Curve controlled deformations

Abstract

Presenting a method using a curve guided deformation between two meshes with matching vertex count. This allows control over the intermediate positions of the vertices being deformed. Through this method it is also possible to gain control over the animation timing on a per vertex basis.
In addition several methods has been described that would allow for the setup and use of the system to be quick and easy, while preventing the user from being overwhelmed by the complexity.

A rough implementation has been created for Autodesk/Alias Maya as a proof of concept, showing off the functionality and usability of the methods described.

Introduction

Animated deformation has been used for many years by the computer graphics industry, enabling the creation of realistic characters, special effects and various other productions.

In the area of character animation for example, animated deformations are used extensively on characters to get the required look and movement. This may be from subtle wrinkle changes on a characters forehead, to larger deformations showing muscles, or even full body deformations for more stylised characters.
As an example of an extreme deformation, there could be a need to transform a human characters' head into a completely different shape like the head of a lion, horse or any other creature. Here a variety of techniques are used in order to create the illusion in a convincing way. The artist needs methods to control the surface in a quick and easy way without restrictions on what can be done. Unfortunately the available tools are not always sufficient, or require a significant amount of time to set up, causing frustration for the artist and expense for their employers.

This paper will try to provide a solution that removes some of these restrictions met by the artists when handling the deformations of surfaces.

Related Systems

Blend-shapes.[5]
Blend-shapes are created using a method where the vertices of meshes are linearly interpolated from multiple vertex sets into one set.
It has shown to be a very useful method for facial animations and other such subtle deformations. When using blend shapes it is common to have a selection of different vertex sets and then blend between them . This gives the user the possibility to achive a large variation of deformations based on a relatively small amount of position sets.

A major disadvantage with blend shapes is that in many situations they do not allow for a non-linear change between the position sets. If there is a need for more detailed control over the intermediate deformation, more base shapes would have to be added to the system and then blended consecutively. This would normally be a very time consuming process since the shapes have to be created by modifying other base shapes. At the same time it would be hard for the artist to imagine how the intermediate steps should be in order to get the wanted deformation.
Yet another limitation on blend shapes is that they do not allow for any local control over the timing of the deformation, therefore being limited to one interpolation period for the entire set.

Bone structures with skinning[7].

Another widely used method for deformations is to have an underlying bone structure and then deforming the surface based on the changes in the bone structure. This method can give increased local control by adding more bones to the structure. Full local control though will result in an impractical system where there is a bone for each vertex and no influence to other vertices. Here there is only a possibility for individual animation timings on a per bone basis and therefore normally not on a per vertex basis.

Animated Free Form Deformations (FFD's)[10],[3],[4]

FFD is a method primarily used for modelling. With this method a surface is deformed by translating the points on a driving Free Form Surface(FFS)[10]**.**The translations on the FFS can then off course be animated in order to get a deformation over a time period. This method allows for a non linear deformation of the surface. Unfortunately this requires the changes to be hand animated. A very time consuming task if the FFS is very dense.

The FFS points have an influence on an area of the deforming surface, and do not allow local control unless the FFS is very dense. An FFS with the requisite density would result in a control level that would be very similar to moving the points on the deforming surface by hand.

Non-linear deformers.

Most modelling/animation packages have a variety of smaller non-linear deformer tools. Bend, stretch, squeeze and twist are a few examples(see manual for your modelling/animation software package). These tools are useful but only in very few situations for which they were specifically created. They only allow for general non-local deformations of the entire object and have no options for local animation timing control.

Theory

The main theory of the system is having two objects, referred to as base shapes, where one morphs into the other. This is based on the vertices of the first base shape being animated along paths created by curves called base curves.

Creating this system requires some considerations on various topics.

Translating the vertices.

Animating an object along a path[9] can be an elaborate setup. Since the vertices of the deforming shape only exist as a point in space, This greatly simplifies the required calculations and considerations since the orientation of the vertex along the path is unimportant. Placing a vertex on the curve would require the calculation of a position on the curve at a certain parametric value.

Choosing the most suitable curve type.

One of the main areas of research for this system is curves. It is necessary to use curve types that give the simplest control and best access to the curve shape.

Four requirements for the curves have been defined:

1. It must be a curve that gives the user local control when manipulating the points that define the curve(control points). Without this it would be very difficult to place the curve into the desired shape.

2. The curve would have to be an interpolated curve[2], having its control points placed on the curve.

The problem with the approximated curves[2] is that are defined by approximating points not on the curve path. Larger translations of the control points would result in their removal from the area of interest. Having multiple curves with all their individual control points shown , would result in a large number of control points. It would not be possible to visually correlate a control point to its curve.

3. It is required that the curve is easily controlled. Requiring no more than a translation of the control points. With a large number of available curves, too much interaction required from the user would make it would slow the process. Therefore additional information besides the control points should not be exposed to the user.

4. A parametric curve is required to allow the vertices to be animated along the curve.

Most animation systems support NURBS curves[2] , which are good for a variety of uses. Unfortunately they do not fulfill the second of the requirements alone, as they are not interpolated curves. NURBS and the other curves in the B-spline family[2] will not be ideal for the system since they are all approximated curves.

A solution to the interpolation requirement is the use of cubic Hermite splines[2].

Catmull-Rom curves[9], Kochanek-Bartels splines (KB splines)[6] and Cardinal splines[6][11] all belong to this family.
KB splines are interpolated cubic splines where three parameters exist to control the tension, continuity and bias of the spline, giving a detailed control over the curve.
Cardinal splines have their tangent vectors defined by their control points and have a single parameter to control the tension of the curve.
Catmull-Rom curves are then a subset of Cardinal splines where the tension parameter is defined as 0.5, making the curve only dependent on the control points specified. This makes the cardinal splines a Catmull-Rom curves extremely simple to control.

Since the curves are not directly created by the user the Catmull-Rom curves are not necessarily superior to the KB and Cardinal Splines.
If the system would find a suited set of settings for the curve type, based on the specified amount of control points and the distances between the two vertices of the base shape. The user will be left with the same required interaction as with the Catmull-Rom curves.
The parameters of the curves should not be directly controllable by the user since it would lead to an overly complex interface. Any of the three curves types could be used, however it is up to the creator of the system to decided how much he wants to control.

Some packages like Alias Maya only supports NURBS and not KB splines. Unlike 3ds Max and Lightware.
Therefore an implementation of the curves would have to be added. If for instance a Catmull-Rom curve is implemented an optimisation may be needed due to the relatively long evaluation time for the curve[1].

Handling complexity and visual feedback.
An inherent problem with having curve driven deformations is the sheer amount of curves that are used. When all displayed at once they would immediately prevent the user from gaining any intuitive access to the system.
Therefore curves should only be shown when needed by the user. Additionally changing the intermediate deformation of the shape being manipulated would require changes to a vast number of curves.
A solution found for both the display and control issues was to sort the curves into a hierarchy structure consisting of additional layers of curves controlling the base curves.
Base layer curves would be linked to a new control curve that would then be driving the base curves. They would in turn be controlled by and grouped to yet another layer of curves, continuing until there was only a few top layer curves left.

In this way it would be possible to apply general changes through a top layer curve. The changes would then distribute themselves down the hierarchy resulting in a deformation of all the base curves in that branch.

If there is a need for more local control, a curve further down the hierarchy could be used. In this way changes would only occur to its descendants. If full control is needed the base curves could then be changed.

Creating this hierarchy would lead to additional curves being created cluttering up the scene further.
The display of curves could now be done making use of the existing hierarchical structure. A display chain would be setup between the groups.
As a default only the top layer curves and their controls would be visible to the user. Once the user selects a top curve it would trigger a display of its children and their controls. Then picking one of the children would trigger a display of its children and hide siblings of the now selected curve.
In that way the only curves visible would be the selected curves, their children and their ancestors.

These features would give a clear overview of the scene and allow changes at various levels. This allows quick access down the hierarchy.

The simplification of the the scene still leaves problems. Figuring out which direction to take down the hierarchy can be hard, therefore additional feedback is needed. This can be implemented by high lighting vertices belonging to the descendent base curves of the selected curve. High lighting of vertices would happen when hovering over a curve.

To give a better overview when creating the first couple of layers in the hierarchy, adding an option for quick feedback showing where to find free vertices or base curves can be done.

Deformation relationships between the curves.
Controlling the deformation down the hierarchy is an important area to be addressed.

Two different approaches have been found for this.

First one is to have a direct relationship between the control points on the curves, and the second is a relationship based on the deforming space of a parent curve.

The first approach is a simple and easy method that allows for certain relationships not easily available with the second method.
The method sets a initial constraint. The number of control points has to be the same on all curves in the hierarchy.
With this method the relationship between the curves is based on direct influence between the control point on the curves. A positional relationship would occur between he first control point on the parent curve and the first control points on each of the child curves.
This relationship can be defined with various algorithms. The most simple one is having any transitions on the parent control point be a linear change to the positions on the child control points.
This simple relationship allows for another relationship going upwards in the hierarchy. By placing the parent control point in a central position relative to the children , a feedback will traverse up the hierarchy with local changes.
If there is a change to the children then the parent will be moved to a the midway point of the children curves.
With the linear approach a problem arises. If having a large group of curves being controlled by a parent curve, then a single child curve would have little influence on the position of the parent curve. Therefore if the single child curve is moved far away from the parent curve (and the average centre for the children group) then the positional change up to the parent will not be enough to describe the general area and direction that the group of children creates.
This problem would be addressed by using a relationship based on the squared distances between the control points, like for instance the Least Squares fit algorithm[8].
Using squared distances results in control points further away from the main cluster of control Points having a greater influence on the parent control point in comparison to the others. The results of the changes would be a better representation of the general position of the group of curves.

The second method was to deform the underlying curves according to the changing space of the parent curve. The control points of the children would be placed locally in a space defined by the parent curve. Changes on the parent curve would then result in a new position for the children in world space, but they would still remain at their local position.
A final position of the control points for the child curves would the be decided by a transformation queue coming down the hierarchy.
A limitation of this method would then be that upwards influence in the hierarchy would be very hard, if not impossible.
The strength of the method is that it allows for differences in the number of control points on the curves.

Changing base shapes

If used as part of a deformation hierarchy this system would have to be setup and manipulated separately, giving only the local positions of the deformed shape vertices to the hierarchy.
This is needed to ensure ease of use and usability. A similar case exists when using blend shapes. Having the entire system placed together with for instance a rig system would result in a confusing and and complex relationship with the existing scene elements.

If there is a need by the users for a system allowing base shape changes a proposal for a solution have been created.
The end Control Points on the base curves would have to be attached to the position of the base shapes. This is a trivial task in contrast to the problem of handling the in between control points.
A possible solution would be to create a space defined by the vector between two corresponding vertices, placing base curve control points in that space. Then when moving the base shape vertices the space would deform and the control points of the base curves would receive transformations relative to that change.

This method would work fine using the relationship technique where the control points are directly dependent on each other. The base curve Control Points would change and that change would then traverse upwards moving each curve it passes up the hierarchy.
With the curve space transformation method you take the transformation done to the space created by the vectors. Then insert it into the end of the transformation queue coming down from the hierarchy onto the base curve. Since these transformations will be on a per vertex base then it will change the relationship between the curve's world space position and local parent curve space.
To resolve that a relative transformation would have to be calculated and applied to all the parent curves of the changed base curve without triggering in a change that would then traverse back down the hierarchy.
The resolution of that problem wont be discussed here since its reaches beyond the scope of the paper.

Vertex remapping

A final improvement for the system is the ability to change the mapping between the vertices of the two base shapes. Such a change would be relatively trivial with a curve based system. This is due to it only being a matter of changing the base curves to another vertex location on the destination base shape. This could lead to problems when grouping the curves since it would not be possible to keep it as groups of curve clusters, requiring the user to be thoughtfull when remapping.

Implementation

A prototype system has been implemented in Autodesk/Alias Maya, using MEL (Maya Embedded Language), in order to quickly test out some of the methods,usability and functionality of the theory outlined in the previous chapter. MEL makes it possible to avoid a need to implement several features already existing inside the Maya package, such as curves, animation paths, and constraints.
The implementation of a proper system would be done inside the Maya API due to MEL's limitations and speed. But since this system it only meant as a proof of concept, using MEL was sufficient and it made the development process faster and easier.

Worth noticing is that the system does for the most part not have any error checking and therefore does not prevent the user from doing certain actions that will break the system. Since this is still only a proof of concept it was decided not to be an issue, and its then required by the user to be very accurate with selections and operations being called.

The deforming surface is created by duplicating the first base shape and attaching clusters on each vertex of surface. Then these clusters are placed on a curve spanning between the corresponding vertices of the two base shapes. When first created the curve is a straight line between the vertices, which simplifies the initial setup.

The first attempt involved NURBS curves with the control vertices (cv's) point's being the control points for the curve. As theorised in the above section it did not turn out to be a good solution. The CV's quickly became cluttered between each other and it was increasingly hard to identify which curve the control points belonged to.
Edit Point (EP) curves(NURBS curves created using Edit Points) can be created and tweaked in Maya but unfortunately there is no way to attach clusters to the EP points(Maya converts the selection to CV points before attaching the cluster).
In order to get around that it was decided to use pointOnCurve constraints(see maya manual) to act as control points. A slower method, but enables the user to have the control points on the curve at all times.

The constraint uses a locator as a control object and forces the curve to go though the position of the locator by altering the positions of the CV's.
In order to get a smooth and local deformation of the curve, the CV points of the curve should be placed in a very particular way. The best solution found is to create two CV points closely on each side of the desired constraint position(as seen Fig. 1). This gives a local and sharp deformation depending on the distance between the CV's and the control point.

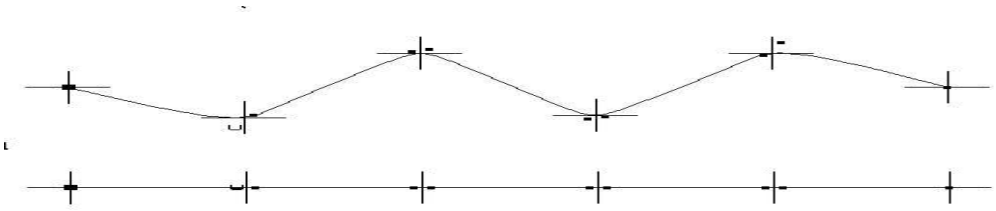

Fig.1 Showing the cv spacing in relationship to the point constratints on the curve. At the top is an example of the deformation.

At the ends there is no need for two CV's and only a single CV has been created with constraint attached to it.
The end constraints would have to be constrained to their corresponding vertex on the base shapes. Since that is not a strait forward thing in MEL, it was decided that it was not necessary since the base shapes would not have to be moved to illustrate the system.

Placing the CV's in these formations creates a problem. The pointOnCurve constraints requires an parametric parameter in order to be placed, and with the CV formations the parameter spacing was suddenly changed to values hard to calculate.

Solution for this is then to use the clostestPointOnCurve command(see Maya bonus tool manual). The command is given a calculated world space position for the constraint and is then returning the parametric value for the closest point on the curve. The method works fine but does result in a small processing time, since Maya has to create the nodes, fill in the data, evaluate and then delete the closestPointOnCurveNode. This will then happen for each base curve created on the shape and the time will add up to be significant. The evaluation of the node being the major time consumer.

For the curve relationships the the method using direct control point dependencies was chosen. The method is faster to implement and would still be able to show of the main concept of the whole system.

This then introduces the restriction of only having fixed amount of control Points for all the curves, with exception with use of linear base curves.
Linear base curves can be created setting the intermediate control Point amount to 0 in the GUI. Detecting this setting the system will create simple first degree curves and no control points.
This has been implemented since the user may decide that a linear interpolation is sufficient for certain areas, and therefore a simpler and faster setup can be used.

A selection of procedures to create new base curves, defined by either vertex or face selection have been created to give the options of different work flows. The user can paint a selection of vertices and create base curves. Thereafter the user can quickly group the new curves and in that way keep the density of curves to a minimum.

With the curves created the clusters can be attached to the curve. Changes to the surface deformation can now be done through the control points on the individual base curves.

To take care of all the curves being created the hierarchical control point relationships and the hierarchical curve display methods described in the theory section has been implemented.

To link the curves together a set of procedures were created. The procedure creates a parent curve with the same amount of control points as its children and places it in a central position in relation to the children. An attribute is then added to the parent containing the names of its children, making it easier for the rest of the system to traverse the network.
These procedures can then be used to construct the hierarchy all the way up to the top layer curve, assigning unique colours to the curves on each layer.

Once a setup of the curves and groups have been completed the procedures to control the positional and display relationships can be run. These procedures are divided into 2 sets, each taking care of one type of relationship.

Transformation relationship set:
Consists of the setupMove(), slaveToMaster() and masterToSlave() procedures.
The setupMove() procedure traverses down the tree structure created by the grouping. Running the slaveToMaster() and masterToSlave procedures on each set of parent-child groups.
Each of these two procedures creates new procedures that is then told to trigger if there is a movement on any of the curves.
The procedures generated by the slaveToMaster() procedure takes care of position changes done on the children. If a translation of the child's the procedures are activated and will adjusts the position of the parent according to the position of its children.
Similar is the masterToSlave() except it takes care of change from the parent down to the children.
In order to avoid an endless loop(child changes parent that is then changed and changes child...etc) each procedure does an extra check to see if the control points are already in place. If it is the case the procedure will not preform any changes to the curves.

Selection system set:
Consists of the resetProcedure() and the selectionProcedure() procedures.
These procedures are in place to setup the display properties of the system.

Maya has no MEL functionality to directly inform when a object has been selected. It only informs if a selection has changed. Therefore it is necessary to setup a large script that checks what is selected and then display things accordingly.
This is what the selectionProcedure() creates. The selectionProcedure() traverses through the hierarchy collecting informations about the groups and creates a list of IF statements defining what to do if a certain object is found to be selected according to the method described in the theory section.

This now enables the user to traverse through the tree of curves by clicking on the first curve, getting access to the sub level, and then continuing that process by clicking on the next curve.

Both sets of procedures will create a scriptNode and store the generated procedures and scriptJob calls in the node. This node will then be saved with the scene file and will re-run the scripts if the file is opened again, making it possible to save and reload the scene without loosing the many procedures to keep it running.

Feedback:

Various features have been implemented to give the user feedback on which CV's are not attached to curves and then which base curves are not grouped. Colour feedback on the base shapes shows the user what vertices or what base curves have not yet been used.

This enables the user to have a good overview during the setup without loosing track of the progress.

Animation control:

An important aspect of the system was to be able to re time and control the animation of the deformation. Several procedures was implemented to enabled the user changing time of the start, end and duration(offset from start) of the curve animation. This can be done in pre set increments or by directly specifying the time range.

Further use of the Maya Artisan tool (attribute paint tool) was used to enable the user to paint the start,end and duration directly on the base shapes in order to change the timings. This method is very intuitive and using the smooth functionality it can create a smooth change of timing across the surface.

Access to procedures

The access to to all the various procedures created for the system is a significant aspect of the system. Many of the operations has to be done frequently and having to remove the control from the main view port would result in a large time waste, while also be a tedious task. Therefore procedures has then been assembled in a “marking Menu” triggered by a key shortcut. The marking menu allows the user to have instant access to the main functions used for the current stage in the setup. Once a stage is complete and the user starts the next, they can switch the mode of the menu, revealing a tool set for the selected stage. Constantly having the fastest access to the most relevant tools.

Results and tests:

The test system was tried on various scenarios that has shown to be difficult to handle with existing methods.

A test was done morphing a human head to the head of a lion. To start with all the vertices was connected by linear curves. A process that takes a couple of seconds for the user, but also took several minutes for the scripts to generate.

After that a quick painting of the start values were done on mouth and nose of the lion, making that part deform faster. This setup was created within minutes.

The visual result can be seen in figure 2 together with the results from a normal blend shape setup for comparison.

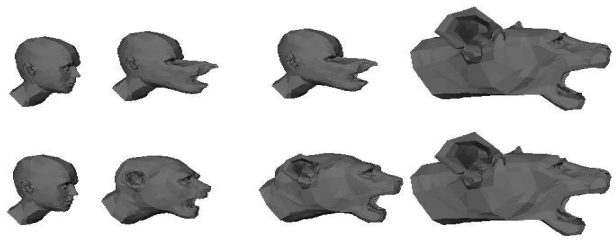

Fig.2 Demonstration of local changes in animation timing Blend shape at bottom, Curve system at top

The test is unfortunately not the most beautiful one, since it is done to show the change in deformation between the two.

A more elaborate test was done. Here there were several problems encountered. The most occoring problem was that there was no feedback on what curves belonged to which vertices.

This made it near impossible to keep track of curves on a denser shape.
Creases were seen in the mesh between vertices that belonged to different subsets of the groups and where the groups were not following the same direction. In order to resolve that problem a large amount of local tweaks would have to be done to smooth out the difference between the two groups.
Also it was experienced that the system could in rare cases crash Maya. Its not believed that the cause is the system itself but instead Maya's handling of very intensive calculations for a longer period.

The mock-up system is lacking in some of the feedback and functionality described in the theory section. Despite that the system has turned out to be very useful on smaller surfaces and by using it it has been found that its a quick and intuitive system. There are an considerable amount of waiting for the system at moments. Therefore a faster system is needed to increase productivity.

Once the scriptjobs has been run its not possible to change the curves or groupings of curves at all, therefore leaving the user with the only option of starting from scratch(or last save) if something has gone wrong.
Close deformations does not work to well due to the fact it is hard for the user to control the points so close to the surfaces. Here a normal blend-shape technique would be better, or using the linear curve setting on those areas.

Under the testing it was noticed that since the system is most suitable deformations over relative larger distances it could very well be sublemented with an overlaying blend shape deformation. Having the curve driven system handling the rough deformations while using the blendshapes to take care of the finer areas. There is normally not any need for anything more than a linear interpolation on small areas so this would work out well.

Future Work / *Improvements.*:
Keeping the proof of concept system in mind, there is a multitude of improvements that could be done.
Further testing with the system should be done to identify additional problems if any.

The test system only allows for changes in animation timing on the start,end and length for each vertex. In a full scale implementation option of attaching an actual animation data set to the curves would improve the functionality of the system.
As discussed in the results section there is huge need for a feedback system that can tell the user which vertecies belongs to which curves. Implementing a such system would greatly improve the users overview of the system, and remove some of the of trial and error seaching.

Despite that the method with a direct positional relationship between the control point worked, it had its limitations. Mainly due to the restriction with the control point number. Therefore the method with the transformation hierarchy would probably work better. Having the option of controlling exactly how many curves is needed on a curve would be benefiting to an more customizable but also simpler system for the user. Allowing for both global and local control on the curves.

The option to change the mapping of the base curves should also be implemented. Opening the for solving a new set of problems normally not solvable. Preferably the option would be accompanied with a manipulation tool enabling the user to drag the base curve around and snapping it to vertices.

A major step in the right direction would be to implement the system directly in an API, such as the Maya API.

The MEL approach is very slow, and it quickly clutters up the scene with large amount of nodes.
With the API most of the tasks and control could be contained in one single node. The node would take in the geometry from the two base shapes and then output a deformed shape that then could be used in further deformations.

The Maya API allows for much greater control with selections and custom manipulators, making it possible to create faster and more intuitive set of controls.

The creation of more suited curve types is also made available and a implementation of one of the qubic hermite curves could simplify the control of the curves.
Implementing the space deforming relationship for the relationship between the curves would also be made easier. With a centralised place to keep the informations (new node created) it would be possible to store all the transformations needed for the curves and keep everything in a relative space.

Having most of the system stored in a single node would also remove the need for the vast amounts of scriptJobs running, making things much faster and less wasteful in terms of checking. Keeping track of relationships between the many curves would also be easier since having it stored as data would be less of a problem. Re editing would also be an option. With the current system, there is no support for changes in the system after the selectionProcedure and moveProcedure have been run.

Conclusion:
With the proof of concept system it has been showed that there is great potential in the system. While at the same time showing some of the limitations, and revealing some problems that would need to be adressed in future implementations. It also indicates that the ideas devised for the user interaction helped make it an easier and more responsive system to handle.
With further tests and a more elaborate implementation the system would be a valuable deformation technique for a variety of tasks.

Acknowledgements
Ian Stephenson, for tutoring and giving advice through out the project.

Chris Bull, for the models used for the test morphing between human and lion.

Eike Anderson, for helping out with advice on the structure of this paper.

David Stopford, for helping to guide me through the English language.

References:

[1]
BARRY, P.J. AND GOLDMAN, R. N., 1988. A recursive evaluation algorithm for a class of Catmull-Rom splines, *ln: BEACH, R.J ed. Proceedings of the 15th annual conference on Computer graphics and interactive techniques 1 - 5 august 1988 Atlanta,GA* .New York, NY:ACM Press, 199-204

[2]
COMNINOS, P. 2003, Cubic Polynomial Curves and Surfaces, *ln:* J. VINCE, ed. *Handbook of Computer Animation.* London : Springer, 123-212

[3]
COQUILLART, S. , 1990. Extended Free-Form deformation: a sculpturing tool for 3D geometric modeling, *Proceedings of the 17th annual conference on Computer graphics and interactive techniques 6 - 10 august 1990 Dallas,TX* .New York, NY:ACM Press, 187-196

[4]
COQUILLART, S. AND JANCÉNE, P, 1991. Animated free-form deformation: an interactive animation technique. *Proceedings of the 18th annual conference on Computer graphics and interactive techniques 28 July – 2 august 1991*.New York, NY:ACM Press, 23-26

[5]
JOSHI, P. , TIEN, W.C., DESBRUN, M AND PIGHIN, F, 2003. Learning Controls for Blend Shape Based Realistic Facial Animation. Eurographics/SIGGRAPH Symposium on Computer Animation, 187-192.

[6]
KOCHANEK, D.H.U. AND BARTELS, R. H., 1984. Interpolating splines with local tension, continuity, and bias control. *Proceedings of the 11th annual conference on Computer graphics and interactive techniques july 1984 Atlanta,GA* .New York, NY:ACM Press, 33-41

[7]
LEWIS, J.P., CORDNER, M AND NICKSON, F, 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques New Orleans, LA.* New York, NY:ACM Press/Addison-Wesley, 165 - 172

[8]
MATHWORLD,1999,*Least Squares Fitting*[online]. CRC Press LLC.
Available from: http://mathworld.wolfram.com/LeastSquaresFitting.html
[Accessed 10 March 2006]

[9]
PARENT, R. , 2002, Computer Animation : Algorithms and techniques. London : San Francisco : Morgan Kaufmann Publishers

[10]
SEDERBERG, T.W. AND PARRY, S. P., 1986. Free-Form deformation of solid geometric models. *ln: EVANS, D.C. AND ATHAY, R.J. ed. Proceedings of the 13th annual conference on Computer graphics and interactive techniques 18 - 22 august 1986 Dallas,TX* .New York, NY:ACM Press, 151-160