a skeleton's joints, including position, rotation, and scaling.

Bones and Bones System

A Bones system is a jointed, hierarchical linkage of bone objects that can be used to animate other objects or hierarchies. Bones are especially useful for animating character models that have a continuous skin mesh. Bones can be animated with forward or inverse kinematics. For inverse kinematics, bones can use any of the available *IK solvers*, or through *Interactive* or *Applied IK*.

Bones are renderable objects. They have several parameters, such as taper and fins, that can be used to define the shape the bone represents. The fins make it easier to see how the bone is rotating.

For animation, it is very important that you understand the structure of a bone object. The bone's geometry is distinct from its link. Each link has a pivot point at its base. The bone can rotate about this pivot point. When you move a child bone, you are really rotating its parent bone.

Constraints

''Constraints'' enable you to constrain the position, orientation, or scale of an object to other objects. Further, with constraints you can impose specific limits on objects and automate animation processes.

Deformers

''deformers'' are high-level tools that you can use to manipulate (when modelling) or drive (when animating) the low-level components of a target geometry. In other software packages, the terms modifiers and space warps are used to refer to deformers.

Element (software)

An element is one of two or more individual mesh objects (that is, groups of contiguous faces) grouped together into one larger object. For example, if you attach one box to another, you create one mesh object from the two boxes. Each box is now an *element* of the object. Any function you perform on that object affects all its elements. However, you can manipulate the

elements independently at the Element sub-object level.

ELEMENT (construction)

A component part.

Forward Kinematics

Forward Kinematics is an animation method that involves moving each joint without the restriction of an expected final position. Thus, the 'goal' is to move a joint (or series of joints) as desired, and the final pose is a consequence of those movements. Forward Kinematics is often used for finely-tuned joint movement (such as hands & fingers), as it allows for more complete control over posing.

Joints (software)

Joints control the rotation and position of an object with respect to its parent.

Any object has a maximum of two joint-type rollouts: One rollout contains settings to control the object's position, and the other controls the object's rotation. There can be many different types of positional and rotational joints. Which joint parameters are available is determined by the type of IK solver assigned to an object. HI Solvers, for instance, are controlled with a preferred angle setting found in the Rotational Joint parameters. HD Solvers have additional parameters for spring back, precedence, and damping, not found in the HI Solver.

Any hierarchy of object or bone systems can have its joint limits defined.

Anatomical joint types

Various joint attributes determine how joints behave. By adjusting the attributes of a joint, you can limit how far a joint can rotate or restrict what planes it can rotate about. For example, to create a joint chain that moves and rotates like the human neck, you would set the degrees of freedom for the joints so that they rotate about only 2 axes and then you would limit each joints range of rotation.

Ball joint

A ball joint is a joint that can rotate about all three of its local axes. For example, the human shoulder is a ball joint.

Universal joint

A universal joint is a joint that can rotate about only two of its local axes. The human wrist is a good example of a universal joint, though the wrist has limitations on the extent it can rotate.

Hinge joint

A hinge joint is a joint that can rotate about only one of its local axes. For example, the human knee is a hinge joint.

JOINTS (construction)

The surface at which two or more mechanical or structural components are united. Whenever parts of a machine or structure are brought together and

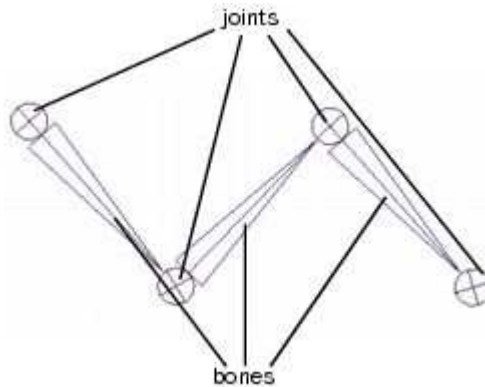
fastened into position, a joint is formed.

Mechanical joints can be fabricated by a great variety of methods, but all can be classified into two general types, temporary (screw, snap, or clamp, for example), and permanent (brazed, welded, or riveted, for example).

Joints and Bones

Joints are the building blocks of skeletons and their points of articulation. Also, joints have no shape and therefore can not be rendered. Each joint can have one or more bone attached to it, and more than one child joint. Joints let you transform a skeleton when posing and animating a bound model.

Bones do not have nodes, and they do not have a physical or calculable presence in your scene. Bones are only visual cues that illustrate the relationships between joints.



Picture GT-1. The bones and joints system

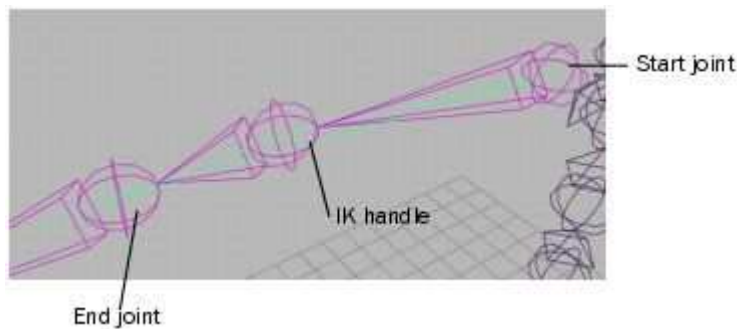
IK Chain

In animation, a joint chain that has an IK handle.

IK handle.

In an IK chain, the joint where the IK handle begins is called the *start joint* and the joint where the IK handle ends is called the *end joint*. All the joints between the start and end joints are driven by the IK handle and its solver. In the scene view, the IK handle is drawn as a line from the start and end joints of the IK chain.

The end of the IK handle, which is located at the last joint of the IK chain by default, is called the *end effector*. When you move the IK handle, the IK solver uses the end effector's position and orientation in its calculations to rotate the joints in the IK chain accordingly. The end effector tries to follow the IK handle's position at all times. However, depending on the rotational limits and fully extended length of the IK chain, the end effector might not be able to reach IK handle.



Picture GT-2. Representation of the IK handle and IKchain

IK Solvers

Inverse Kinematics (IK) describes a class of algorithms for animating linkages of rigid bodies based on a set of goals and constraints placed on the linkage. An IK solver is a mathematical procedure for finding a set of rotations and offsets for the links in order to satisfy the goals and constraints.

Solvers can be tailored for different types of linkages or to behave a certain way, such as minimizing the motion of certain joints or keeping certain joint angles between certain ranges.

An IK solver creates an inverse kinematic solution to rotate and position links in a chain. It applies an IK Controller to govern the transforms of the children in a linkage. You can apply an IK solver to any hierarchy of objects. You apply an IK solver to a hierarchy or part of a hierarchy.

Each type of IK solver has its own behavior and workflow, as well as its own specialized controls and tools that display in the Hierarchy and Motion panels. IK solvers are plug-ins, so programmers can expand the software's capabilities by customizing or writing their own IK solvers.

Inverse Kinematics

The reverse of Forward Kinematics, Inverse Kinematics is a method that involves defining a final pose, and generating joint movement as needed to reach that pose. Thus, the 'goal' is for all joints to be in a final pose, and the individual joint movements are a consequence of getting to that final pose. Joints must have carefully defined limits to their possible motion for Inverse Kinematics to work well, or the joints can end up 'flopping' before reaching the goal pose. Inverse Kinematics is often used for large limb movement (such as walking, reaching, etc.).

Kinematic Chain

Inverse kinematics calculates the position and orientation of objects in a kinematic chain.

The kinematic chain is defined as a single branch of the hierarchy that starts with a selected child object and continues up through its ancestors until it reaches the base of the chain. The base of the chain is either the root of the entire hierarchy or an object that you specify as a terminator for the chain.

Keyframes/Keys

During an animation, on the time slider you create keyframes in order to represent the objects' or the characters' pose in the specific moment.

Keyframes record the beginning and end of each transformation of an object or element in the scene. The values at these keyframes are called *keys*.

The red boxes indicate keyframes, the dotted line shows the interpolated trajectory.

Meshes

A mesh is the “skin” geometry that *Physique* deforms. Physique will work on any point-based object, including geometric primitives, editable meshes, patch-based objects, NURBS, and *FFD space warps*. For NURBS and FFDs, Physique deforms the control points (control vertices), which in turn deform the model.

Modifiers

Modifiers, as the name implies, modify an object's geometrical structure, deforming it in some way. Modifiers make changes in the geometry that stay in effect until you adjust or delete the modifier.

Object

"Object" means an object in the scene, such as primitive geometry like boxes and spheres, more complex geometry such as Booleans, and so on. Geometric objects are renderable. A scene can also contain non-renderable objects such as lights, cameras, helpers, and space warps.

Pivot Point

The transform center, or pivot point, is the spot about which a rotation takes place, or to and from which a scale occurs.

All objects have a pivot point. You can think of the pivot point as representing an object's local center and local coordinate system.

The pivot point of an object is used for a number of purposes:

- As the center for rotation and scaling when the Pivot Point transform center is selected.
- As the default location of a modifier center.
- As the transform origin for linked children.
- As the joint location for IK.

You can display and adjust the position and orientation of an object's pivot point at any time using the Pivot functions in the Hierarchy command panel. Adjusting an object's pivot has no effect on any children linked to that object.

Primitive Object

Primitives are three-dimensional geometric shapes. The primitive shapes include spheres, cubes, cylinders, cones, planes, and many others. You can modify the attributes of basic primitives to make them more or less complex. You can also split, extrude, merge, or delete the various components on the primitive's polygon mesh in order to modify the primitive's shape. Many 3D modelers begin with polygon primitives as a basic starting point for their models. This technique is referred to as primitive-up modeling.

Script

A sequence of instructions used to automate a task. Scripts are typically text files containing coded instructions for a particular application.

Skeleton

The skeleton to which you attach a skin using Physique can be a hierarchy, bones in a hierarchy, bones not in a hierarchy and splines. Physique deforms the skin based on the relative position of the *bone* or links in the hierarchy. Specifically, it uses the length of each link and the angle between two connected links; it can also use the scale of a link.

The skeleton hierarchy can also be a object that defines a behavior as well as a hierarchy. Three kinds of objects that are especially useful with Physique:

- Biped is provided by character studio.
- Bones are a standard Systems object provided with 3ds Max.
- Splines can be used rather than a "bones" hierarchy.

Skeletons are hierarchical, articulated structures that let you pose and animate bound models. A skeleton provides a deformable model with the same underlying structure as the human skeleton gives the human body.

Just like in the human body, the location of joints and the number of joints you add to a skeleton determine how the skeleton's bound model or 'body' moves. When you bind a model to a skeleton, it is called skinning.

Skinning

"Skinning" is the process of setting up a character's model so that it can be deformed by a skeleton. You skin a model by binding a skeleton to the model. You can bind a model to a skeleton by a variety of skinning methods, including smooth skinning and rigid skinning. Smooth skinning and rigid skinning are direct skinning methods. You can also use indirect skinning methods, which combine the use of lattice or wrap deformers with either smooth or rigid skinning.

Skin

A mesh deformed by a skeletal structure is called a *skin*.

Specifically, a skin can be:

- An editable mesh or editable poly object. This is the most commonly used type of object for Physique. Often, it has been collapsed from an object with modifiers, or a compound object.
-

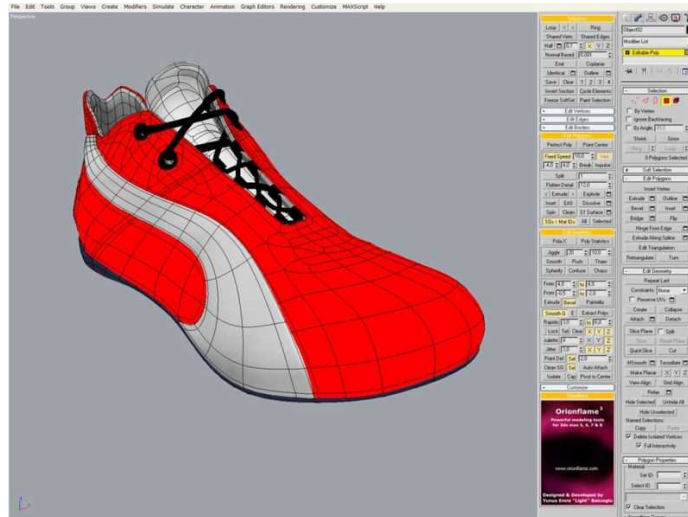
- An uncollapsed object with modifiers or a compound object.
- A parametric geometry primitive such as a cylinder.
- Geometric primitives are useful mainly for simple applications of Physique; for example, a cylinder with two bone links to depict an arm.
- A patch object.
- A spline or text shape.
- A NURBS object.
- A Free From Deformation (FFD) modifier.

A mesh object you import from another application such as AutoCAD®.

Appendix A: Description of the software

Note: the following information regarding the animation software is a selection from different internet sources given in References

3ds Max



Picture AB-1. Screenshot of 3DS Max (Internet)

3ds Max, previously known as **3D Studio MAX** and **3dsmax** and also called simply "Max" by its users, is a modelling, animation and rendering package developed initially by the Yost Group for Autodesk and further developed by Autodesk and its various Multimedia divisions throughout the last decade : Autodesk Multimedia Division; Kinetix, a division of Autodesk; Discreet, a division of Autodesk and recently Autodesk Media and Entertainment (AMED).

3ds Max Dynamics

3ds Max includes a physics engine, called reactor, originally created by Havok. Reactor can simulate rigid bodies, soft bodies, cloth, gravity, and other forces. Like many physics engines, reactor uses a simplified convex hull, but can be customized to use all vertices, at a cost in processing time.

3ds Max Animation

Every property of every object in 3ds Max can be exposed by the developer as animatable - there are few properties that do not make sense to be animatable like the "Generate Texture Coordinates" checkbox in geometry primitives, but these are the exception. In general, almost every property can be animated. 3ds Max uses a special object class known as animation controller to manage animation data (similar to Houdini's CHOPs). Animation controllers are plug-ins that can be assigned to any animatable track and can either manage key frames in various ways (like Bezier, Linear, Boolean controllers), generate animation data procedurally (Noise, Expression, Script controllers) or read data from external sources (Wave, Motion Capture etc.)

Controllers can be assigned to tracks by the user if needed or will be assigned

by the system automatically either at object creation or at first animation change using a list of default controllers. For example, when a Box is created, its Position, Rotation and Scale properties will have default controllers already assigned since it is assumed that these properties will be most likely animated. At the same time, the Box base object's properties like Width, Length, Height etc., while animatable, will have no default controllers assigned until the user tries to animate them - by default, they will be assigned a Bezier Float controller.

Controllers can also be stacked using compound controllers - for example, a Position_XYZ controller has 3 sub-controllers managing the data of the X, Y and Z position of the object; the Float List controller can have an arbitrary number of Float sub-controllers which can be blended using Weights in various ways and so on. This means that 3ds Max allows multiple controllers to be assigned to the same track and mix animation data on the fly without any intermediate null objects - for example, to make an already animated camera shake a bit, the user can assign a List controller on top of the existing Position controller and assign a Noise controller to generate additional random shakes on top of the key frame data.

All controllers in 3ds Max can be assigned optional Ease and Multiplier curves to manipulate the final value over time. An Ease curve changes the time the final value is taken from. The Multiplier curve multiplies the controller's value with the multiplier's value. These curves can also be stacked, allowing Eases of Eases and so on.

3ds Max provides two main animation modes - Auto Key and Set Key. The Autokey mode was the original and only option in the earlier versions of the software up until 3dsmax 5 and was initially called simply "Animate". When Auto Key mode is on, any changes to animatable properties cause a new animation key to be generated automatically. If there are no keys in the track and the time is different than 0, a default key with the original track value will be generated on frame 0. When using the Set Key mode, changes to animatable tracks will be recorded but not applied until the user presses the Key button. Keyable tracks can be specified using filters and flags in the Track View. Keys can also be set manually outside of the animation modes - to keyframe position, rotation and/or scale only, the user can move the time slider to the desired frame and right-click the time slider to open a dialog. Right-clicking on one frame, then holding and sliding to another frame will copy the values from the initial time to keyframes on the new time. Spinner values of animatable properties displayed in the command panel, material editor etc. can be keyframed manually by holding down the SHIFT key and right-clicking the spinner's arrows. Keyframes will be displayed as red brackets around an animated spinner.

3ds Max provides multiple ways to drive values with other values: The Wire Controller is a simplified version of a Script controller where the user can select between one-directional or bi-directional wiring and provide MAXScript expressions for both sides. The Reaction controller (similar to Set Driven Key in Maya) allows the user to set the values of the driven controller based on values of the driving controller, while providing curve controls to adjust the interpolation between these states. The Expression controller allows the user to define variables which can be connected to object tracks and calculate a new value using these variables and a set of useful mathematical functions (including an IF statement). Finally, the Script controller can use the full power of MAXScript to calculate the final value of the animated track. In addition, due to the fact that the animation of each parameter is managed by a dedicated controller object, controllers can be instanced between tracks of compatible type, for example the radius of a sphere could share the same controller as the height of a box, thus linking the

two in an absolute fashion. Using instanced controllers inside List controllers allows this absolute relationship to be mixed with other controllers for varying results. Instanced controllers have been available since Release 1.0 and are the most stable way to connect two parameters' values in an absolute way. The instancing can be performed inside of Track View or by right-clicking value spinners in the UI and using the context menus to Copy and Paste Animation.

Scripting

3ds Max provides a powerful proprietary scripting language which can be used to control the application from the command line, create scripted functions, tools, User Interfaces, plug-ins, read and write binary and ASCII data and bitmaps, drive animations procedurally via scripted controllers and lots more.

Blender



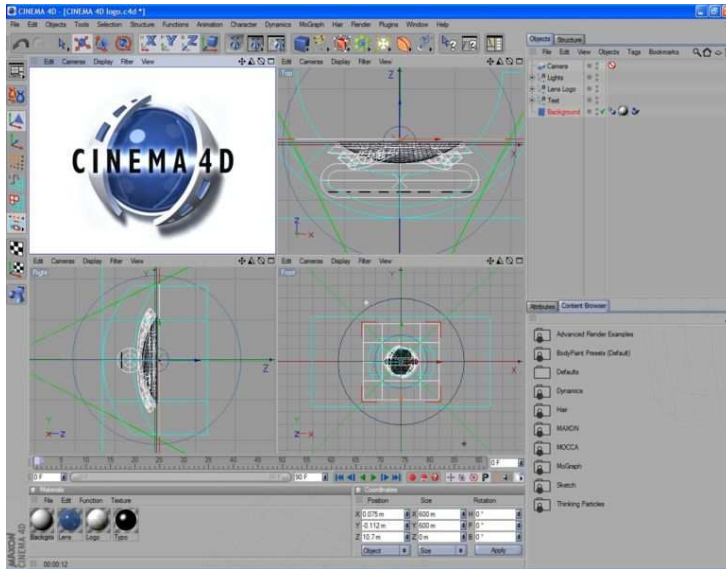
Picture AB-2. Screenshot of Blender(Internet)

Blender is a cross-platform, open source 3D modelling and animation suite developed and maintained by the Blender Foundation. Blender features many powerful features normally associated with commercial 3D applications, which is unsurprising considering it was originally developed as an in-house animation tool by Dutch company NaN (Not a Number). Blender has a huge user community and is arguably among the best free software of its kind.

Scripting

Blender has python based scripting. Blender is also 'open source' which means that coders can add features directly to the core code base or even rewrite core pieces of functionality, or expose additional parts of the internal API to the python scripting base.

Cinema 4d



Picture AB-3. Screenshot of Cinema 4D(Internet)

CINEMA 4D is a popular 3D application amongst matte painters in film largely due to the Bodypaint 3D module and is equally popular amongst motion graphics artists thanks to an excellent integration with compositing application pipelines coupled with a very artist-friendly and customizable workflow and interface. CINEMA 4D is made by Maxon Computer GmbH in Friedrichsdorf, Germany and as it was originally created for the Amiga, it was later ported to Apple Macintosh and Windows systems.

The application relies on widely used computer 3D technology which works by creating groups of "points" (known as [vertices](#)) which form surfaces when connected. The illusion of three dimensions is created by modeling objects out of multiple surfaces. Still pictures, movies and game environments (among other things) can be created with this technique.

CINEMA 4D has its own runtime scripting and programming language C.O.F.F.E.E. which is rather similar in form to JScript and an advanced modern C++ API with extensive SDK to develop platform independent plug-ins. The free C++ SDK for creating plug-ins comes with the software (including the Demo version) and is also available on Plugin Cafe. Four different packages have been released by MAXON: the core CINEMA 4D application, the XL-Bundle (including NET Render [3 lic], PyroCluster, Advanced Render, MOCCA and Thinking Particles), the Studio Bundle, which includes all modules, and the Production Bundle, which comes with a service contract, Linux version, 32-bit painting support in BodyPaint 3D and a number of other high-end studio-specific features. This version is only available upon request.

Initially, CINEMA 4D was developed for Amiga computers in the early nineties but has since been released for Mac OS X, Windows, and most recently via the Production Bundle the Linux OS.

As well as the core application (for modeling, texturing, lighting and

rendering), CINEMA 4D also has several add-on programs (modules) available that expand its capabilities. These programs include:

- Advanced Render (global illumination/HDRI, caustics, ambient occlusion and sky simulation)
- BodyPaint 3D (direct painting on UVW meshes)
- Dynamics (for simulating soft body and rigid body dynamics)
- HAIR (simulates hair, fur, grass, etc.)
- MOCCA (character animation and cloth simulation)
- MoGraph (Motion Graphics procedural modeling and animation toolset)
- NET Render (to render animations over a TCP/IP network in render farms)
- PyroCluster (simulation of smoke and fire effects)
- Sketch & Toon (tools for cel shading, cartoons and technical drawings)
- Thinking Particles (enhanced particle system based on nodes)

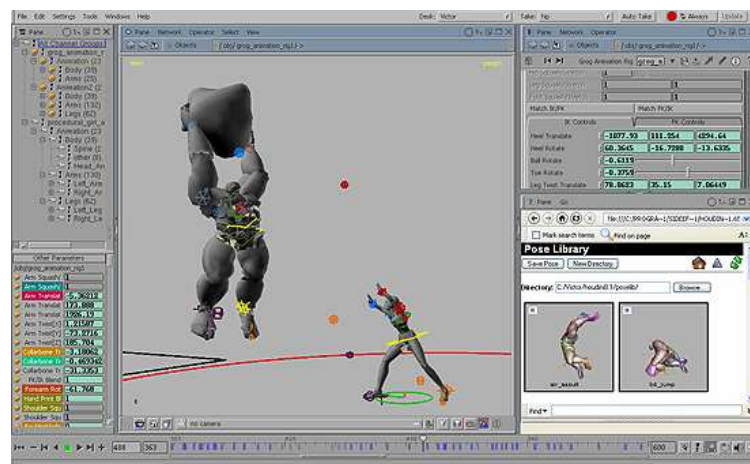
As-of 2006, Four high-end alternate rendering engines are currently available for CINEMA 4D as plug-ins:

- finalRender stage 2.0SP2 from Cebas Computer GmbH
- Maxwell Render from Next Limit Technologies
- Renderman Connection from Maxon Computers GMBH
- Indigo Renderer from Radiance

Animation Advantages

Bringing your 3D objects to life through animation can be the most rewarding and challenging aspect of any 3D project. CINEMA 4D provides the animation tools to achieve almost any animation effect. An extensive timeline allows easy organization and manipulation of animation tracks, and all parameters can be automatically keyframed as you make changes.

Houdini



Picture AB-4. Screenshot of Houdini(Internet)

Houdini is a high-end 3D animation package developed by Side Effects Software. It is often preferred over competing packages, such as Maya or XSI, for pure visual effects tasks, as opposed to character animation. Like Maya, Houdini is an open-environment and uses the popular Tcl/Tk scripting language and toolkit (as opposed to the proprietary MEL used in Maya). Houdini also has its own CShell-like scripting language, Hscript. However, any major scripting languages which support socket communication can interface with Houdini. Houdini's primary distinction from other packages is that it has been designed as a purely procedural environment.

Digital assets are generally constructed by connecting sequences of operations (or **OPs**). This proceduralism has several advantages: it allows users to construct highly detailed geometric or organic objects in comparatively very few steps compared to other packages; it enables and encourages non-linear development; and new operators can be created in terms of existing operators, a flexible alternative to non-procedural scripting often relied on in other packages for customisation. Houdini uses this procedural paradigm throughout: for textures, shaders, particles, "channel data" (data used to drive animation), rendering and compositing.

Houdini's operator-based structure is divided into several main groups:

SOPs - surface operators.

POPs - particle operators.

CHOPs - channel operators.

COPs - composite operators.

DOPs - dynamic operators.

ROPs - render operators.

VOPs - VEX operators

Operators are connected together in networks. Data flows through, manipulated by each operator in turn. This data could represent 3D geometry, bitmap images, particles, dynamics, shader algorithms, animation, audio, or a combination of these. The paradigm is similar to that employed in node-based compositors such as Shake or Nuke.

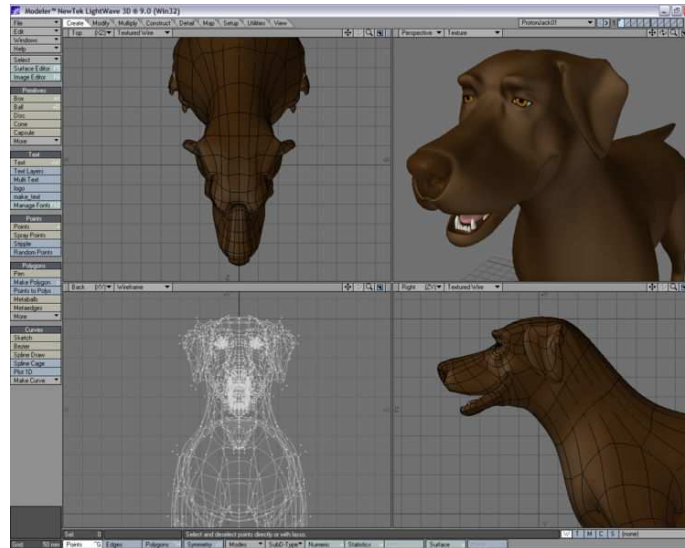
Complex networks can be grouped into a single meta-operator nodes, allowing users to create their own sophisticated tools without the need for programming. These nodes can be regarded as a visual programming language made more interactive and accessible to artists.

Houdini is unique in its very rich set of tools, which are usually implemented as operators. This has lead to a higher learning curve than other comparable tools. Also unique to Houdini is the range of I/O **OPs** available to animators, including MIDI devices, raw files or TCP connections, audio devices (including built-in phoneme and pitch detection), mouse cursor position, and so on. Of particular note is Houdini's unique ability to work with audio, including sound and music synthesis and spatial 3D sound processing tools. These operators exist in the context called "CHOPs" for which Side Effects won a Technical Achievement Academy Award in 2002.

Historically, Houdini's main strength has been its particle animation system, and its major weakness its unintuitive animation tools. As a result, its use was relegated to special effects. However, in more recent versions these tools have been vastly improved and it has been used in the various feature animation productions, such as *The Wild*, a major Disney feature film as well

as the feature animation Ant Bully.

LightWave



Picture AB-5. Screenshot of LightWave 3D(Internet)

is a computer graphics program for 3D modeling, animating and rendering. The original program was shipped with the Video Toaster, an expansion card for the Commodore Amiga 2000 computer. It was later sold as a standalone program for AmigaOS until the mid 1990s, when it was ported to the Windows, Mac OS X and Sgi IRIX operating systems. The rendering engine, ScreamerNet has also been ported to Linux platforms.

LightWave has long been known for its excellent rendering abilities and unusual user interface (for example, icons are not used; instead functions are all given descriptive titles).

LightWave was one of the first high profile industry standard 3D packages featuring a built-in radiosity render engine, complete with a complex light calculation model for support of caustics.

Some functions within LightWave are multi-threaded, which means that those components can simultaneously use as many as eight processors in the same machine when performing complex calculations.

Programmers can expand LightWave's capabilities using an included SDK and also a special scripting language called LScript. This SDK is based on the powerful C language and almost anything can be created, from a custom shader to a different scene format exporter. LightWave itself includes dozens of free plugins and many more can be obtained from different developers around the globe.

Unique Features

Modeler and Layout

Unlike other 3D packages, LightWave is composed of two separate programs: Modeler and Layout. Each program is specifically designed to provide the most efficient workspace for specific tasks. Appropriately,

Modeler provides tools for creating 3-dimensional objects, whereas Layout provides a workspace for lighting, animating, and rendering. When the two programs are running simultaneously, a feature called the Hub can be used to automatically synchronize data between the two applications via TCP/IP protocol.

Layout contains the LightWave renderer which provides the user with several options including Ray Tracing options, multithreading, and output parameters. This differs from most 3D computer graphics packages such as the popular Maya and 3D Studio Max, whereas they contain the renderer and the modeler all in the main program. This attributes plenty of versatility to LightWave as the user has several specific options they can customise via either program. It creates less confusion as well, and can provide easier methods of organisation.

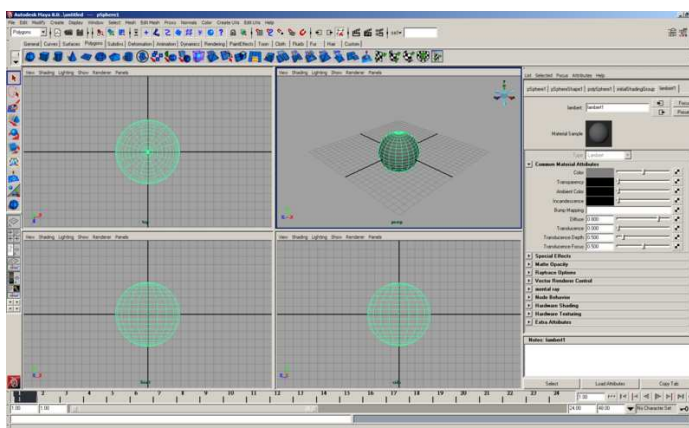
Integration

A long-standing debate in the LightWave user community has consisted of whether or not to integrate Modeler and Layout into a single program. In response to this, NewTek has begun an integration process by including several basic modeling tools with Layout. The initial results of this integration process can be seen in the latest version of LightWave.

Maya

Maya is a high-end 3D computer graphics and 3D modelling software package, originally by Alias Systems Corporation but now owned by Autodesk under its Media and Entertainment division. Autodesk acquired the software in October 2005 upon purchasing Alias. It is often used in the film and TV industry, as well as for computer and video games.

Maya, used in many films today, is named for the Sanskrit word meaning *illusion* and is a popular, proprietary integrated 3D software suite, evolved from Alias PowerAnimator. Maya comes in two main versions, Maya Complete (the less powerful package) and Maya Unlimited. Maya Unlimited is now priced similarly to other proprietary 3D programs, but used to be considerably more expensive. Maya Personal Learning Edition (PLE) is available for non-commercial use, and is available at no cost. Images rendered with Maya PLE are watermarked.



Picture AB-6. Screenshot of Maya(Internet)

The most important feature of Maya is its openness to third-party software, which can strip Maya completely of its standard appearance and, using only the Maya kernel, can transform it into a highly customized version of the software. Apart from its intrinsic power and flexibility, this feature in itself made Maya appealing to large studios which tend to write quite a lot of custom code for their productions using the provided software development kit.

Maya also features a powerful, interpreted, cross-platform scripting language called *Maya Embedded Language* (MEL), which is similar to Tcl. It is not only provided as a scripting language, but as means to customize Maya's core functionality (much of Maya's environment and tools are written in the language). Additionally, user interactions are implemented and recorded as MEL scripting code which users can view and drag onto a toolbar to create new 'macro' tools instantly. This provides animators with the power to add functionality to Maya without experience in C or C++ programming and compilers, though that option is provided with the software development kit.

Project files, including all geometry and animation data, are stored as sequences of MEL operations which can be optionally saved as a 'human readable' file (.ma, for Maya ASCII), editable in any text editor outside of the Maya environment and allows for a tremendous level of flexibility when working with external tools. It can also be edited to allow the file to be opened on previous versions of the software.

Hotbox provides instant access to the majority of features in Maya via a large menu that surrounds the mouse pointer at any time when a user holds down the space bar.

Nonlinear Animation

After animating a character with key frames or motion capture, you can collect its animation data into a single, editable sequence. This animation sequence is called an animation clip.

In Maya, there are two types of clip: source clips and regular clips. Maya preserves and protects a character's original animation curves by storing them in source clips. You do not use source clips to animate your characters. Instead, you use copies or instances of source clips called regular clips to animate your characters nonlinearly. See Animation clips.

Moving, manipulating, and blending regular clips to produce a smooth series of motions for a character is the basis of nonlinear animation. The tool with which you manage all these aspects of a character's nonlinear animation is the Trax Editor.

Path Animation

A path animation controls the position and rotation of an object along a curve. A NURBS curve cannot be designated as a motion path. An object must first be attached to the curve for it to become a path curve. See To create a motion path by attaching an object to a curve. You can also generate motion paths by animating objects using motion path keys. See To create a motion path using motion path keys.

Skeletons

Skeletons are hierarchical, articulated structures that let you pose and animate bound models. A skeleton provides a deformable model with the same underlying structure as the human skeleton gives the human body.

Just like in the human body, the location of joints and the number of joints you add to a skeleton determine how the skeleton's bound model or 'body'

moves. When you bind a model to a skeleton, it is called skinning.

Forward Kinematics

Forward Kinematics is an animation method that involves moving each joint without the restriction of an expected final position. Thus, the 'goal' is to move a joint (or series of joints) as desired, and the final pose is a consequence of those movements. Forward Kinematics is often used for finely-tuned joint movement (such as hands & fingers), as it allows for more complete control over posing.

Inverse Kinematics

The reverse of Forward Kinematics, Inverse Kinematics is a method that involves defining a final pose, and generating joint movement as needed to reach that pose. Thus, the 'goal' is for all joints to be in a final pose, and the individual joint movements are a consequence of getting to that final pose. Joints must have carefully defined limits to their possible motion for Inverse Kinematics to work well, or the joints can end up 'flopping' before reaching the goal pose. Inverse Kinematics is often used for large limb movement (such as walking, reaching, etc.).

Full Body IK Solver

When Alias bought Kaydara, Maya got an upgrade, from Kaydara Motion Builder, with a full body IK solver (FBIK Solver) which simulates real body kinematics. The package comes with a biped and a quadruped FBIK samples.

Skinning

''Skinning'' is the process of setting up a character's model so that it can be deformed by a skeleton. You skin a model by binding a skeleton to the model. You can bind a model to a skeleton by a variety of skinning methods, including smooth skinning and rigid skinning. Smooth skinning and rigid skinning are direct skinning methods. You can also use indirect skinning methods, which combine the use of lattice or wrap deformers with either smooth or rigid skinning.

Constraints

''Constraints'' enable you to constrain the position, orientation, or scale of an object to other objects. Further, with constraints you can impose specific limits on objects and automate animation processes.

Character Sets

In Maya, a character set is a node that brings together into a set all the attributes of any collection of objects that you want to animate together. The character set could be anything: a well-armed robot, an automobile, or even some seemingly unrelated collection of objects. Maya enables you to bring together all the attributes together in a character node, so you only have to select one node, the character node, when you want to animate all the various attributes.

Deformers

''deformers'' are high-level tools that you can use to manipulate (when modeling) or drive (when animating) the low-level components of a target geometry. In other software packages, the terms modifiers and space warps are used to refer to what Maya calls deformers. The following are the many types of deformers: Blend Shape deformer, Lattice deformer, Cluster deformer, Nonlinear deformer, Sculpt deformer, Soft Modification deformer, Jiggle deformer, Wire deformer, Wrinkle deformer, Wrap deformer, Point On Curve deformer.

XSI

Softimage|XSI is a high-end three-dimensional (3D) graphics application developed by Softimage, Co., a subsidiary of Avid Technology, Inc., which is used predominantly in the film, gaming and advertising industries for the production of 3D environments and scenes.

Three different versions of the software have been developed by Softimage to cater for the differing needs and budgets of users: Foundation, Essentials, and Advanced.

Additionally a limited and watermarked version of XSI called Softimage|XSI EXP has been made available by Softimage to enable users to learn how to use XSI without purchasing the software. A further two versions of the program have been developed for the game modding community called Softimage|XSI Mod Tool and Softimage|XSI EXP for Half-Life 2. The latter version of XSI was developed in close collaboration with Valve Software to allow the game modding community to create content specifically for Half-Life 2.

OpenXSI

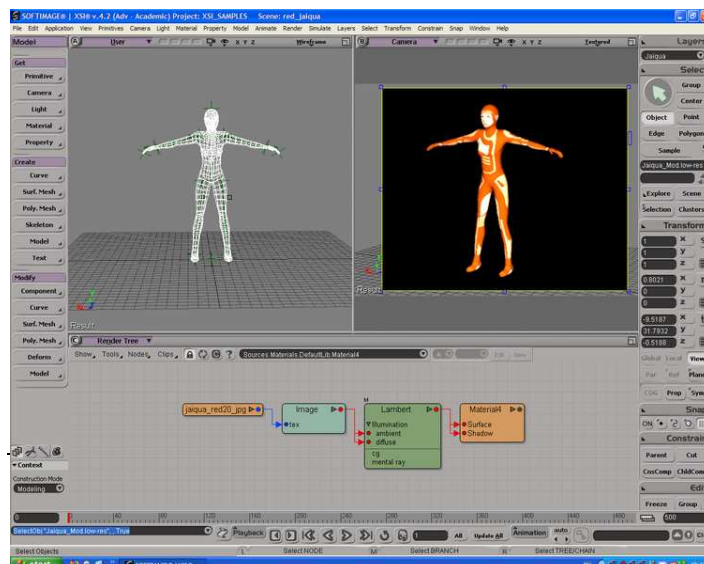
As with all high-end 3D graphics applications a core feature of XSI is its customizability and extensibility. This is provided through a scripting interface which provides access to the core functionality of XSI and enables users to tailor XSI to meet the specific needs of a production environment. While Alias decided to develop their own scripting language called Maya Embedded Language (MEL) for Maya, Softimage incorporated existing scripting languages within XSI. Support is given for JScript, VBScript, Python, PerlScript, ScOps, and OMScripting.

Custom extensions to XSI called plug-ins can also be developed to add new features and tools through the use of a C++ SDK API and the Custom Display Host API can be used to incorporate proprietary applications into the XSI environment.

XSI files can be written and read using the dotXSI file format and File Transfer Kit (FTK).

XSI workforce

A related set of tools grouped under XSI workforce are provided with XSI to enable greater productivity when working in teams:



Picture AB-7. Screenshot of XSI(Internet)

Net View is an integrated web browser which can be used to view documentation and other files while working on a project. Additionally it can be used to display complex HTML-based user interfaces e.g. to invoke scripts that interact with the scene.

XSI performance

Non-modal interaction (meaning that multiple dialog boxes can be opened to modify multiple parameters on a model) enables users to work iteratively and non-destructively. Visual properties can also be set on a per-object basis so that scene display and updates can be optimized while working on a scene.

Rendercore

The core of the rendering system is based on mental ray which has been tightly integrated within XSI. All advanced mental ray features can be rendered in any viewport while interactively working on a model. Distributed rendering is also supported and a web-based render queue management system called BatchServe is provided to manage batch rendering of scenes in XSI. Additionally the plug-in architecture of XSI allows custom plug-in renderers and display engines to be hosted.

Mixer

XSI can also be used for animation. Animatable inputs can be mixed, blended, layered, and edited using the Animation Mixer, and audio-playback features can be used to achieve lip-synching for character models. The Character SDK provided with XSI can be used to customize character creation, animation, and editing.

XSI intuition

The XML-based UI engine provided with XSI can be used to optimize the user's ability to interact with XSI's tools.

XSI fx

Similar to *Houdini*'s earlier integrated compositor, COPs, the FX Tree in XSI provides 2D raster and vector based resolution independent paint tools. It is based on Avid Media Illusion technology.

Xgs

Real-time shading languages such as *Cg* and *HLSL* have been fully incorporated within XSI so that artists can see exactly what the final output will look like without leaving XSI.

Appendix B: Description of scripting languages

Note: the following information regarding the scripting languages is a selection from different internet sources given in References

MAXScript

MAXScript is the scripting language of 3ds Max. It is possible to edit MAXScripts using the built-in MAXScript Editor or alternatively using external text editors. MAXScript allows an arbitrary number of Editor windows to be opened at any given time.

MAXScript provides a MacroRecorder which can generate code automatically based on user interactions with the UI. Selected code can be dragged from the top (pink) area of the MAXScript Listener onto a toolbar to create a new button with this functionality.

C Scripting Language

C Scripting Language (CSL) is an embeddable scripting language with C syntax. A comprehensive set of libraries is included in the base package, and writing your own libraries is possible with an easy API for C programs, as well as a class interface for C++ programs. If you are looking for a compact and powerful scripting engine for your application, CSL might be the choice.

C Scripting Language (CSL) is a powerful and easy programming language available for Windows, OS/2 and Unixish systems. CSL follows the C syntax very closely and programmers used to C, C++ and Java will immediately be familiar with it.

CSL is used like an interpreter: You write the program with your favorite editor and run it directly like any shell script.

More than that, the CSL scripting engine can be integrated into your own applications as a macro language. CSL has 2 programming interfaces: A "C" API for virtually any 32 bit compiler, and C++ class interface for selected compilers.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

At a glance:

- C syntax, very familiar to C, C++ and JAVA programmers
- Sophisticated error handling by exceptions
- Comprehensive standard libraries included (system, strings, maths, files, regular expressions and more)
- Interfaces ORACLE, DB2, MySQL and every ODBC source
- Automate regular tasks or build application benchmarks with the windows control library
- Develop libraries to meet your special requirements using your favorite C/C++ compiler

Embed the compact CSL scripting engine into your applications enhancing them with a powerful macro language

Python

Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains. Python is often compared to Tcl, Perl, Ruby, Scheme or Java. Some of its key distinguishing features include:

- very clear, readable syntax
- strong introspection capabilities
- intuitive object orientation
- natural expression of procedural code
- full modularity, supporting hierarchical packages
- exception-based error handling
- very high level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
- embeddable within applications as a scripting interface

COFFEE

COFFEE is the scripting language used in Cinema 4D. It is syntactically similar to Java and JavaScript, and can be used for scene-specific expressions as well as plugin development. The COFFEE SDK is freely downloadable and available from the official support site.

HScript

Hscript, or Houdini Script, is the scripting language of Side Effects Software's Houdini program.

It allows you to execute a script, which generates the HTML output to include on the page. The scripting syntax is very similar to the well-known ASP syntax and you can use both VBScript and JavaScript as scripting language. To make the transition even smoother from ASP scripting, an object model is available, which partly implements the ASP Server, Request, Response, Session and Application objects.

The HScript control allows you to use ASP syntax scripting to create your HTML contents. You also customize this control by defining a number of arguments, whose value can be changed in the DTC's property page.

The HScript control runs a kind of embedded scripting language that allows you to create truly dynamic HTML contents.

Configuring the control

To define arguments and other properties for the control, you can use the same template file as the HTemplate control uses. It's is however easier to just point to a file with the ASP extension as input, and then use the ASP processing directives to configure the control.

The HScript Object Model

To assist you in creating rich and dynamic HTML, the HScript language has access to several predefined objects that help you create better web solutions. These objects are scaled-down versions of the Server, Request, Response, Session and Application objects already known from ASP. They contain most of the commonly used methods, such as

```
Response.Clear  
Response.Write("text")  
Response.Stop  
Application.Contents("variable")  
Session.Contents("variable")  
Request.QueryString("argument")  
Request.ServerVariables("variable")
```

VEX

VEX (Vector Expression) is one of Houdini's internal languages. It is similar to the RenderMan shading language. Using VEX a user can develop custom SOPs, POPs, shaders, etc. The current implementation of VEX utilizes SIMD style processing.

LScript

The LScript language is a simple and small scripting language based on LISP. It is written in Java and could easily be used to customize applets or applications. It provides a simple extension mechanism which allows adding further capabilities by providing language plugins written in Java (similar to TCL/TK). Due to the nature of LISP (almost everything worth noting is a function), even new control structures could be added easily.

The LScript interpreter does not attempt to emulate large LISP systems like Common Lisp or Scheme. Instead, it provides a small and easy-to-use language which has inherited its major character from LISP like languages. The basic LScript system comes with the following plugins:

The CorePlugin provides basic services and control structures.

The NumberPlugin offers mathematical capabilities.

The StringPlugin provides basic String manipulation features.

The PredicatePlugin offers relational and logical operators.

The ListPlugin provides list manipulation features.

Please note that the current state of the LScript system is unfinished. Actually, it was a two-day hack plus one or two days debugging and performance tuning (thanks to the JProbe profiler from KL Group). There is still no proper documentation (even no inline documentation) and a number of features are not yet implemented (i.e. user-defined functions).

MEL

The Maya Embedded Language is a scripting language used to simplify tasks in Alias' 3D Graphics Software Maya. Most tasks that can be achieved through Maya's GUI can be achieved with MEL, as well as certain tasks that are not available from the GUI. MEL offers a method of speeding up

complicated or repetetitive tasks, as well as allowing users to redistribute a specific set of commands with others that may find it useful.

MEL is syntactically similar to Perl. It provides some memory management and dynamic array-allocation, and offers direct access to functions specific to Maya.

Maya also offers an expression language that is a superset of MEL, and results in nodes that are executed as part of Maya's dependency graph. Expressions are developed with Maya's expression editor, and allow scripts to act while Maya evaluates the scene file, to simulate complex behaviors or perform other useful tasks.

VBScript

VBScript (short form of Visual Basic Scripting Edition) is an Active Scripting language interpreted via Microsoft's Windows Script Host. The language's syntax reflects its pedigree as a variation of Microsoft's Visual Basic programming language. It has initially gained support from Windows administrators seeking an automation tool more powerful than the batch language first developed in the late 1970s.

VBScript is interpreted by a script engine `vbscript.dll`, which can be invoked by ASP engine `asp.dll` in a web environment, `wscript.exe` in a Windows GUI environment, and `cscript.exe` in a command-line environment. When VBScript source code is contained in stand-alone files, they have the file extension `.vbs`.

When employed in Microsoft Internet Explorer, VBScript is very similar in function to JavaScript — it processes code embedded in HTML. VBScript can also be used to create stand-alone HTML applications (file extension `.hta`) which require Internet Explorer 5 or later to run. Web developers may prefer to use JavaScript instead for better compatibility with web browsers other than Internet Explorer.

VBScript is the language used to write some notable e-mail worms, such as ILOVEYOU. Although the language is not inherently insecure, the Scripting Host has little or no security features. The scroll-like icon representing VBS files may lure users with little Windows experience to think that it is a text file. As VBScript has tight integration into Windows, a basic e-mail worm can be written in VBScript in just a few lines of code.

More productive examples of how VBScript can be used is the SYDI project, or in software packaging, where VBScript can be used to automate slipstreaming, and other aspects of the process.

Advantages

- Easy to understand and master. VBScript's biggest asset is that it's Basic, a language designed to be learned quickly. Visual Basic has brought Basic users some of the advantages of more complex languages, while not losing sight of the fact that the language should be easy to understand.
- It's Visual Basic. Visual Basic has been phenomenally successful since its introduction in 1991; there are now over 3 million Visual Basic developers. All the skills they've learned with it instantly apply to VBScript. Also, an enormous community exists to provide training, books, and magazines for learning Visual Basic.

- Flexible. VBScript can be used in a wide variety of applications, and Microsoft is committed to ensuring that wherever script is part of an application, VBScript will be included.

Language features:

- Error handling. VBScript has a subset of the error handling provided by Visual Basic. This includes the error object and on error resume next. Error handling is very important when developing server-side code, since most of the functionality will require access to external COM objects, which could throw errors.
- Formatting. VBScript has the ability to format dates, numbers, and currency built into the language.
- Easier COM Interop. Many COM objects return information in the form of a collection. VBScript has built-in support for iterating through collections
- Standard Event-binding Syntax. Visual Basic developers will immediately recognize the object_event (sub button1_onclick) naming convention for event handlers. VBScript works in exactly the same way, so in any application that supports event binding (such as Internet Explorer, Outlook, and Windows Script Host), you can use this syntax for your event hook-up code. This also means that you can cut and paste existing Visual Basic code into VBScript (assuming you are using features that are in VBScript).

JavaScript

JavaScript is Microsoft's Active Scripting implementation of ECMAScript. JavaScript was first supported in Internet Explorer browser 3.0 released in August 1996. As with any other Active Scripting engine, it is available through Internet Explorer, Windows Script Host, and Active Server Pages. The typical file extension of JavaScript source code files is .js.

The most recent version of JavaScript is JavaScript .NET, which is based on the yet-unfinished version 4 of the ECMAScript standard, and can be compiled for the Microsoft .NET platform. JavaScript adds several new features to ECMAScript, such as optional static type annotations.

In addition to other internal implementation differences, JavaScript uses non-generational mark-and-sweep garbage collection whereas JavaScript (the original implementation of which is the SpiderMonkey engine) uses a generational mark-and-sweep system.

JavaScript was designed as a general-purpose scripting language that would appeal to the many programmers who use C, C++, and Java. This means that it "borrows" features from these languages where appropriate, but is a language in its own right and includes many features not found in C or Java. It's important to note that JavaScript is not just confined to use in the browser; you can use JavaScript in most applications in which you can use VBScript.

Advantages

- Broad reach. JavaScript is pretty much guaranteed to run in

any browser, anywhere. If your page really must work in any browser, this is the language for you.

- Many books to learn JScript
- Similarity to C and Java.

Language features:

- **Dynamic:** JScript was designed as a completely dynamic language; that is, you can effectively redefine your program on the fly. While this has a number of potential disadvantages, it does give you the ultimate flexibility in your scripts. This is particularly useful in DHTML programming, since DHTML allows you to dynamically manipulate the object model. If you really want to drive DHTML, you might want to consider using JScript instead of VBScript.
- **Object oriented:** JScript certainly isn't a traditional class-based, object-oriented language, but it does provide an effective alternative based on prototypes. This allows you to reap the benefits of object orientation without the statically defined nature of classes.
- **Regular expressions:** A main reason why Perl has such a huge following. Regular expressions add the ability to search for expressions in strings. This is exceptionally useful on the server and, increasingly, on the client.
- **Eval:** Provides the ability to immediately evaluate code at runtime. This allows you to dynamically redefine logic dependent at run time. This is especially useful when used in conjunction with Remote Scripting, which works with VBScript as well.

Language features:

- **Dynamic:** JScript was designed as a completely dynamic language; that is, you can effectively redefine your program on the fly. While this has a number of potential disadvantages, it does give you the ultimate flexibility in your scripts. This is particularly useful in DHTML programming, since DHTML allows you to dynamically manipulate the object model. If you really want to drive DHTML, you might want to consider using JScript instead of VBScript.
- **Object oriented:** JScript certainly isn't a traditional class-based, object-oriented language, but it does provide an effective alternative based on prototypes. This allows you to reap the benefits of object orientation without the statically defined nature of classes.
- **Regular expressions:** A main reason why Perl has such a huge following. Regular expressions add the ability to search for expressions in strings. This is exceptionally useful on the server and, increasingly, on the client.
- **Eval:** Provides the ability to immediately evaluate code at runtime. This allows you to dynamically redefine logic

dependent at run time. This is especially useful when used in conjunction with Remote Scripting, which works with VBScript as well.