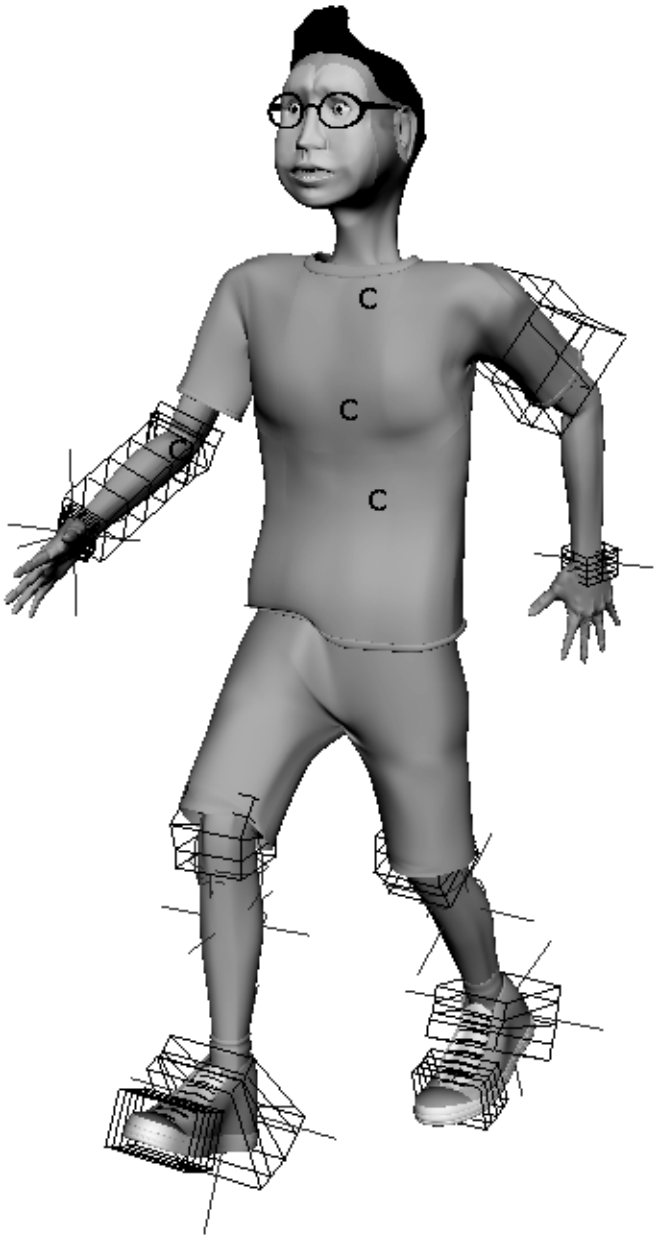


Character Animation in Maya

Alias|Wavefront
a Silicon Graphics Company
AP-M-CAM-03

Character Animation in Maya

Assist Team:

Damon Riesberg, Lee Graft, Matt Baer,Andy Harris, Cory Mogk, Eric Saindon

Special thanks to:

Corban Gossett, Kevin Lombardi, Jason Schleifer

Education Publishing Group:

Bob Gundu, Steve Christov, Robert Magee, Deion Green

Alias | wavefront

A SiliconGraphics Company

Introduction

Character Animation

Animating digital characters is a growing field that is being incorporated into commercial and feature projects. The demands put on today's characters is growing at a rapid pace. Thanks to Maya the job of setting up and animating a character to meet these demands is made possible and fun.

Introducing Melvin

The *Character Animation in Maya* course explores the set-up and animation of Melvin. As you setup Melvin you will explore methods for creating skeletal and muscular systems, enabling the animation of this character in Maya.

Melvin

By the end of this course, you will have learned the following:

- How to build and operate a complex hierarchy of joints and deformers that have been applied to a character-based model.
- How to create custom controls that streamline your workflow to allow for more creative and direct animations.
- How to animate Melvin performing in typical animation scenarios such as walking and talking.

COURSE STRUCTURE

This course is taught in a lecture / lab format. Some lessons focus on character setup while others focus on animating the character in its current state of development. Premade files are provided for each lab. In lessons where involved concepts are introduced, simple test beds are explored as well.

Who this class designed for

If you want to learn how to set up and animate a character in Maya this is the class for you. This course is designed as a stepping stone for those who have completed either *Learning Maya* or the *Maya Essentials* course and wish to develop their abilities with character animation. This course is also designed as a transition course for experienced animators who are new to Maya's character animation toolset.

Technical Directors will also find this course important for learning good methods of setting up their characters.

Melvin's bones

You will first learn how to set Melvin up. Here are examples of two different techniques that have been used when setting up Melvin's skeletal controls:

- **foot and leg**—Single chain solved IK chain for leg with pole vector constraint determining knee direction. Foot controls for rotating ankle and bending toe controlled by selecting one object at the ankle.
- **hand**—Instead of using Inverse Kinematics, *Set Driven Key* is used to drive Forward Kinematics of the finger joints. You will create a custom interface to ease the keyframing process for the hands.

Melvin's skin

Melvin will be skinned using three different methods:

- **feet, legs, arms, hands**—traditional methods of skin to the joints
- **upper and lower torso**—lattices (deforming skin) and wrap deformers bound to the joints
- **complete body**—smooth skinned wtih influence objects parented to the joints

Melvin's muscles

Many of the different flexor and deformers that Maya provides to create realistic control of both muscles and clothing are incorporated into this class.

- **bicep muscle**—influence objects set up with Set Driven Key
- **elbows**—cluster, lattice flexors, and influence objects

- **face**—blend shape targets created using lattices, flexors, clusters, and wire deformers

Set Driven Key is used in some cases to control the behavior of some of these deformers.

Animating Melvin

Once you've worked through character set-up techniques, you'll take Melvin through several animation scenarios to test out both his construction and the controls that you create. These include:

- Walk cycle
- Kicking the can
- Facial animation / lip sync

You'll explore several techniques to aid in animating Melvin more efficiently and realistically:

- motion studies
- keyframe techniques
- using breakdown keys to keep positional relationship between standard keyframes
- using image planes for motion study
- using character sets to make attributes centrally located and easily keyed.
- using sets to simplify selection for skinning and weighting
- make animating convenient by setting up attributes accessible through the channel box
- shelf buttons
- utilizing pick masks
- using layers for display, hiding, and templating
- scene organization

Character optimization

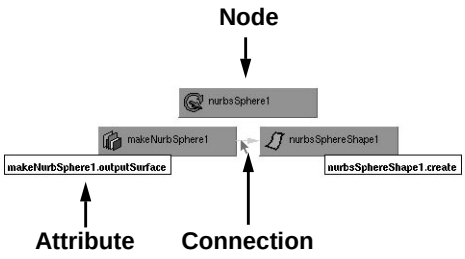
In the coming lessons, all the modelling work has been done for you. For reasons of efficiency, you should be aware of the geometry and its properties of construction at specific spots on the character. Pay attention to areas that are undergoing extreme deformation and thus require special consideration.

Also, note where the character may seem limited for specific applications. Generally a character is designed to perform specific actions and has been optimized for those. To make a character that will perform in all situations flawlessly could be an effort in futility as well a waste of time that could otherwise be spent animating. When you design your characters you should keep a specific list of abilities that are required by the script. This will go a long way in keeping the character from being overbuilt and too cumbersome to operate.

THE DEPENDENCY GRAPH

While creating characters, it is a good idea to have a basic knowledge of how Maya's system architecture works. Maya's system architecture uses a procedural paradigm that lets you integrate traditional keyframe animation, inverse kinematics, dynamics and scripting on top of a node-based architecture that is called the **Dependency graph**. If you wanted to reduce this graph to its bare essentials, you could describe it as *nodes with attributes that are connected*. This node-based architecture gives Maya its flexible procedural qualities.

Below is a diagram showing a primitive sphere's Dependency graph as shown in the Hypergraph view. A procedural input node defines the shape of the sphere by connecting attributes on each node.

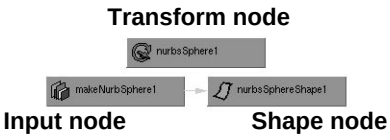


The Dependency graph

Nodes

Every element in Maya, whether it is a curve, surface, deformer, light, texture, expression, modeling operation or animation curve, is described by either a single node or a series of connected nodes.

A *node* is a generic object type in Maya. Different nodes are designed with specific attributes so that the node can accomplish a specific task. Nodes define all object types in Maya including geometry, shading, and lighting. Shown below are three typical node types as they appear on a primitive sphere.



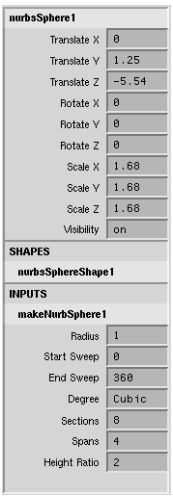
Node types on a sphere

Transform node - Transform nodes contain positioning information for your objects. When you move, rotate or scale, this is the node you are affecting.

Input node - The input node represents options that drive the creation of your sphere's shape such as radius or endsweep.

Shape node - The shape node contains all the component information that represents the actual look of the sphere.

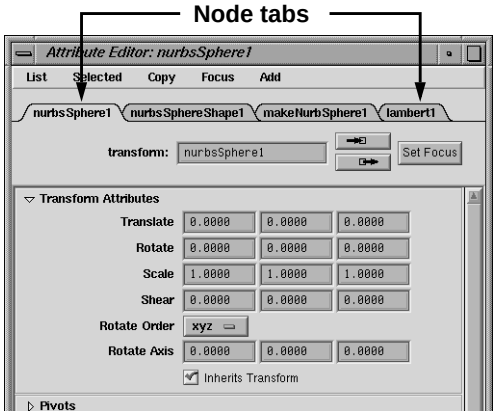
Maya's user interface presents these nodes to you in many ways. Below is an image of the Channel box where you can edit and animate node attributes.



Channel box

Attributes

Each node is defined by a series of attributes that relate to what the node is designed to accomplish. In the case of a transform node, *X Translate* is an attribute. In the case of a shader node, *Color Red* is an attribute. It is possible for you to assign values to the attributes. You can work with attributes in a number of user-interface windows including the *Attribute Editor*, the *Channel box* and the *Spread Sheet Editor*.



The Attribute Editor

One important feature in Maya is that you can animate virtually every attribute on any node. This helps give Maya its animation power. You should note that attributes are also referred to as *channels*.

Connections

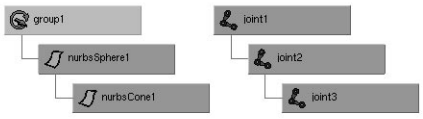
Nodes don't exist in isolation. A finished animation results when you begin making connections between attributes on different nodes. These connections are also known as *dependencies*. In the case of modeling, these connections are sometimes referred to as *construction history*.

Most of these connections are created automatically by the Maya user-interface as a result of using commands or tools. If you desire, you can also build and edit these connections explicitly using the *Connection editor*, by entering *MEL* (Maya Embedded Language) commands, or by writing MEL-based expressions.

Hierarchies

When you are building scenes in Maya, you can build dependency connections to link node attributes. When working with transform nodes or joint nodes, you can also build hierarchies which create a different kind of relationship between your objects.

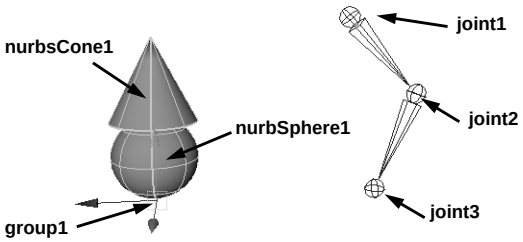
In a hierarchy, one transform node is *parented* to another. When Maya works with these nodes, Maya looks first at the top node, or *root* node, then down the hierarchy. Therefore, motion from the upper nodes is transferred down into the lower nodes. In the diagram below, if the *group1* node is rotated, then the two lower nodes will rotate with it. If the *nurbsCone* node is rotated, the upper nodes are not affected.



Object and joint hierarchy nodes

Joint hierarchies are used when you are building characters. When you create joints, the joint pivots act as limb joints while bones are drawn between them to help visualize the joint chain. By default, these hierarchies work just like object hierarchies. Rotating one node rotates all of the lower nodes at the same time.

When you are working with characters, you can use *inverse kinematics* to reverse the flow of the hierarchy.



Object and joint hierarchies

The Hypergraph

In Maya, you can visualize hierarchies and dependencies using the hypergraph. The following steps demonstrates how to work with various node types in the Hypergraph.

Working with hierarchies and dependencies

If you understand the idea of *nodes with attributes that are connected*, then you will understand the Dependency graph. You can see what this means in Maya by building a simple primitive sphere.

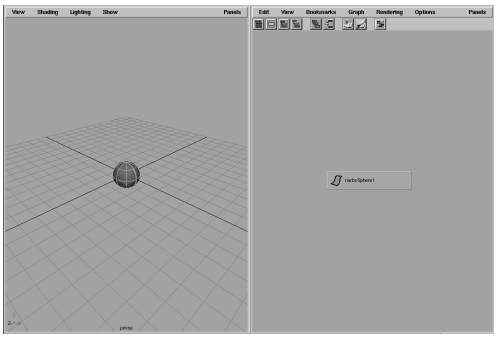
1 Set up your view panels

To view nodes and connections in a diagram format, the Hypergraph panel is required along with a Perspective view.

- Select **Panels** → ***Layouts*** → ***2 Side by Side***.
- Set up a Perspective view in the first panel and a Hypergraph view in the second panel.
- Dolly into the Perspective view to get closer to the grid.

2 Create a primitive sphere

- Go to the modeling menu set.
- Select **Create** → ***NURBS Primitives***→ ***Sphere***.
- Press **5** to turn on smooth shading and **3** to increase the surface smoothness of the sphere.



New sphere

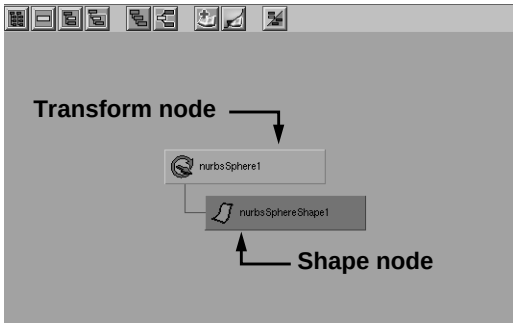
3 View the shape node

In the Hypergraph panel, you are currently looking at the Scene Hierarchy view. This Scene Hierarchy view is focused on *transform nodes*. This node lets you set the position and orientation of your objects.

Right now, only a *nurbsSphere* node is visible. In actual fact, there are two nodes in this hierarchy but the second is hidden by default. At the bottom of most hierarchies, you will find a *shape node* which contains the information about the object itself.

- In the Hypergraph panel, select **Options** → ***Display*** → ***Shape Nodes***.

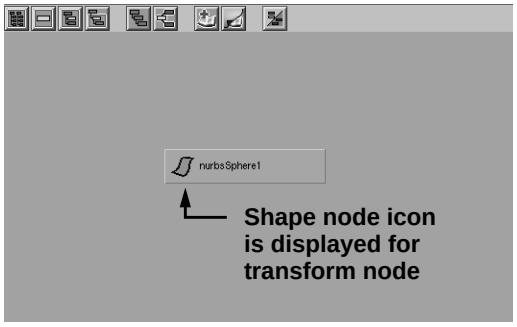
You can now see the *transform node* which is, in effect, the positioning node and the *shape node* which contains information about the actual surface of the sphere. The transform node defines the position of the shape below it.



Transform and shape nodes

- In the Hypergraph panel, select **Options** → ***Display*** → ***Shape Nodes*** to turn these off.

  You will notice that when these nodes are expanded, the shape node and the transform node have different icons. When collapsed, the transform node takes on the shape node's icon to help you understand what it going on underneath.

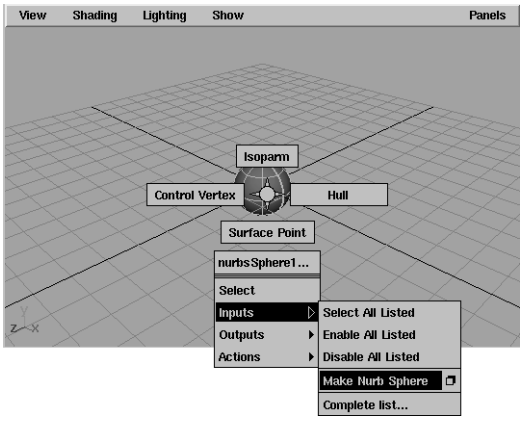


Transform node on its own

4 View the dependencies

To view the dependencies that exist with a primitive sphere, you need to take a look at the upstream and downstream connections.

- Click on the sphere with the right mouse button and select **Inputs** → ***Make Nurb Sphere*** from the marking menu.



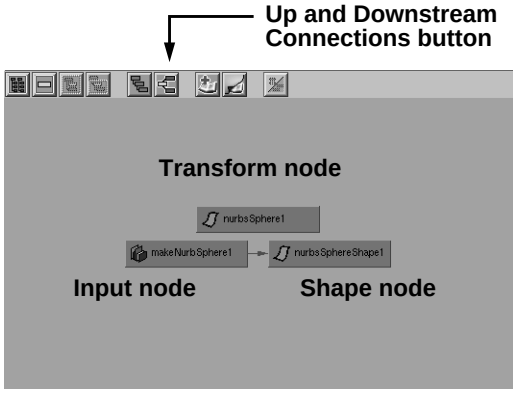
Selection marking menu

Note: You can also select the input node by choosing it in the Channel box.

- In the Hypergraph panel, click on the **Up and Downstream Connections** button.

  See the original transform node which is now separated from the shape node. While the transform node has a hierarchical relationship to the shape node, their attributes are not dependent on each other.

  The *input node* called *makeNurbSphere* is a result of the original creation of the sphere. The options set in the sphere tool's option window, have been placed into a node that feeds into the shape node. The shape node is dependent on the input node. If you change values in the input node, then the shape of the sphere changes.



Sphere dependencies

5 Edit the attributes in the Channel box

In the Channel box, you can edit attributes belonging to all of the node types. This lets you affect both hierarchical relationships and dependencies.

If you edit an attribute belonging to the *makeNurbSphere* node, then the shape of the sphere will be affected. If you change an attribute belonging to the *nurbSphere* transform node, then the positioning will be changed. Using the Channel box will help you work with the nodes.

- For the transform node, change the **Rotate Y** value to **45**.
- For the input node, change the **Radius** to **3**.

  You can set attribute values to affect either the scene hierarchy or the Dependency graph.

Animating the sphere

When you animate in Maya, you are changing the value of an attribute over time. Using keyframes, you set these values at important points in time, then use tangent properties to determine how the attribute value changes between the keys.

The key and tangent information is placed in a separate animation curve node that is then connected to the animated attribute.

1 Select the sphere

- In the Hypergraph panel, click on the **Scene Hierarchy** button.
- ***Select*** the *nurbsSphere* transform node.

2 Return the sphere to the origin

Since you earlier moved the sphere along the three axes, it's a good time to set it back to the origin.

- In the Channel box, change the **Rotate Y** attribute to **0**.

3 Animate the sphere's rotation

- In the Time slider, set the playback range to **120** frames.
- In the Time slider, go to frame **1**.
- Click on the **Rotate Y** channel name in the Channel box.
- Click with your right mouse button and select ***Key Selected*** from the pop-up menu.

  This sets a key at the chosen time.
- In the Time slider, go to frame **120**.
- In the Channel box, change the **Rotate Y** attribute to **720**.
- Click with your right mouse button and select ***Key selected*** from the pop-up menu.
- Playback the results.

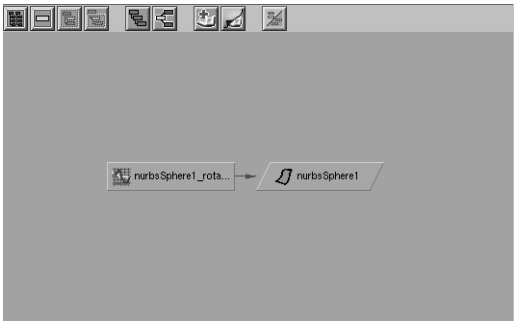
  The sphere is now spinning.

4 View the Hypergraph dependencies

- In the Hypergraph panel, click on the **Up and Downstream Connections** button.

You see that an animation curve node has been created and connected to the transform node. The transform node is now shown as a trapezoid to indicate that it is connected to the animation curve node. If you click on the connection arrow, you will see that the connection is to *Rotate Y*.

If you select the animation curve node and open the Attribute Editor, you will see that each key has been recorded along with value, time and tangent information. You can actually edit this information here, or use the Graph Editor where you get more visual feedback.



Connected animation curve node

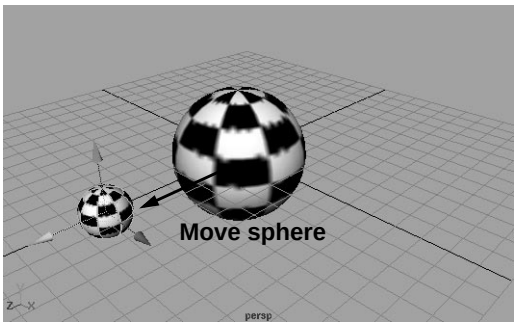
Parenting in the Hypegraph

So far, you have worked a lot with the dependency connections but not with the scene hierarchy. In a hierarchy, you always work with transform nodes. You can make one transform node the *parent* of another node, thereby creating a child which must follow the parent.

You will build a hierarchy of spheres that are rotating like planets around the sun. This example is a helpful way to understand how scene hierarchies work.

1 Create a new sphere

- In the Hypergraph panel, click on the **Scene Hierarchy** button.
- Go to the **Modeling** menu set.
- Select **Create**→ ***NURBS Primitives***→ ***Sphere***.
- **Move** the sphere along the Z axis until it sits in front of the first sphere.
- Press **3** to increase the display smoothness of the sphere.
- Go to the **Rendering** menu set.
- Apply a checker shader to both spheres

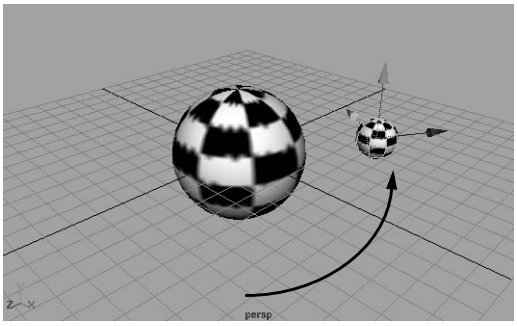


Second sphere

2 Parent the sphere to the first sphere

- In the Hypergraph, drag the node icon with **MMB** for the second sphere onto the first sphere. Now they are parented together.
- Playback the scene.

  The second sphere rotates along with the first sphere. It has inherited the motion of the original sphere.



Rotating hierarchy

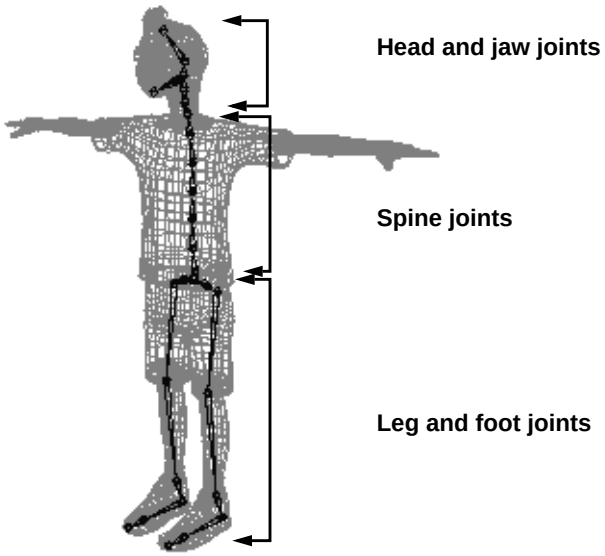
Summary

You should now know the objectives and outline and how the Dependency Graph works in Maya.

Lesson 1

1 Skeleton setup

To begin, you are going to set up the Melvin Character. To help you organize your character's movement and provide a framework for applying deformations, you must first build a skeleton chain by creating a series of joint nodes.



Melvin's bones

In an ideal world, you would create the skeleton perfectly the first time you create the joints. Realistically, you may want to later update or correct how your character is set up. To create the most flexibility for animating, you will set up the character's skeleton by combining several techniques to create the lower body skeleton.

In this lesson, you will learn the following:

- How to use layers and templating
- How to create joints for the legs, feet, back and neck
- How to connect and parent joints

CREATING THE LOWER BODY SKELETON

You will start work on Melvin by constructing the leg using existing geometry as a guide. To build the leg you will complete the following:

- Import Melvin's body geometry
- Add this geometry to a layer
- Template the geometry
- Build the skeleton

LAYERS AND TEMPLATES

A good tool for organizing the various parts of a character is the Layer Bar. This tool provides an easy way to separate all the parts of Melvin – geometry, skeletons, IK, etc. – into logical groups. In the Layer Editor, you can hide, show, and or template selected layers to speed up interactivity by reducing the visible elements in the scene.

A view of the layer bar

Show or Hide groupings of elements

The more elements you can hide in your scene, the quicker you can interact with your scene.

Select objects or groupings of elements

Sometimes it is difficult to select objects and groups of objects efficiently in the interface. If you find yourself picking the same object or group of objects, make a layer for them.

Template or untemplate groupings of elements

By templating objects, you will still see a transparent representation of them but they are not selectable. This enables you to pick some types of objects, and not pick the same type of others but still see them. Pick masks would not help you with this.

Tip: Selecting and displaying only the elements of the scene that you are working on is crucial to successfully operating in a complex scene.

Layers can also be used to logically break down your scene. You can make your background elements a separate layer and your foreground elements another layer. Characters and effects can be a on a layer that you may want to render separately as compositing passes.

Creating a layer and templating geometry

You are going to use Melvin's body to help position the skeleton properly, but you don't want to accidentally modify it, so you will create a separate layer just for the geometry, allowing you to template it. Templating is a means of inactivating something without necessarily hiding it.

1 Open a file.

- Select **File** → **Open**.
- Navigate to the `$HOME/maya/projects/melvin/scenes` directory.
- Open the file called *Melvin_01.geometry.mb.*

2 Create a new layer for Melvin's geometry

- Display the Layer Bar by selecting **Options** → **Layer Bar**.
- Select **Edit** → **Create Display Layer**.

  A new layer should appear in the Layer Bar.

Tip: This same function can be achieved by pressing the farthest button on the left in the Layer Bar. This button creates a new display layer.

- **RMB** on *layer1* and select **Layer Attributes...**
- Name the layer *melvin_skin_all.*
- In the Outliner, **Select** the following group nodes:

  Body, *Head*, *Shorts*, *Legs*, and *Hair*.
- **RMB** click on the *melvin_skin_all* layer and select **Assign Selected.**

  Those selected attributes are now apart of the layer.

3 Template this layer.

- **RMB** click on the *melvin_skin_all* layer and switch from **Standard** to **Template.**

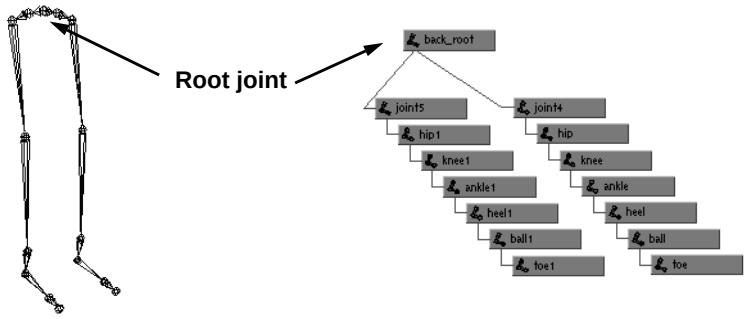
  Now you can see Melvin's geometry but you cannot select it using the default select modes. Later, you will learn how to select templated geometry.

BUILDING SKELETON JOINTS

In Maya, a skeleton chain is made up of joint nodes that are connected visually by bone icons. A skeleton chain creates a continuous hierarchy of joint nodes that are parented to each other. The joint and bone icons help you visualize the character's hierarchy in the 3D views but will not appear in your final renderings. The top node of the hierarchy is known as the root joint.

Traversing hierarchies

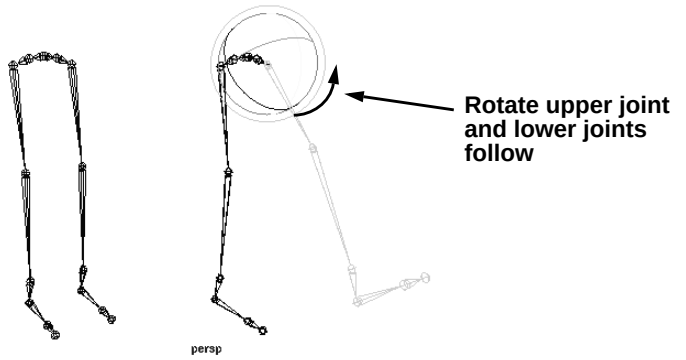
Use the up, down, left and right arrow keys to quickly traverse through the joints in a hierarchy.



Joints and bones showing nodes in Hypergraph

Joint hierarchies let you group or bind geometry. You can then animate the joints, which in turn, animates the geometry. When you transform the joints, you will most often use rotation. If you look at your own joints you will see that they all rotate. Generally joints don't translate or scale unless you want to add some squash and stretch to a character's limbs.

When you rotate a joint, all the joints below it are carried along with the rotation. This behavior is called **Forward Kinematics** and is consistent with how all hierarchies work in Maya. In the next lesson, you will learn how to use **Inverse Kinematics** to make it easier to animate joint rotations.



Joints rotations

To build Melvin's skeleton, you will first create joints for his legs and feet. Later you will build his back and neck, then this system will be connected to the legs at the hip joints.

Creating the leg skeletons

You will start by creating the left leg skeleton, then mirroring it to create the right leg skeleton.

1 In the side view, draw five joints for the left leg.

- Select the **Skeleton** → **Joint Tool** - ❐.
- In the option window, set the following options:

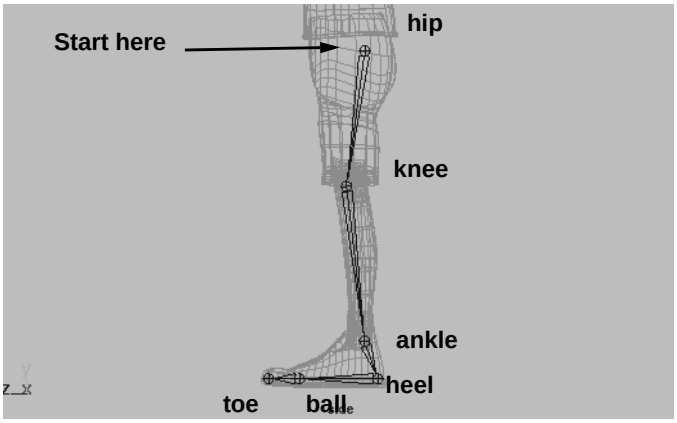
  Auto joint orient to **XYZ**.

  (Joint Orientation will be discussed further in chapter 6.)

- Starting with the hip, place 6 joints for the leg as shown below.

Note: Be sure to draw the knee in a bent position. This will make it easier to apply an IK solver to the chain in the next chapter.

- Rename the joints: *hip*, *knee*, *ankle*, *heel*, *ball*, and *toe*.

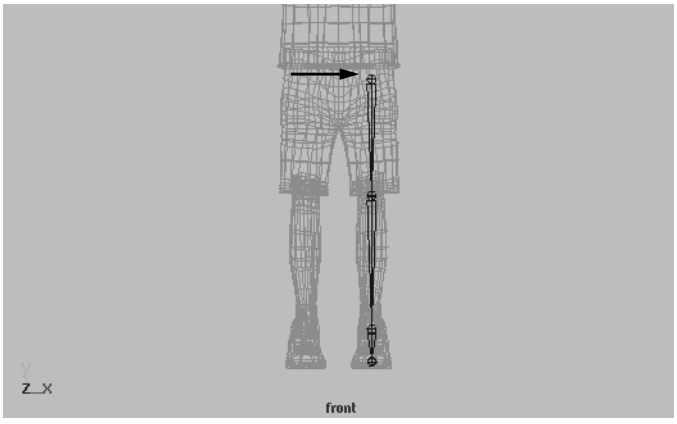


Left leg joints

2 Move the joint chain to Melvin's left side

The *hip* joint is now the root node of the leg's joint chain hierarchy. If you pick this node then you can move the whole chain together.

- Select the *hip* joint and move it along the X axis so it fits inside the geometry of Melvin's left leg.

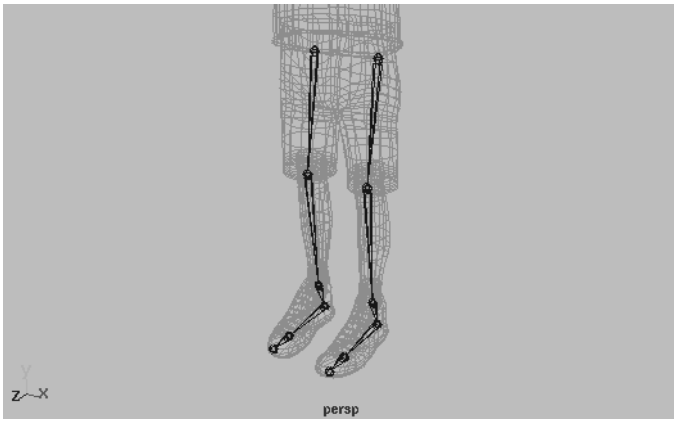


Moving the joint chain to Melvin's left

3 Duplicate the leg to create the right leg

- **Select** the *hip* joint.
- Select **Edit** → **Duplicate**- ❒ and press **Reset**.
- **Move** the new leg over to the right.

If you look at the X translate of the left hip you can put this value on the right hip and make it negative so that the skeleton is symmetrical.



Copied Legs

4 Rename your joints to include left_ and right_ prefixes.

- Select the root joint of the left leg.
- Select **Modify** → **Prefix Hierarchy Names...**
- Enter *left_* in the text field. Click **OK** to validate.
- Repeat, using *right_*, for the right leg joints.
- Edit the right leg joint names to remove the **1** that was appended in the duplicating step.

Tip: It is important to take the time now to name your objects so that down they'll be easy to select and replace.

---

5 Save your work

You will want to save your work in a way that makes it easy to access later. For this course, it is recommended that you save your files using the following convention:

- Select **File** → **Save As.**
- Enter the name *lesson1a.mb.*

  Rather than saving as *lesson1* then overwriting your work with a **File** → **Save**, it is recommended that you always use the **File** → **Save As** command and rename your file as *lesson1a.mb*, *lesson1b.mb* etc... This way you have many files that represent different stages in your work. This is useful if you need to go back a step or two to review some steps.

  The first part of the file name represents the lesson you are working on and the second part indicates the file type. The *.mb* suffix represents **Maya binary**.

Maya offers two file types – **Maya binary** (.mb) and **Maya ascii** (.ma). The binary format is more compact and loads faster making it better for larger files. The ascii format allows you to open the file in a text editor and tweak the file. You will be using **Maya binary** in this course.

Note: From now on, you will be prompted at key points to save your work. Use **File** → **Save As** and name the file accordingly. Saving frequently is always a good idea.

Creating the backbone, neck, and head

You will now create another skeleton hierarchy for the backbone, neck, and head. Start drawing joints from the base of the backbone. This will make the base joint the root joint of the skeleton hierarchy. The root joint is important since it represents the top of the hierarchy, even though it might be the bottom joint in the 3D view. Using the backbone as the root, the upper body and the legs will branch off from this node. You can then move the whole skeleton hierarchy by simply moving the root.

Placing joints

While placing joints, you can use the **MMB** to move the last joint that was placed.

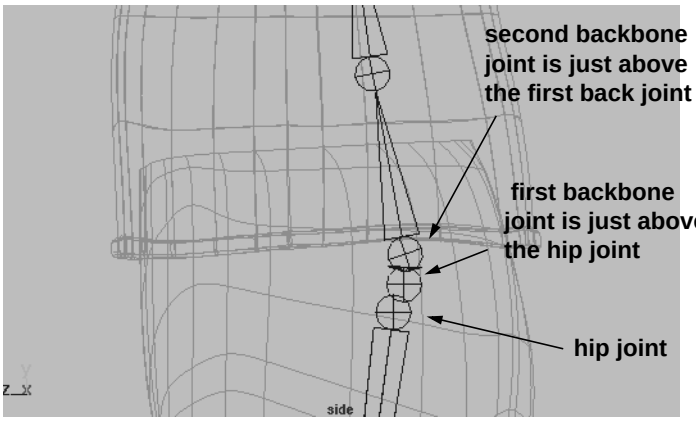
The **move** tool will move the selected joint and any joints below it in the hierarchy.

The **move pivot** tool (while in move mode press the **insert** key to toggle into this mode) will only move the selected joint.

1 Place the first joint of the backbone just above the hips

- Select **Skeleton** → **Joint Tool** - ❑.
- Make sure that **Auto Joint Orient** is set to **XYZ**.

  By creating the joints with **Auto Joint Orient** set to **XYZ** the X axis will always point towards the child joint. You set the orientation to XYZ so that the local rotation axis of the joints will be aligned in the direction of the backbone. This topic will be covered in more detail at the end of this chapter and throughout the course.
- LMB click just above the hip joints to place the first joint.

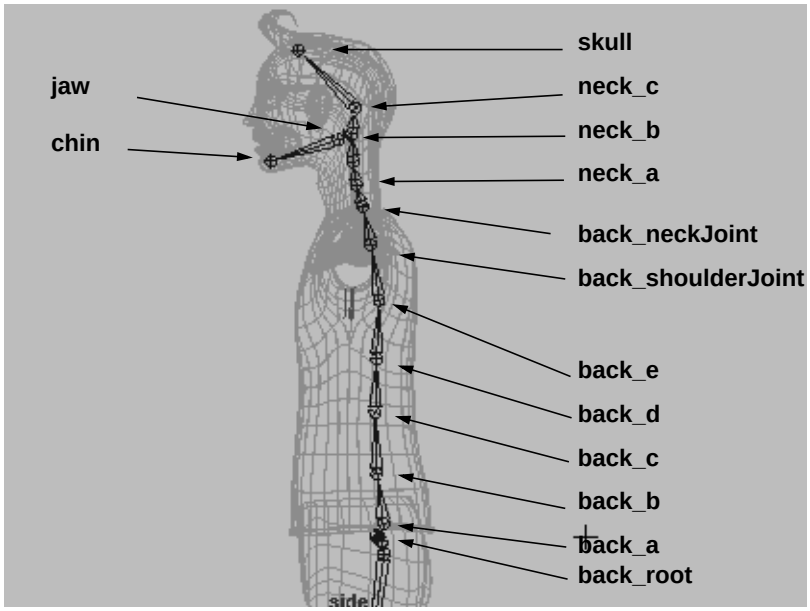


First two backbone joints

2 Draw more joints in the back to complete the backbone

- Draw 6 to 12 more joints until you reach the base of the neck.

- Continue creating joints for the neck and head skeleton. From the last joint in the back, create 3 - 4 joints for the neck and one for the skull.
- After placing the *skull* joint, press the up arrow twice to go back to the middle neck joint.
- Draw two new branching joints for the jaw and the chin.
- Rename the back joints as follows:



Joint names for spine and head

Tip: There are no strict rules about how many joints are needed to create a backbone. It really depends on how you want to animate the motion in the back. You will be using an IK Spline solver for the back to simplify the control of such a large number of joints. Here are some guidelines for determining the number of joints to use in a typical biped character:
• backbone (6 - 12 joints)
• neck and head (4-6 joints)

3 Save your work

PARENTING SKELETONS

You now have three separate skeletons: one for each leg and the backbone / neck / head skeleton. These represent three different hierarchies. To create one branching hierarchy you need to connect the three hierarchies using the parent command. You will start connecting these skeletons by connecting the hip joints to the backbone. Remember that although you are using the term "connect" you are parenting joints. Connecting joints in Maya is a different operation that moves the joint to a position on top of the connected joint. This is much different then

parenting which connects the joints together with an intermediary bone segment.

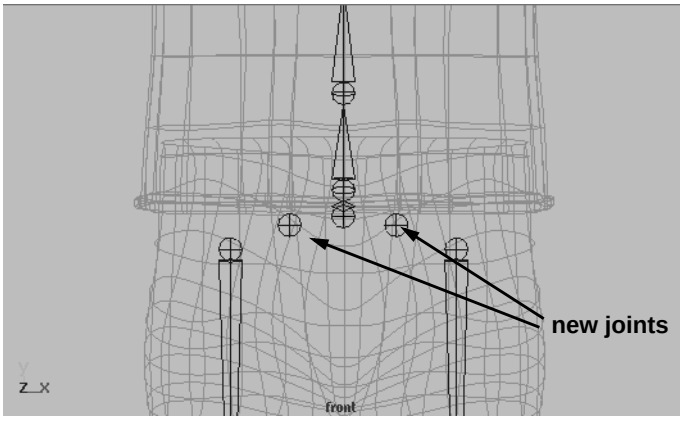
Attaching the Hips with a joint between

There are a several ways you can connect the hips to the backbone. The method you use depends on what the animation requires and how much control you need.

You can gain more flexibility in the hips, and greater anatomical correctness, by adding an extra joint between each hip and the spine. In essence, you will create a pelvis for the skeleton. Later you will create a similar situation when you connect the shoulders to the back. By creating this *pelvis*, you can rotate the hip / pelvis area independent of the backbone.

1 In a front view, create two new joints

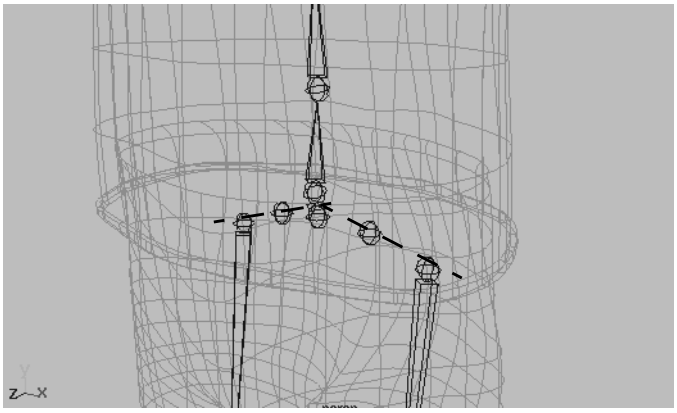
- Select **Skeleton** → **Joint Tool** and place one joint between *left_hip* and the bottom of the spine.
- **Mirror** this joint around the YZ plane.
- Rename the joints *left_pelvis* and *right_pelvis*.



New joints

- In the side / perspective view, confirm the two pelvis joints are in line with the hip and *back_root* joints.

Tip: You may want to open up a Hypergraph view so that you can see how the skeleton hierarchy is developing. You may also want to use the Hypergraph's freeform mode to create a more intuitive layout of the nodes, making it easier to review them.



Perspective view of new joints

2 Parent the hip joints to the pelvis joints

- Select *left_hip* then **Shift-Select** *left_pelvis*.
- Select **Edit** → **Parent** or press the **p** key.
- Repeat these two steps for *right_hip* and *right_pelvis*.

3 Parent the pelvis joints to the back's root joint

- Repeat these steps to parent *left_pelvis* and *right_pelvis* to *back_root*.

Tip: You may want to explore using the Hypergraph view to parent these joints. Click drag with your middle mouse button to drag the child node onto the intended parent.

---

4 Test the leg

- Test the leg behavior by rotating the pelvis joints. Use the pelvis joint to rotate sideways and the hips to rotate forward and back.

5 Reset the leg

- When you are finished testing the leg, select the *back_root* joint.
- Select **Skeleton** → **Assume Preferred Angle**.

6 Save your work

Summary

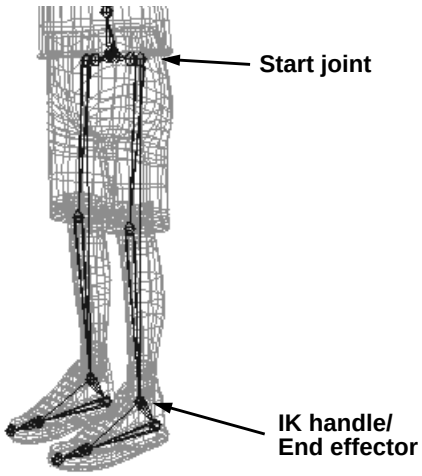
You have now completed the following tasks:

- created a skeleton hierarchy for a human bi-ped torso, with legs, feet, back and head that match the character's geometry. Later you will skin the geometry to the joint hierarchy so that they work together.
- set your joint names in an organized manner to help navigate the skeleton hierarchy as you work

  Later you will skin the geometry to the joint hierarchy so that they work together.

2 Inverse Kinematics

In order to make it easier to animate Melvin's skeleton, you will use **Inverse Kinematics** to help drive the motion of your joint chains. Inverse Kinematics can place your character's feet on the ground and keep them planted there while you move the hips. This kind of control is very difficult with **Forward Kinematics**.



Melvin's legs with IK handles

Set up Melvin's legs starting with simple IK solvers that are then grouped into a more complex hierarchy for animating the foot. The goal is to create a simple control for driving the roll of the foot from heel to toe. Set Driven Key will be used to establish how this control works with the foot hierarchy.

In this lesson, you will learn the following:

- How to set up IK Single Chain solvers on Melvin's legs and feet
- How to group the IK handles to help control the feet
- How to set up a heel to toe foot using Set Driven Key
- How to add an attribute to drive the heel to toe rolling motion
- How to apply selection handles to important nodes

Lesson 2

FORWARD VS. INVERSE KINEMATICS

In the last lesson, you learned that you can use **Forward Kinematics** to rotate joints one at a time. The resulting poses can be keyframed by setting keys on the joint rotation channels.

While the use of Forward Kinematics is very powerful, it has some limitations when animating a character. Since all of the animating is accomplished using the rotation of joints, it is not possible to take a joint lower down in the hierarchy and *fix* it in space. For instance, if you were to rotate a character's foot so that is sits on the ground, any movement in the pelvis area would move the foot out of place.

Inverse Kinematics solves this problem. Inverse Kinematics lets you control a series of joints using an IK handle. Moving either the handle or an upper start joint, evokes an Inverse Kinematic solver that calculates the joint rotations for you. Maya contains three main IK solvers that you will learn about in this course:

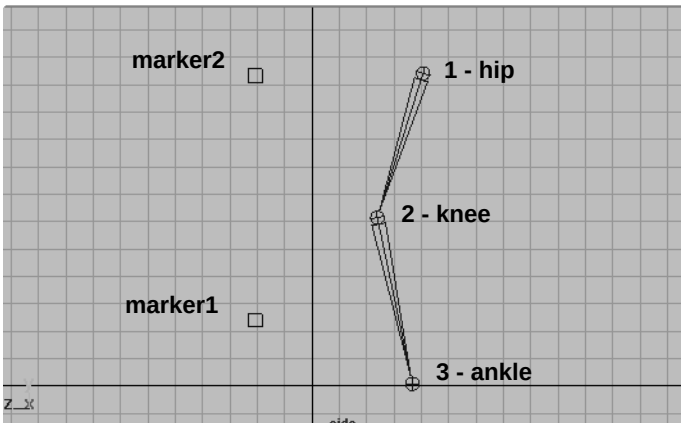
- **IK Single Chain solver** - This solver provides the simplest solution. By moving the IK handle, the chain will update so that the joints lie along a fixed plane. You will use the IK Single Chain solver in this lesson to set up Melvin's legs.
- **IK Rotate Plane solver** - This solver gives you more control over the position of the intermediate joints. You can use the IK handle so that the joints lie along a plane and then you can rotate the plane using a *twist* attribute or by moving a *pole vector* handle. The IK Rotate Plane solver will be explored in Lesson 3 to set up Melvin's arms.
- **IK Spline Solver** - This solver lets you control the joint rotations using a spline curve. You can either move the chain along the curve or update the shape of the curve using its CVs. The IK Spline solver will be discussed in Lesson 4 to set up Melvin's back.

A simple leg example - Forward Kinematics

The following example shows a simple leg being controlled by Forward Kinematics. Be sure to note what happens to the feet as you move the hip.

1 Create 3 joints to represent a hip, knee and ankle

- In the side view panel, draw three joints as shown in the following illustration.
- In front of the joints, place and scale two primitive **Cubes** to act as positioning markers for the joints. These will help you visualize what is happening as you work with the joints.

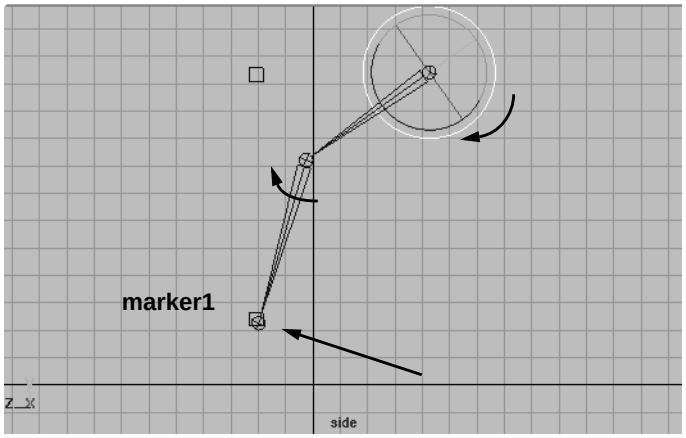


Three skeleton joints and positioning markers

2 Rotate the joints

- **Rotate** the hip and knee joints so that the ankle joint is positioned at marker 1.

  With Forward Kinematics, you must rotate the joints into place to position the ankle.

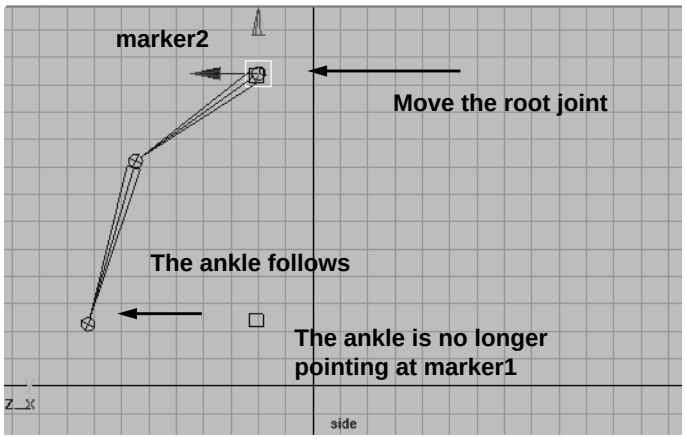


Positioned skeleton

3 Move the hip joint

- **Move** the *hip* joint forward to place it on marker 2.

  You will see how the knee and ankle joints also move. Now the ankle joint is no longer pointing towards the first marker. You would have to rotate the joints back to put the ankle back into its previous position.



Moving the hip joint

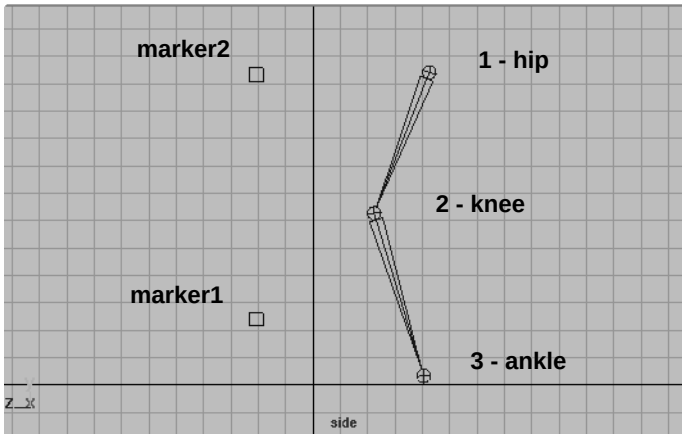
Compare this to the IK solution outlined below:

A simple leg example - Inverse Kinematics

The following example shows a simple leg being controlled by the IK Single chain solver. Be sure to observe what happens both when you move the leg and when you move the foot.

1 Create 3 joints to represent a hip, knee and ankle

- Draw another three joints as shown below:



Three skeleton joints

2 Add a Single Chain IK Handle

- Select **Skeleton** → **IK Handle Tool** - ❒. In the option window, set the following:

  Current Solver to **ikSCsolver**.

- Click first on the *hip* joint to establish the root of the solver.
- Click next on the *ankle* joint to place the IK handle.

Set Key hotkeys

In Maya, there are default hotkeys for all the active channels, and for the translate, rotate and scale channels on their own:

Set key

- Press **s** to set keys on all active channels displayed in the channel box.

Translate only

- Press **Shift w** to set keys on all the translation channels.

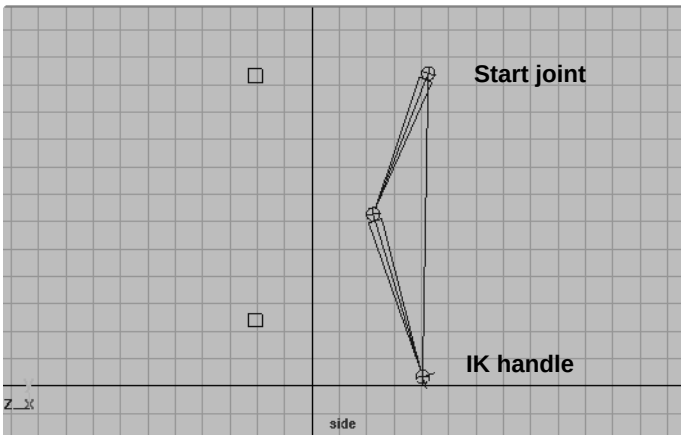
Rotate only

- Press **Shift e** to set keys on all the rotation channels.

Scale only

- Press **Shift r** to set keys on all the scale channels.

You may notice that the last three hotkeys correspond to the hotkeys for Move (w), Rotate (e), and Scale (r).



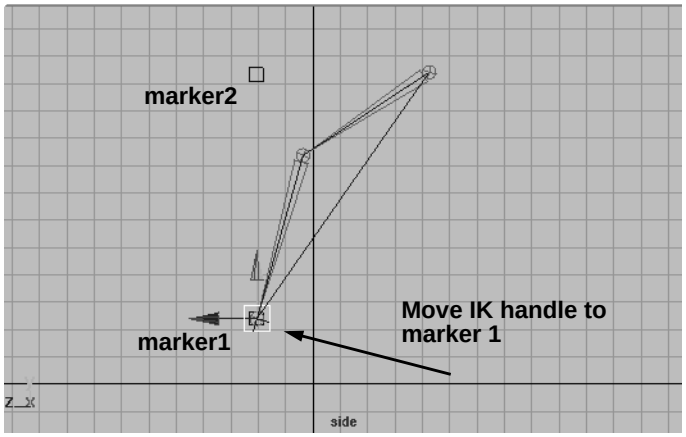
IK handle applied to leg

3 Move and key the IK handle

- **Move** the IK handle to the right so that it is placed on marker 1.

  Notice how the knee and hip joints rotate so that the end effector, currently at the ankle, is always at the ankle. This is a very easy way to control the leg.

- Press **Shift w** to set keys on the translation channels of the IK handle.



Moved IK handle

4 Move the hip

- **Select** the *hip* joint.
- **Move** the *hip* joint to the right to place it at marker 2

  The IK handle keeps the ankle joint at the first marker as you move the hip forward. The ankle will remain with the IK handle until you pull the hip too far. Then the ankle joint will pull away from the IK handle.

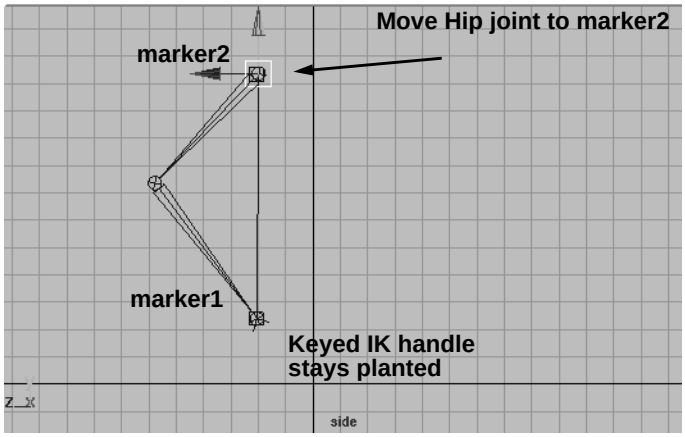
Note: The IK handle at the ankle is staying in place because it has been keyframed. Later in this lesson, you will learn how to set the stickiness of the IK handle so that it will stay in place without requiring keyframes.

Snapping hotkeys

To help you accurately place points in 3D, you can press and hold on the following hotkeys:

- For **Grid snap** press **x**.
- For **Curve snap** press **c**.
- For **Point snap** press **v**.

Once you release the hotkey the snapping is turned off.



Moved Hip joint

PREFERRED ANGLE

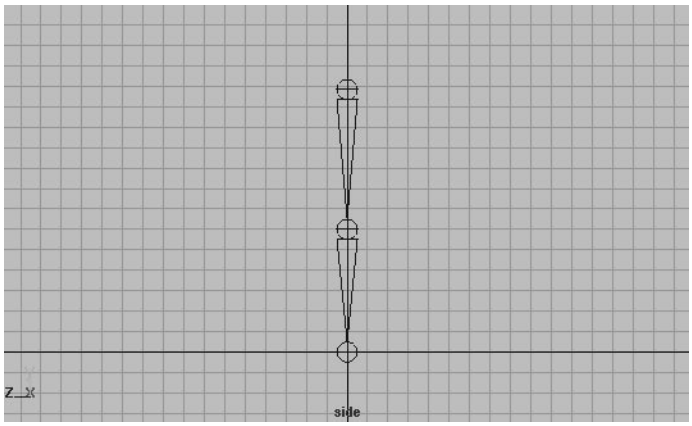
A joint's *preferred angle* establishes the direction a joint will bend when driven by an IK solver. It's sort of like a default bend direction. For example, if you create a knee joint straight up and down, then run IK through that joint and try to manipulate it, the solver will not be able to bend the joint. By setting the preferred angle, the solver has a guideline to follow.

Working with preferred angle

In the following example, you will explore a situation where the preferred angle must be altered.

1 Create 3 joints to represent a hip, knee and ankle

- Draw 3 joints a straight line using grid snap to draw them in so that there should be no bend in the knee.



New joints

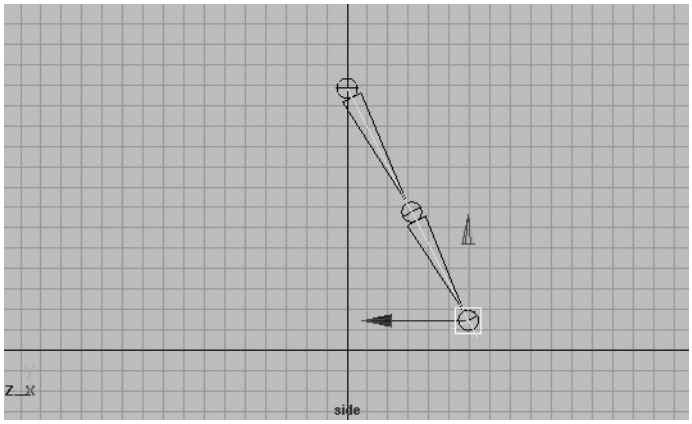
2 Add a Single Chain IK Handle

- Select **Skeleton → IK Handle Tool.**
- Select the *hip* joint as the root and then the *ankle* joint to place the IK handle.

3 Move the IK handle

- **Move** the IK handle to affect the chain.

  You should see that the knee does not bend. This is because there is no bend in the bones on either side of the knee. The solver is therefore not able to figure out which direction to bend.



Moved IK handle

4 Undo

- Undo the last move on the IK handle to return the chain to its original position.
- Delete the IK handle.

5 Edit the preferred angle on the knee

- **Select** the *knee* joint.
- **Rotate** the joint so that the knee is bent back.

RMB marking menu

You can use the RMB on a selected joint to access the **Set** and **Assume preferred angle** commands.

- Select **Skeleton** → **Set preferred angle - ❐.** In the option window turn **Selected joint** to **On**.

6 Add another IK Handle

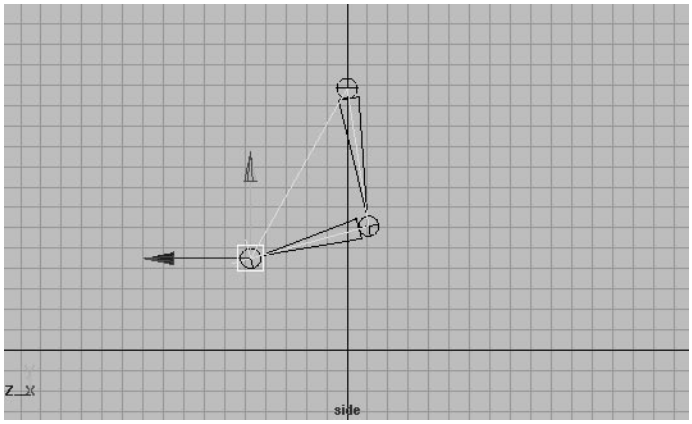
- Select **Skeleton** → **IK Handle Tool.**
- Select the *hip* joint as the root and then the *ankle* joint to place the IK handle.

  The reason that you deleted the IK Handle and created it again after you set the preferred angle is that the preferred angle will not be validated if an IK handle is present through the selected joints.

7 Move the IK handle

- **Move** the IK handle to affect the chain.

  Now the knee should be bending since the preferred angle gives the solver a clearer idea of which direction to bend.



Moved IK handle

8 Assume Preferred Angle

You can also use the preferred angle to return a skeleton to its starting pose.

- **Select** the *hip* joint.
- Select **Skeleton** → **Assume preferred angle**.

  The leg skeleton chain should reorient itself to its default position.

Tip: Assume Preferred Angle can be used as a tool to quickly set your character back to its default position.

STICKINESS

By default an IK handle will move when you move the root joint of the chain. Stickiness will help you keep the IK handle in one place. With a leg,

the foot should stay planted on the ground and not move if the person moves their hips.

Setting stickiness

1 Move the top joint from the simple leg just created

The IK handle moves as if it were a child of the hip. If you look in the hypergraph you will see that this is not true.

2 Turn on stickiness

- **Select** the IK handle.
- Open the Attribute Editor
- Under **IK Handle Attributes**, set **Stickiness** to **Sticky**.

3 Move the hip joint again

You should see that the IK handle stays in one place, as you would expect the foot to do.

Tip: Stickiness can be on when you create the IK handle if you turn the option on in the IK handle option window. Also remember that once keys are set on an IK handle then it will behave in a similar manner.

IK PRIORITY

IK Priority is the order in which IK solvers are evaluated. A solver with a priority of 1 is evaluated before a solver with a priority of 10. This is important to keep in mind as you build up the controls for a character. An IK solver in the hand or fingers should be evaluated after the IK solver in the arm. The joints in the finger are lower in the skeleton hierarchy so they depend on the joints in the arm for their placement.

If it seems that an IK chain is not updating properly in the interactive display or you notice differences between your interactive and your final rendering, you should check the IK priority of the solvers.

IK Priority can be set at the time of creation or can be changed later through the Attribute Editor.

Changing IK priority

To change IK priority for an individual handle:

- **Select** an IK handle.
- Open the Attribute Editor. IK priority is found in the **IK Handle Attributes** section.

Tip: To see and change the priority of more than one IK handle at a time you can use the Attribute Spreadsheet.

IK priority can be changed for an entire character in one pass with a MEL command.

- **Select** all the IK handles for the character.
- Enter the following command:

```
ikHandle -edit -autoPriority;
```

edit will put the command into edit mode.

autoPriority will automatically prioritize the selected IK handles based on their position in the hierarchy.

SET DRIVEN KEY

Throughout this course you will make extensive use of Maya's *Set Driven Key* tool. Set Driven Key (SDK) is used to establish a relationship between two or more objects. More specifically it is used to control an object's attributes based on "keyed" values of another object's attributes. This is the common use of Set Driven Key but the possibilities are quite limitless. You could, for example, establish a Set Driven Key relationship between different attributes on the same object.

Set Driven Key relationships will appear in the Graph Editor and are editable curves in terms of Value and Tangency.

This lesson will use SDK to build a "heel to toe" *roll* of Melvin's foot and establishing links and limits of movement between a hierarchical grouping of IK chains. Before starting this foot setup, here is a simple example of this method.

Rolling cube - Set Driven Key example

In this exercise you learn how to create an attribute that will control the rocking of a cube from side to side.

1 Create a new scene file

- Select **File** → **New Scene.**

2 Create a primitive poly cube

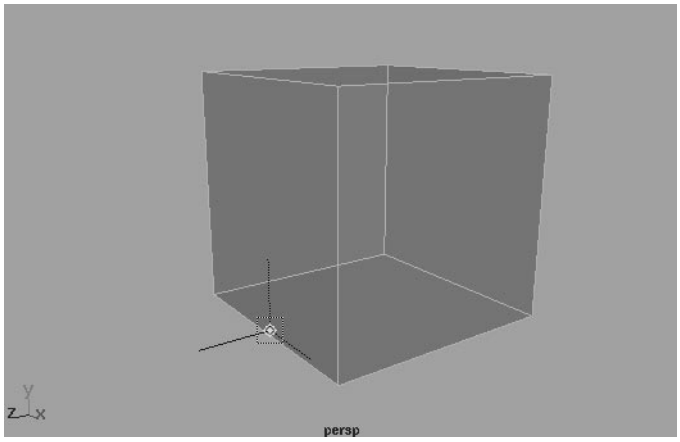
- Select **Create** → **Polygons Primitives** → **Cube.**

3 Create a group node above top of the cube

- Select the poly cube then select **Edit** → **Group.**
- Rename this new group *frontPivot.*

4 Move the pivot of the new node to the front of the cube

- Select the *frontPivot* group.
- Select the **Move** tool.
- Press the **Insert** key to get access to its pivot manipulator.
- Select the Z and Y manips to translate the pivot to the front and bottom of the cube. Use the side orthographic view to get proper placement.



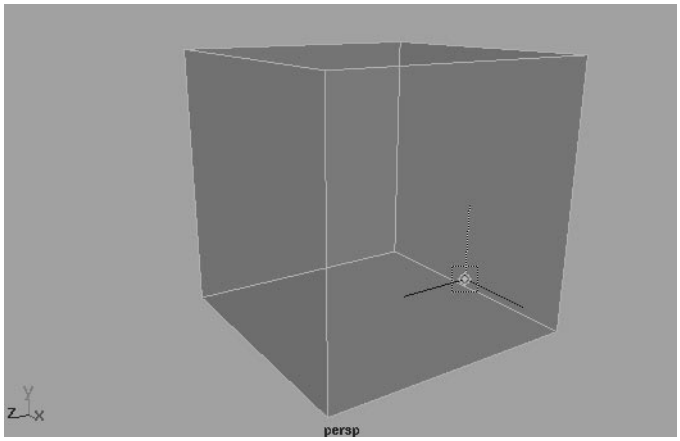
Moving the frontPivot

5 Create a new group for the frontPivot

- Select the *frontPivot* group and select **Edit** → **Group.**
- Rename this group *backPivot.*

6 Move the pivot to the other side of the cube

- Press the **Insert** key and translate the *backPivot* pivot to the back and bottom of the cube.



Moving the backPivot

7 Create a node above backPivot

- Select the *backPivot* group and select **Edit** → **Group.**
- Rename this group *cubeControl.*

8 Add an attribute to cubeControl

- Select the *cubeControl* node.
- Select **Modify** → **Add Attribute...** and set the following

  Attribute Name to **Roll**;

  Make Attribute Keyable to **On;**

Data Type to **Float;**

Attribute Type to **Scalar;**

Minimum value to **-10;**

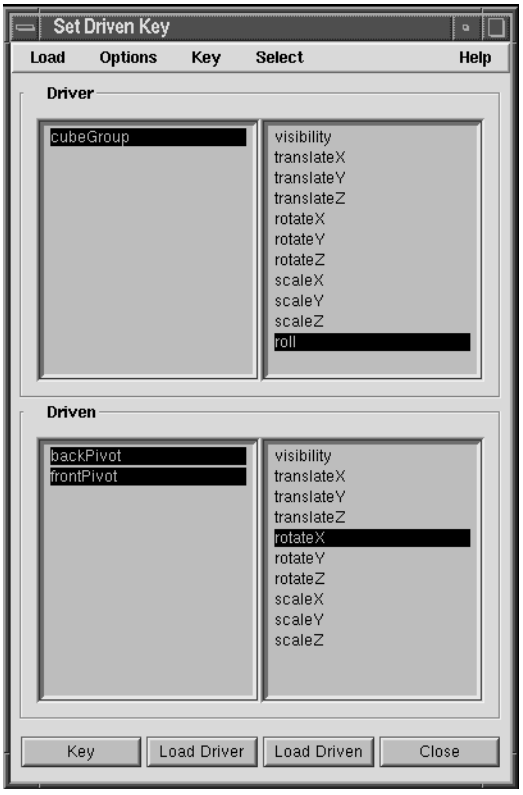
Maximum value to **10**

Default value to **0**

- Press **OK**

9 Create a Set Driven Key relationship

- Open the Set Driven Key window by selecting **Animate** → **Set Driven Key** → **Set** - ❐.
- Select the *cubeControl* group and press **Load Driver.**
- Select the *Roll* attribute in the right-hand side list of Driver attributes.
- Select the *frontPivot* and *backPivot* groups and press **Load Driven.**
- Select the *rotateX* attribute in the right-hand side list of Driven attributes.
- **Shift-select** both the *backPivot* and *frontPivot* objects in the left column so that both are selected.



Set Driven Key window

10 Set a driven key for the cube sitting flat

- Verify the following settings:

  Roll attribute at **0**;

  rotateX for *frontPivot* and *backPivot* at **0**;

- Press **Key.**

  This will set an SDK on both groups for the *Roll* attribute at **0**. This will be the default position of the cube as the *Roll* attribute's default setting is **0**.

Tip: You can do your object selection from within the SDK window by selecting the object in the left fields of the window.

---

11 Set a driven key at 45 degrees

- In the SDK window, select the *CubeControl* group.
- In the Channel Box *Roll* attribute text field, enter **-10.**
- In the SDK window, select the *frontPivot* group.
- In the Channel Box *rotateX* text field, enter **45.**
- Press **Key** in the SDK window.

12 Set a driven key at -45 degrees

- In the SDK window, select the *cubeControl* group.
- In the Channel Box *Roll* attribute text field, enter **10.**
- In the SDK window, select the *backPivot* group.
- In the Channel Box *rotateX* text field, enter **-45.**
- Press **Key** in the SDK window.

13 Test out the setup

Test by highlighting the *Roll* attribute and middle mouse dragging in the viewport. You should see the cube rocking back and forth.

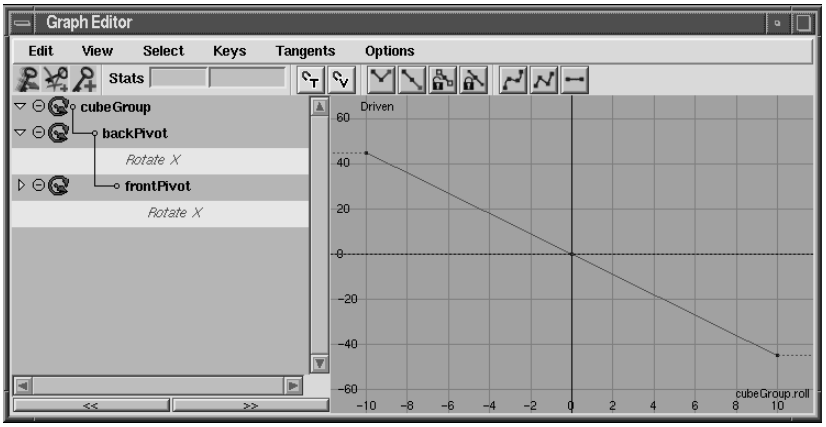
How it works

The Key that was established at **Roll 0** and **RotateX 0** for both pivots is the handoff between which pivot is working with respect to the value of the **Roll** attribute. Positive values drive the *backPivot* **rotateX** and negative values drive the *frontPivot* **rotateX**.

Look at this in the Graph Editor

1 Open the Graph Editor

- Select **Window** → **Animation Editors** → **Graph Editor...**
- Select the *cubeControl* group and open up the hierarchy in the Graph Editor. You will see the curves that are displayed for the *RotateX* attributes.
- Select the individual *Rotate X* curves.



SDK curve in the Graph Editor

Tip: You can traverse a hierarchy by using the up and down arrows on the keyboard.

Notice that in the Graph Editor when you are looking at SDK curves you are no longer working with a time based relationship. You are looking at a relationship between values of the attributes involved.

2 Modify the tangency of the curves

- Left mouse drag the *frontPivot* **-10** key.
- **LMB-drag** the out tangent handle.
- **MMB-drag** to change the out tangency.

  Use **Keys** → **Break Tangents** and **Keys** → **Free Tangent Weights** to get individual control of the in and out tangent curve handles

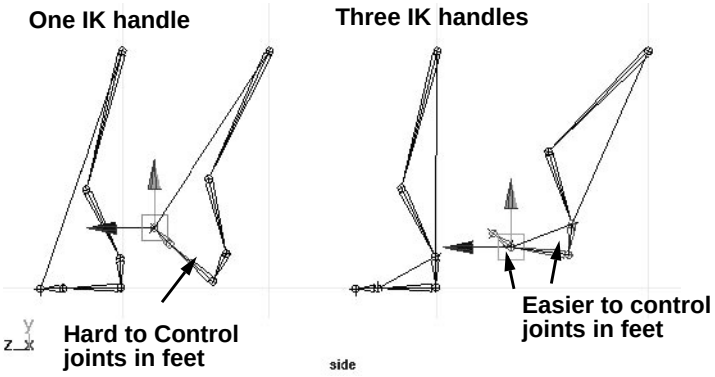
Note: If you can't free the weight of your tangents, you might need to change your General Preferences. Go to **Options** → **General Preferences** and under the Animation tab, check on **Weighted Tangents**.

Experiment with different tangent shapes and note the resulting behavior of the cube and the *Roll* relationship.

ADDING IK TO MELVIN'S LEGS

You will now use Inverse Kinematics to set up Melvin's legs. Several IK Single Chain solvers will help define the motion between the hip and the ankle of each leg, the ankle and ball and ball and toe.

It seems like a good idea to create a skeleton with a single IK handle that flows from the hip to the toe as a quick way to set up a leg. The problem is that this kind of setup makes it hard to control the joints in the feet since you are relying on the IK solver to calculate all the rotations.



Different IK chains

A better way of setting up Melvin's legs would be to use several IK handles to control different parts of the leg. One chain will work from the hip to the ankle and two more will help define Melvin's foot.

The foot IK handles can then be grouped into a more complex hierarchy to create a heel to toe motion. You will be able to use **Set Driven Key** to make a single attribute to drive the roll of the feet.

Create the IK handles

You will start building Melvin's leg controls using three IK Single Chain solvers on each leg.

1 Open an existing scene file

- Open the file *Melvin_02_legs.mb* from the scenes directory.

2 Set up an IK Single Chain solver on Melvin's legs

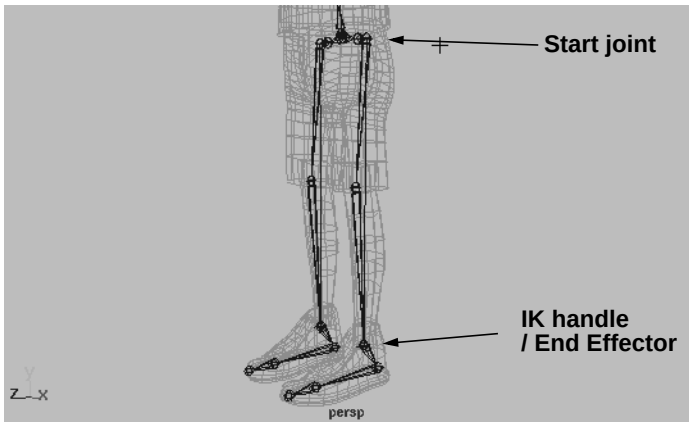
- Select **Skeletons→ IK Handle Tool** - ❐. In the Option window, set the following:

  Current Solver to **ikSCsolver**.

- Select the *left_hip* to establish the start joint of the IK chain and then the *left_ankle* to establish the end effector of the IK chain.

  Notice that an IK chain with joint effectors is automatically created. The end effector is the transform position of the IK handle.

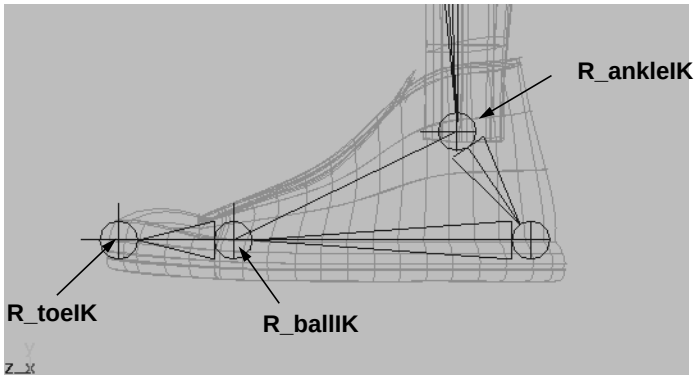
- Repeat for the right leg.
- Rename the IK handles *L_ankleIK* and *R_ankleIK*.



IK added to the legs

3 Create IK chains for the two feet

- Select **Skeleton** → **IK Handle Tool**.
- Click on the *ankle* joint and then the *ball* joint of the left leg. Name this IK handle *L_ballIK*.
- Click on the ball and then the toe of the left leg. Name this IK handle *L_toeIK*.
- Repeat for the right leg.



Foot IK handles

Restricting the heel rotation

Degrees of freedom define which axes a joint may rotate in. With Melvin's foot, the heel joint should not rotate. This joint has been drawn more as a reference so that you can see where the heel should be when you place the feet. To make sure that it doesn't rotate when the *R_ballIK* handle is animated, you will lock it for all three axes.

1 Turn off degrees of freedom for each axis

- **Select** the left heel joint.
- Open the Attribute Editor.
- In the **Joint** section, turn the **X**, **Y** and **Z** degrees of freedom to **Off**.

- Repeat this process for the right heel.

  Now these joints will not rotate as part of any of the IK solutions.

Note: The degrees of freedom cannot be changed on a joint that is the root of an IK chain.

---

2 Save your work

Build a foot control hierarchy

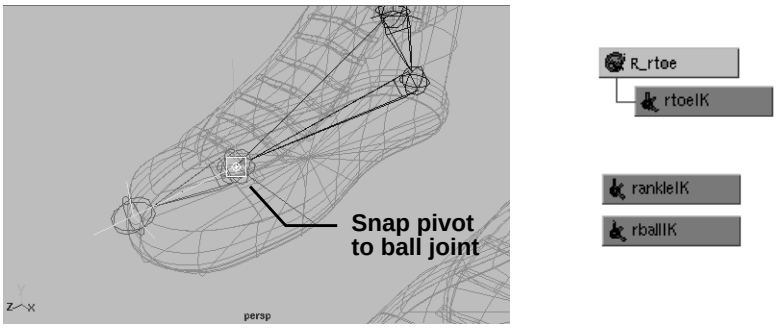
The IK handles will be grouped into a hierarchy so that one node will control many. When the foot is completed, it will roll from heel to toe. Start with the right foot.

1 Group the right toe's IK handle

- Select the *R_toeIK* IK handle.
- Select **Edit** → **Group** - ❐**.** In the Option window, set the **Group pivot** to **center**.

  Grouping creates a new node above the IK handle. The pivot of this new node will now be put near the IK handle, instead of at the origin (0, 0, 0).
- Rename this node *R_toe.*
- Open a Hypergraph panel.

  This will help you visualize the hierarchy as you build it out of the grouped IK handles.



View of pivot and hypergraph view of grouped nodes

2 Move the pivot to the right foot's ball joint

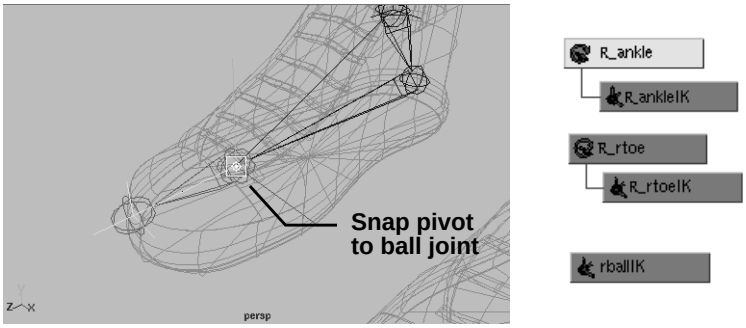
- Select the **Move** tool.
- Press the **Insert** key to move the pivot.

  You will see that the manipulator changes to indicate that you are editing the pivot instead of the object.
- Press the **v** key to temporarily turn on point snapping then move the pivot with **MMB** to the ball joint of the right foot. As you get close, the point snapping will lock the pivot to the joint.

- This new pivot location is designed to control the rotation of the toe IK handle around the ball of the foot.

3 Repeat the same procedure for the right ankle's IK handle

- Select the *R_ankleIK* IK handle.
- Select **Edit** → **Group**.
- Rename this node *R_ankle*
- Select the **Move** tool.
- Press the **Insert** key to move the pivot.
- Use point snapping to move the pivot to the *ball* joint.

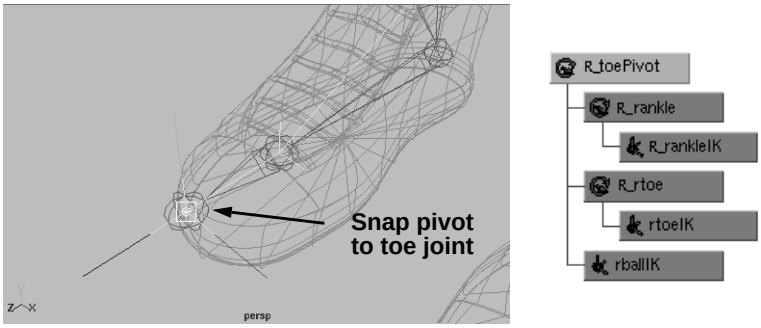


View of pivot and hypergraph view of grouped nodes

4 Make one group of the foot

- Select the *R_ankle*, *R_toe* and *R_ballIK* nodes. You may want to use the Hypergraph to help you.
- Select **Edit** → **Group**.
- Rename this node *R_toePivot.*
- Use point snapping to move the pivot to the *toe* joint.

  This new group will control the rolling of the foot from the toe. The toe leaves the ground last so you need a pivot there.



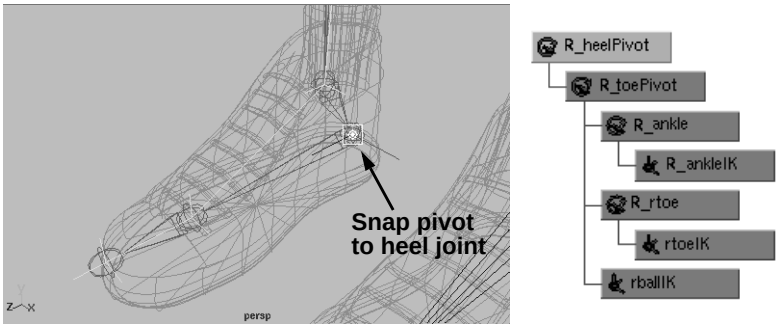
View of pivot and hypergraph view of grouped nodes

5 Group the whole hierarchy

- With the *R_toePivot* node selected, select **Edit** → **Group**.

- Rename this node *R_heelPivot*.
- Use point snapping to move the pivot to the *heel* joint.

  This node will control the rotation of the heel as it contacts the ground and brings the rest of the foot down to the ground.

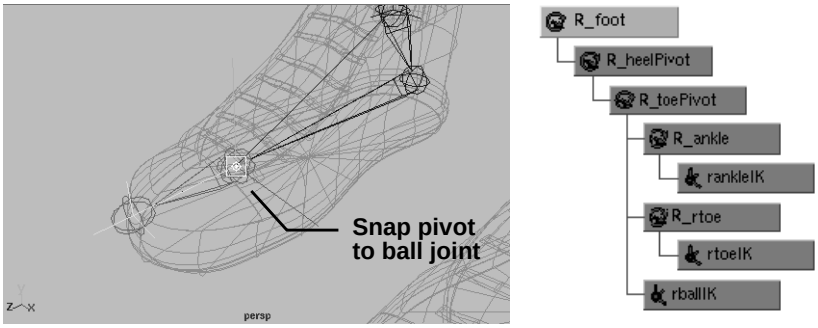


View of pivot and hypergraph view of grouped nodes

6 Group the hierarchy again

- With the *R_heelPivot* node selected, select **Edit** → **Group**.
- Rename this node *R_foot*.
- Use point snapping to move the pivot to the *ball* joint.

  This node will be the main foot control and will pivot the foot at the ball. Most foot pivoting happens around this point.



View of pivot and hypergraph view of grouped nodes

7 Set up the left foot hierarchy

- Repeat steps 1 to 7 shown above for the left foot. Be sure to change the **R_** prefix to an **L_** for the new group nodes.

8 Save your work

Creating a control attribute

Now that the control hierarchy is setup for both feet, you will create the actual controls using a new attribute and Set Driven Key. An attribute named *roll* will be added to both hierarchies then the various nodes in the hierarchy will be rotated to create the heel to toe motion.

1 Add a Roll attribute to the rfoot node

- Select the *R_foot* node.
- Select **Modify** → **Add Attribute**...
- In the Add Attribute window, set the following:

  Attribute Name to *roll*;

  Data Type to **Float**;

  Minimum Value to **-5**;

  Maximum Value to **10**;

  Default Value to **0**.
- Click **OK**.

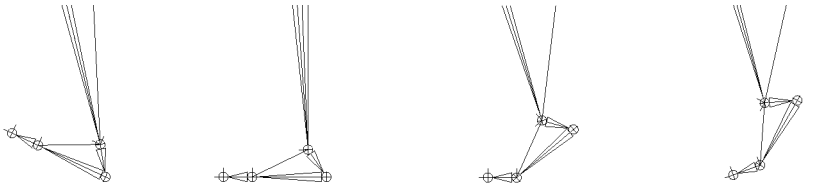
  You should see a new attribute in the channel box called *Roll*. It is set to a value of 0 but can be set to any value between -5 and 10. Right now the attribute is not connected to anything. Set Driven Key will be used to make it functional.
- Repeat these steps to add a roll attribute to the *L_foot* node.

Note: Even though you entered an attribute name of *roll*, it appears capitalized in the Channel box. The channel box can show words in 3 different ways: Nice, Long, and Short. If the attribute translateX was displayed in the "Nice" setting, it would look like "Translate X." In the Long setting, it would look like "translateX" and the short setting would display it as "tx." You can change these settings within the Channel Box by selecting **Channels**→ **Channel Names** and selecting one of the three.

Set Driven Key

The Set Driven Key function allows you to use a curve to define a relationship between two attributes. It is important to note that Set Driven Key creates keys which are not dependent on time. Instead, these keys are dependent on the relationship between two attributes.

Set Driven Key will now set up the *roll* attribute to drive the rotation of various nodes in the foot hierarchy. The final result will be the proper rolling of the foot. When the *roll* is set to *-5*, the whole foot will rotate around the heel. When the *roll* is set to *0*, the foot will be flat with no rotation. When the *roll* is set to *5*, the heel will be rotated around the ball of the foot. When the *roll* is set to *10*, the whole foot will be rotated around the toe.



Roll = -5 **Roll = 0** **Roll = 5** **Roll = 10**

Diagram of heel to toe rotations

1 Setup the driver and driven nodes

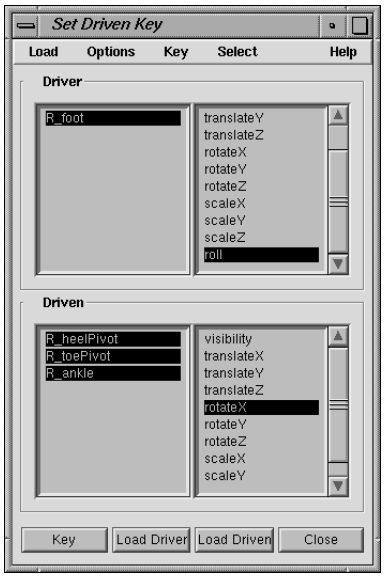
- Select **Animate** → **Set Driven Key** → **Set -** ❐.
- Select the *R_foot* node.
- Click on the **Load Driver** button.
- Select the *R_heelPivot*, *R_toePivot* and *R_ankle* nodes.
- Click on the **Load Driven** button.

2 Set Driven Key for everything in the default position

Driven keys will now be set. The keys that are set should be thought of as extremes.

- Make sure that the **X rotation** on all the driven nodes is **0** and that the *R_foot's* **roll** attribute is also **0**.
- In the right hand list under Driver, select the **roll** attribute.
- Under **Driven**, select all of the nodes on the left hand list.
- Select **X rotation** in the right hand list. This selects the attribute for all of the selected nodes.
- Click on the **Key** button.

  Now the **roll** attribute is driving the **X rotation** of the other nodes.



Set Driven Key window

3 Set the key for the heel contacting the ground

- Click on the *R_foot* node in the Set Driven Key window. This selects it and places it in the channel box.
- In the Channel box, change the **Roll** to **-5**.
- Click on the *R_heelPivot* node in the Set Driven key window.
- **Rotate** the *R_heelPivot*'s **X rotation** to **-25**.
- Click on the **Key** button.

4 Set the key for the heel leaving the ground

- Change the **Roll** to **5**.
- Change the *R_ankle* node's **X rotation** to **40**.
- Click on the **Key** button.

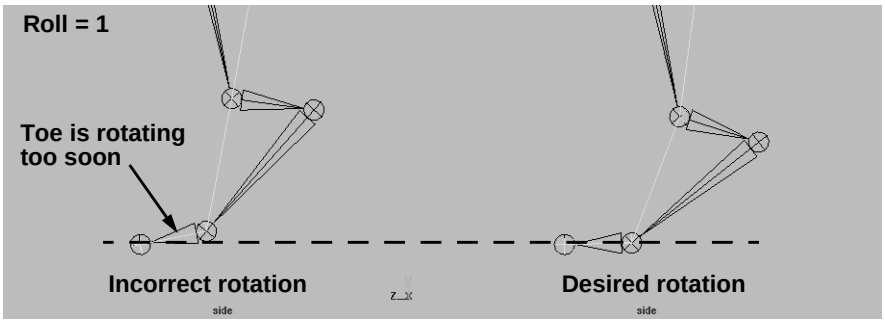
5 Set the key for the ball leaving the ground

- Change the **Roll** to **10**.
- Change the *R_toePivot* node's **X rotation** to **20**.
- Click on the **Key** button.

6 Test the foot roll

- Click in the **Roll** channel in the Channel box.
- Click-drag with your MMB in the modeling window to see the foot roll.

  The roll is almost correct except that the toe starts to pivot too soon. Rather than the ball of the foot staying flat until the toe's rotation starts, it starts lifting too soon. This can be fixed by editing the animation curve for the *R_toePivot* node's keys.

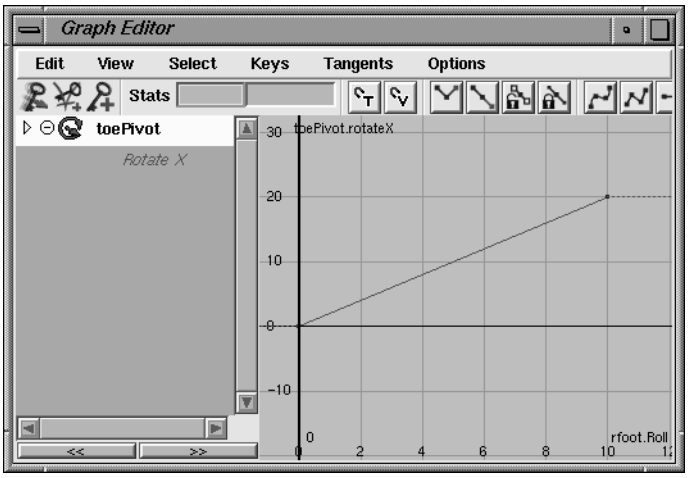


Incorrect rotation of the toe

7 Fix the keys on the toePivot

- Select the *R_toePivot* node.
- Select **Window** → **Animation Editors** → **Graph Editor**...
- Click on the **X rotate** curve then select **View** → **Frame All**.

  The curve you are looking at maps the X rotation of the *R_toePivot* node to the *R_rfoot.roll* attribute. From this graph, you can see that the rotation of the *R_toePivot* starts when the roll attribute is 0. You want the rotation of the *R_toePivot* to wait until the *R_ankle* rotation is finished. Therefore you want the *R_toePivot* rotation to start when the value of the **roll** attribute is 5.

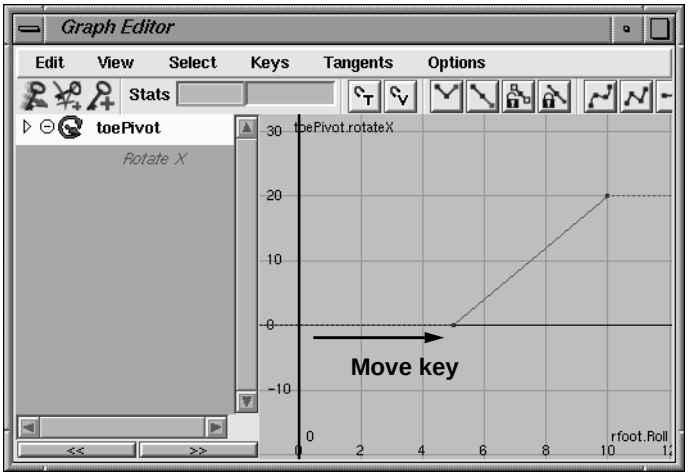


Graph editor view of toePivot

- Select the first keyframe.

  The stats should say 0 for the roll value and 0 for the X rotation value.
- Change this to read **5** and **0**.

  Now the rotation of the toe will not start until the roll has reached 5.



Corrected animation curve

8 Test the foot roll

You now have good control over a foot control that creates a convincing heel to toe motion. This approach is not the only way to do a foot but should give you a good idea of how to approach a foot and create your own control setup.

9 Repeat this procedure to the left foot

10 Save your work

This setup could be taken further to control the toe bending to the ground as the foot lifts.

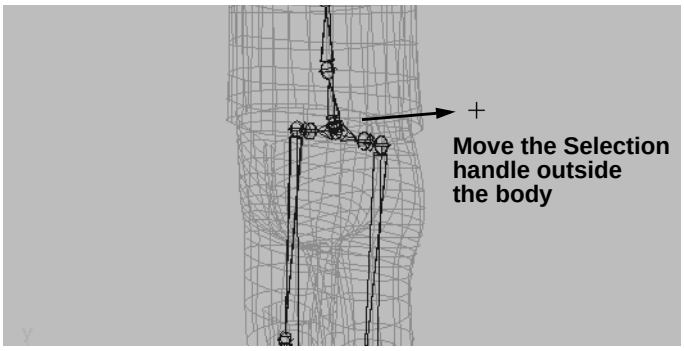
Adding Selection handles

Selection handles make it easier to pick important joints on your skeleton. You will probably want to put selection handles on the *back_root* joint, as well as the *R_foot* and *L_foot* control nodes.

1 Add a selection handle to the root joint

- **Select** the *back_root* node.
- Select **Display** → **Object Components** → **Selection Handles**.
- Press **F8** to go into component mode.
- Select **All Components Off** selection mask and turn on the **Selection Handles** selection mask.
- **Click-drag** a selection box around the root node's new selection handle to select it.
- **Move** the handle so that it lies outside of the body.

  This will make it easier to select later.



Selection handle on the root

2 Add selection handles to the feet

Repeat the steps outlined is step 1 to add selection handles to the *L_foot* and *R_foot* nodes. These are the nodes that contain the **roll** attribute.

3 Save your work

Test the setup

Now it's time to test your setup. You have three main controls:

- **L_foot** - controls the leg and the foot's heel to toe motion.
- **R_foot** - controls the leg and the foot's heel to toe motion.
- **back_root** - controls the root joint of the skeleton.

Go ahead and explore how these three controls work. Move Melvin forward and begin working with the roll of the feet. You even may want to set keys on the controls and begin animating Melvin. In Lesson 5 you will be creating a walk cycle but it doesn't hurt to get familiar with the setup now.

If anything is not working as you would expect then look back over this lesson to see if anything is missing.

Summary

You have now completed the following tasks:

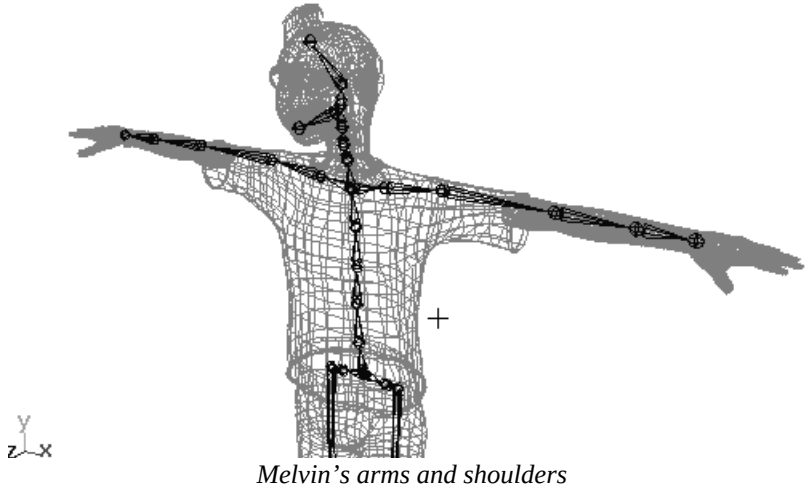
- Created IK handles for easily controlling the legs.
- Created a custom attribute and used SDK to create complex motion.
- Added selection handles for ease of selection.
- Used the freeze transformations tool to make a position easier to return to.

There are several other ways to setup a character's legs and feet. In the Appendix, you will find alternate methods of setting up Melvin's feet and legs. You may try these methods later to help you understand other techniques that can be used for character setup.

Lesson 3

3 Arms and shoulders

In order to animate Melvin's arms and shoulders, you are going to use the IK Rotate Plane solver to help you create the desired motion. To give you more control over how the elbows are working, this solver will be needed.



Melvin's arms and shoulders

Maya provides many ways of building characters. While three joints would be the easiest way of setting up Melvin's arm, you will explore a more complicated setup that uses an extra forearm joint. This set up will introduce several issues that will be dealt with throughout this course.

In this lesson you will learn the following:

- How to use an IK Rotate plane solver for arms
- How to set up the wrist twist in Melvin's forearm
- How to constrain the solver's Pole vector to help aim the elbow
- How to work with the solver's end effectors

IK ROTATE PLANE SOLVER

In the last lesson, you set up the legs using the IK Single chain solver. This let you easily control the forward motion of the leg but if you wanted to rotate Melvin's legs out you would find it difficult to control the position of the knee. For Melvin's arms, you want to be able to rotate the elbows out from the body. To do this you will need the IK Rotate Plane solver which has extra manipulator controls to help you define how the whole arm will work.

When you set up an IK Rotate Plane solver, the whole length of the chain is controlled by a plane that is defined by the *handle vector* which runs between the *start joint* and the *end effector*, and a secondary vector that is called the *pole vector*. The plane acts as the goal for all the joint rotations. By default, the IK handle will manipulate the chain so that it works within this plane. You can then rotate the plane by either editing a *twist* attribute or by moving the *pole vector handle*.

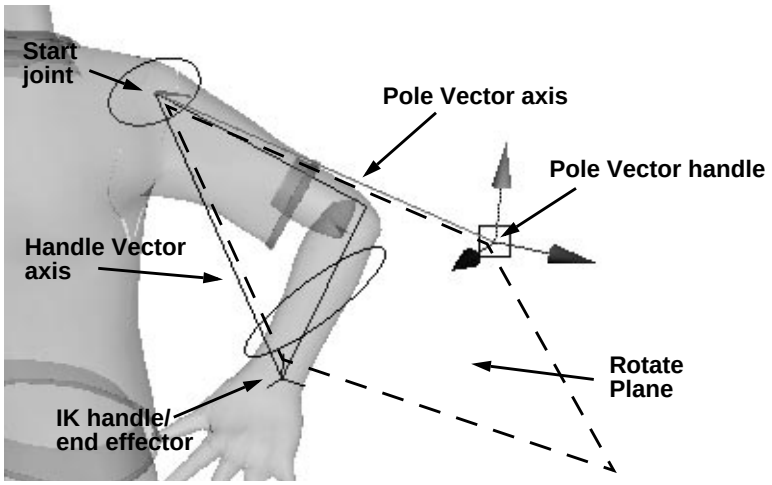


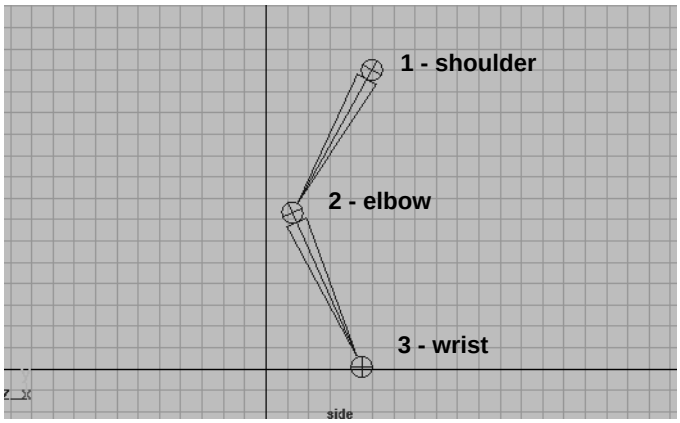
Diagram of IK Rotate Plane solver

Working with the IK Rotate Plane solver

The following example shows an arm being controlled by the IK Rotate Plane solver.

1 Create 3 joints to represent the shoulder, elbow and hand

- In the side view panel, draw three joints as shown in the following illustration.



Three skeleton joints

2 Add a Rotate Plane IK handle

- Select **Skeleton** → **IK Handle Tool** - ❐. In the option box, set the following:

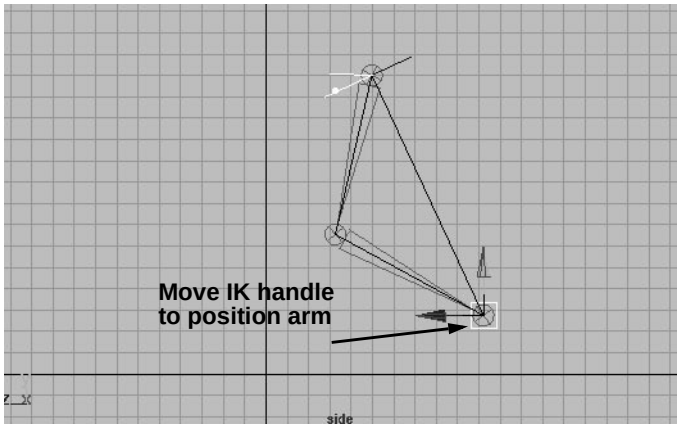
  Current Solver to **ikRPsolver**.

- Click on the shoulder joint to set the start joint of the IK handle.
- Click on the wrist joint to place the IK handle.

3 Move the IK handle

- Select the **Move** tool.
- **Move** the IK handle to the right.

  The IK handle is now working in a similar manner to the IK Single chain solver. Basic IK handle manipulation is the same for both solvers.



Moving the IK handle

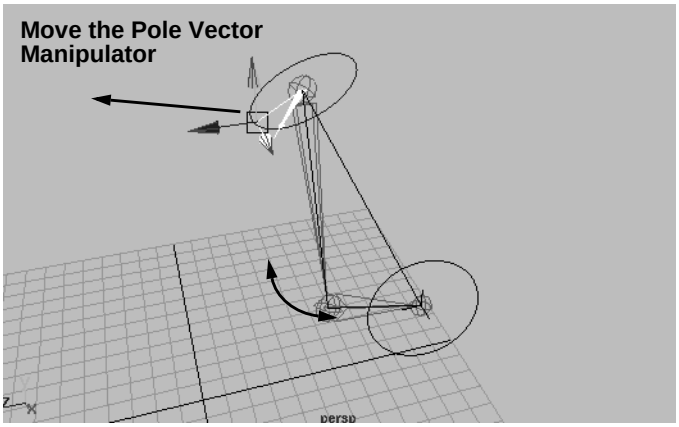
4 Control the handle's Pole Vector

- Select the **Show Manipulator** tool.

  A series of manipulators appear to let you control the IK handle's pole vector and twist.

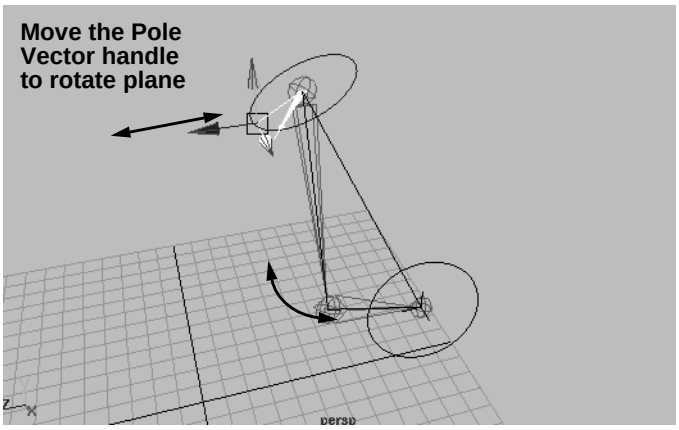
- **Move** the **Pole vector** manipulator behind the arm joints.

  This ensures the elbow is pointing back.



Rotate plane manipulators

- In the perspective view, click-drag on the **Pole vector's** Z-axis handle to rotate the IK solver's plane. For an arm, this lets you control how the elbows will animate in relation to the body.



Moving the Pole Vector handle

5 Manipulate other Rotate Plane solver controls

- In the Channel box, click on the IK handle's **Twist** attribute to highlight it.
- Click drag with your **MMB** to twist the IK solution away from the rotate plane.

  This is a sort of a rotation offset from the actual plane as defined by the pole vector location. You could use this attribute to rotate the elbows if you don't want to move the Pole vector.

How to avoid flipping in the arm

Flipping occurs when the end effector is moved through the plane. If you experience flipping, you can use the **pole vector axis** handle to move the

plane out of the way. You may need to set keys on this handle to control flipping in a more complex arm motion

Pole Vector constraints

To give you easy access to the pole vector, you can constrain it to a locator. In this way, you won't have to use the **Show Manipulator** tool in order to edit the pole vector location. For Melvin, you will use constrained pole vectors to work with his elbows.

MELVIN'S ARMS

You will now create Melvin's arms using skeleton joints and IK handles. Again, you will use the character's pre-made geometry to aid in placing of the joints. You will then define the control of the arms by setting the preferred angle for the elbows, adding IK Rotate Plane solvers and then constraining the solvers' pole vectors to locators.

Creating the arm joints and setting preferred angle

In this exercise you will explore an arm building technique that will set up sophisticated skinning characteristics in the forearm. You will do this by creating an extra joint between the elbow and the wrist. When you later skin and add flexors to Melvin, this extra forearm joint will cause the forearm to twist when the hand / wrist rotates.

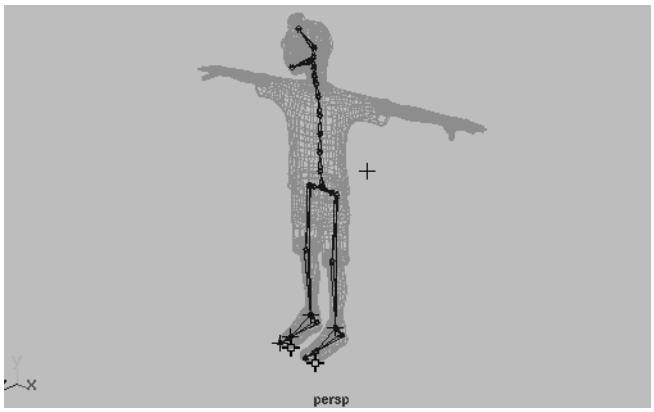
It is important to ensure that joints between the elbow and wrist are created in a straight line, making Melvin's arms as *anatomically correct* as possible.

You will build the arm joints with **Auto Joint Orient** set to **XYZ**. You will also manually set the preferred angle of the elbow joint after the joint has been created.

1 Open an existing scene file

- Open the scene file *Melvin_03_feet.mb*.

  This scene is identical to the scene that you created in the previous lesson.



Melvin skeleton without arms

Outliner tip

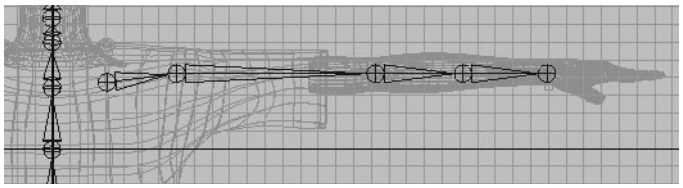
To see all the joints in a hierarchy in the outliner, **Select** the first and last joints then in the Outliner, select **Show** → **Selected.**

2 Confirm that all of the pieces are in the right places

3 Create five joints for the left arm

In the front view, you will start next to the shoulder joint and create a series of 5 joints along the extended arm.

- Select the **Skeleton** → **Joint Tool** - ❐ and confirm that **Auto Joint Orient** is set to **XYZ**.
- Place 5 joints along the extended arm as shown in the following figure.
- Press the **Shift** key as you place the successive joints to make sure that the elbow, forearm, and wrist joints form a straight line.

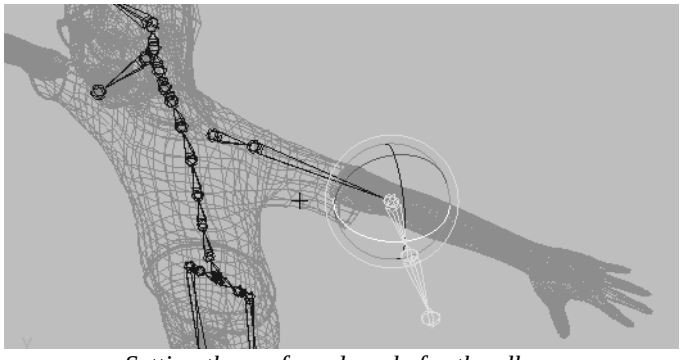


Arm joints

- Check side and perspective views for proper alignment. Rotate and scale as necessary. Do not translate.
- Rename the joints *left_collar*, *left_shoulder*, *left_elbow*, *left_forearm*, and *left_wrist*.

4 Set the preferred angle of the elbow

- Select the *left_elbow* joint.
- Bend the elbow by rotating the *left_elbow* joint around the Y axis until wrist is pointed forward in the Z direction.



Setting the preferred angle for the elbow

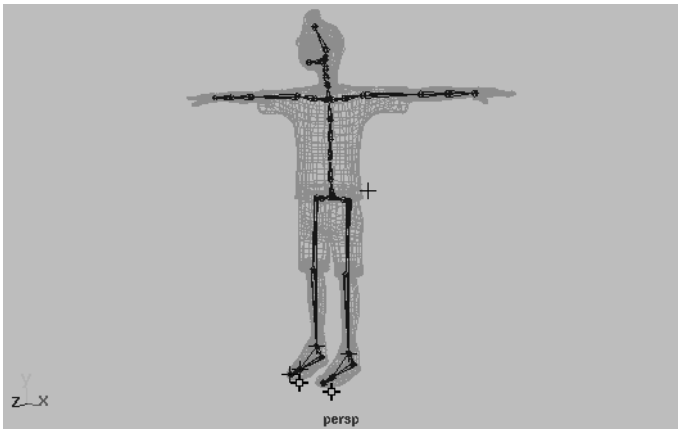
- Select the *left_elbow* and with the RMB, select **Set Preferred Angle**.
- Return the *left_elbow* joint to its rest position by setting the value of all the rotation channels in the Channel Box, to **0**.

5 Parent the collarbone joint to the shoulder joint

- **Select** the *left_collarbone* joint, then **Shift-Select** the *back_shoulderjoint* joint.
- Select **Edit** → **Parent**.

- Repeat steps 1 through 5 for the right arm.

Note: Don't forget to name the joints appropriately.



Completed arm joints

Tip: This process can be streamlined for the opposite arm. A mirror will not work properly because the local axes are not changed, however the mirror can be used to line up the new arm. Mirror the arm and then draw a new skeleton chain by using the **v** key to snap to joints. Then delete the mirror copy.

6 Save your work

Setting up the IK Rotate Plane solver

You will now use the IK Rotate Plane solver for the arm. Rather than placing the IK handle on the wrist joint, you will use the forearm joint. You will then move the chain's end effector to the wrist. This will prepare Melvin's arm for rotating the forearm later.

1 Setup RP IK from the left shoulder to the left forearm

- Select the **Skeleton** → **IK Handle Tool** - ❐.
- Make sure that **ikRPsolver** is selected.
- Create an IK handle from the *left_shoulder* to the *left_forearm*.
- Rename the IK handle *L_armIK* and the end effector *L_armEffector*.

  The end effector is grouped under the forearm joint. You will need to open up the Hypergraph or the Outliner to find this node.

Translating the end effector of the IK chain

The arm's IK handle was placed on the *forearm* joint. When you placed the IK handle you also created another node called the *end effector*. The end

effector defines the end of an IK solver chain. Because you want to control Melvin's arms from his wrist, you will need to translate the *end of the IK chain* from the forearm to the wrist.

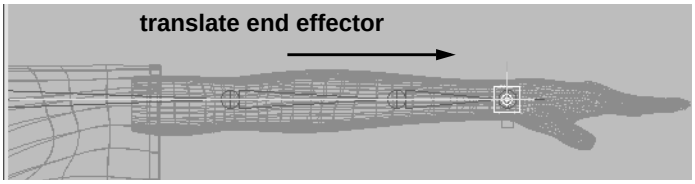
By default, the end effector is hidden and connected to a child joint of the last joint controlled in the IK chain as if it were a sibling of that child joint of the last joint in the IK chain. So when you move that child joint, the end effector will go along for the ride. IK is *not* invoked when an end effector is moved. This gives you the ability to reposition the IK chain / IK handle without invoking IK. As you will see this is what you want to happen for Melvin's forearm By changing the position of the effector you are changing the end position of the IK handle down to the wrist without running IK through to the wrist.

Tip: If you move the end effector, it is advisable to save a new preferred angle.

---

1 Move the end effector to the left_wrist joint

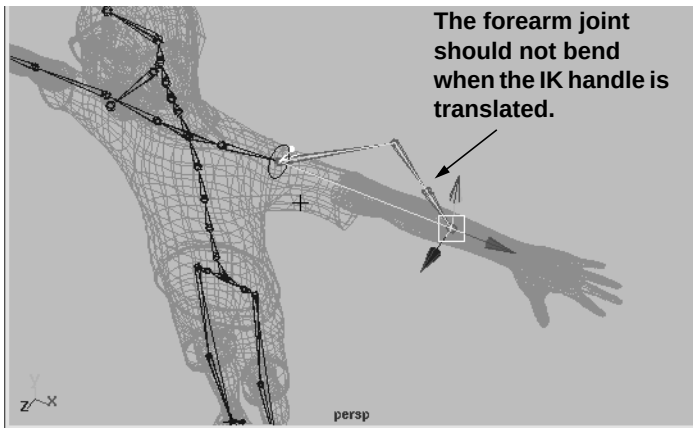
- In the Outliner, select the *L_armEffector*.
- Select the **Move** tool, then press the **Insert** key to work with the pivot point of the chosen node.
- Move the end effector pivot along the X axis to the center of the *left_wrist* joint.
- Press the **Insert** key to return to standard manipulator mode.



Translating the end effector

2 Move the IK handle along the x axis

- **Select** the arm's IK handle.
- **Move** the IK handle along the X axis to confirm that the forearm joint does not rotate.



3 Save your work

4 Repeat for the right arm

- Repeat steps 1 and 2 for the right arm.

Now you can translate the IK handle from the wrist without the arm bending at the forearm. This is a necessary technique that enables you to rotate the hand while creating realistic movement and deformation of the arm joints and skin. Look at the way your wrist rotates or twists from the elbow. You will eventually drive the rotation of the forearm joint based on the wrist rotation.

5 Save your work

WRIST AND ELBOW CONSTRAINTS

Constraints

Constraints are objects that you assign to control specific aspects of other objects' transformations. You will use a point constraint on the wrist to control the movement of the arms. A point constraint is used to make one object move to another object. The IK handle at the wrist will be constrained to a locator.

A pole vector constraint will control the rotation of the arms. The pole vector will always point to the pole vector constraint, providing a nice visual aid for positioning the elbows.

You will use locators as the constraint objects for the wrists and elbows.

Working with Locators

Locators are objects that you can see and pick but they will not render. In this lesson, you will create and use locators to control all the necessary keyframeable attributes for each hand. By using the locators, you only need to select one object to set keys on all the attributes belonging to the hand. Being able to set all the keys for a part of the character using one node speeds up the animation workflow. In later lessons, you will add attributes to the locators to control the hand.

Make a locator button

To add actions to the shelf, you must use a special key combination. By putting the *Create Locator* command on the shelf, you can easily access it as you work.

Press **Ctrl-Shift-Alt** then choose **Primitives** → **Create Locator**.

Now this action is available on the shelf.

Adding Pole Vector constraints to the elbows

As mentioned earlier, it is sometimes easier to control the IK Rotate Plane solver's pole vectors by constraining it to a locator. This will give you easy access to the control of Melvin's elbows.

The tricky part about using Pole Vector constraints on the elbows is deciding where to put them. Generally, a good place for the locators is directly behind the shoulder blades. Determining whether or not to parent these elbow locators depends on how you want to control the elbows. In this lesson, you will parent them under the collar joints so they will move with Melvin.

1 Create 2 locators and place them behind the shoulders

- Select **Create** → **Locator** and create 2 locators.
- Label the locators *L_elbowPoleLocator* and *R_elbowPoleLocator*.
- Position them behind the respective shoulder joints.

2 Freeze Transformations on the locators

- **Select** the locators.
- Select **Modify** → **Freeze Transformations.**

  Freeze Transformations sets the values on Translate, Rotate and Scale to 0 without changing the orientation. If you need to move Melvin back to this setup position later on, you can just pick the locators and move them to 0, 0,0 and they will return to this point.

3 Add Pole Vector constraints to the elbows

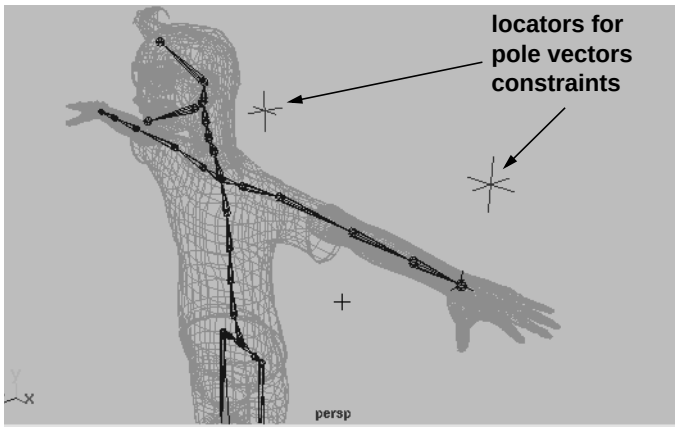
- In the work area, select the *L_elbowPoleLocator*, then **Shift-Select** *L_armIK* handle.
- Select **Constrain** → **Pole Vector**.

4 Parent the locator under the respective collar joint

- **Select** the locator and then the collar joint.
- Press the **p** key to parent the locator.

  Again, note that it is not necessary to parent the locators to the collar but it will make it easier since they will move with Melvin as he walks around.

5 Repeat for the right side



6 Save your work

Constraining the wrists to locators

Just as you point constrained the pole vectors to locators, you will do the same for the wrist IK handles. These locators will provide an easy selection method for grabbing the arm and placing it for keyframing. They will also provide a convenient place to put extra attributes for hand controls.

1 Create two locators

- Select **Create** → **Locator** and create 2 locators.
- Place them at the respective wrist joints. Use the **v** key to snap the locator to the IK handle.
- Rename the locators *L_wristLocator* and *R_wristLocator*.

2 Freeze Transformations on the locators

- **Select** the locators.
- Select **Modify** → **Freeze Transformations.**

3 Point constrain the locators to the wrist IK handles

- Select the *locator*, then **Shift-Select** the corresponding *IK handle*.
- Select **Constrain**→ **Point**.
- Repeat for the other side.

4 Save your work

Testing the character

Melvin is starting to take shape—at least the underlying skeleton is forming. You now have the most basic control points for blocking out motion for Melvin:

- Left and right ankle selection handles

- Left and right wrist locators
- Skeletal root selection handle

Work with the character and the handles you have created. Try placing the elbow pole vector constraints in different places to see how they aid in controlling the arm. Experiment with other set ups like the following:

Pole Vector placement and parenting

In this lesson, you parented the Pole vector locators to the collar bone. You may also want to explore leaving them unparented to see what happens. You may also want to place selection handles on the locators to make it easier to select them later. If all of your control nodes use selection handles then you can use the Selection handle pick mask to easily work with the controls.

No pole vector constraint

Some people prefer to use the **twist** attribute on the IK handle to control the rotation of the arm. An attribute could be added to the locator and then the two could be connected with the connection editor.

Summary

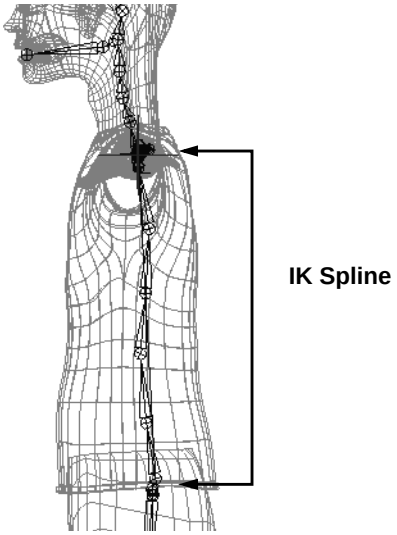
You have now completed the following tasks:

- Created Rotate Plane solver IK chains to control the more complex movement of the arms.
- Created a forearm joint to control the twist of the forearm and add realism to the deformations.
- Moved the end effector to change the effect of an IK chain
- Added pole vector constraints for more precise control of the Rotate Plane IK solver.
- Added point constraints to hold control attributes.

Lesson 4

4 IK Spline solver

In this lesson you will add an IK Spline solver to Melvin's back. This will control how his back sways and bends when he moves. This technique is optional since some animators prefer to control the back by directly rotating joints using Forward Kinematics. For this lesson, you will explore the use of the IK Spline solver so you can decide which way you prefer to animate the back.



Melvin's spine

Once the solver is in place, you will cluster points on the spline to help create controls.

In this lesson, you will learn the following:

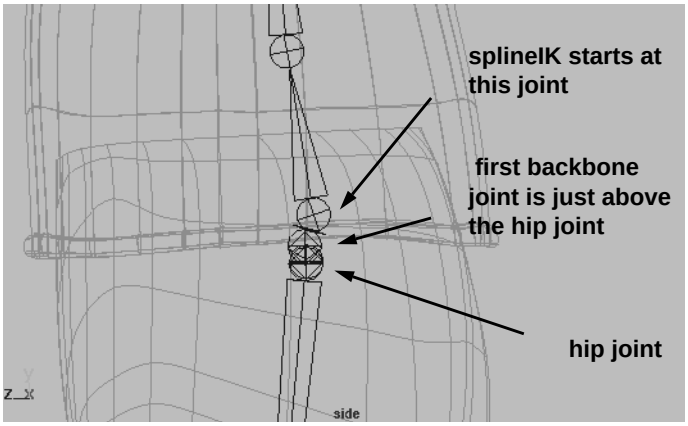
- How to set up a basic IK Spline solver
- How to verify the joint rotations on the joints of the back.
- How to use clusters for added control over the spline curve
- How to create selection handles for clusters
- How to parent the clusters

IK Spline in the back

When you use IK Spline, there are several things to keep in mind.

- Keep the curve as simple as possible for the IK spline. For the most part, the default of four CVs works fine. Note that the curve created when setting up the IK Spline solver will attempt to stay as simple as possible.
- Do not let the IK spline solver touch or cross any root joints. In the case of Melvin, you do not want the IK Spline solver to start or end at the *back_root* joint. You may recall that in Lesson 1 that you added a joint just above the *back_root* joint. You want the IK spline movement to be relative to that of the backbone joint and not the root joint.

  You do not want the solver to cross any root joints because this would cause the rotation of multiple skeleton chains. In the case of the back it would not only rotate the back but both hips as well. While the hips and the back do rotate together in real life, this motion can be difficult to animate on a digital character. For this reason you are separating the control of the pelvis from the control of the back using a small joint.



Joints at base of spine

- Create clusters for the CVs to make selecting and animating easier.

  Clusters have translate, rotate, and scale attributes while CVs only have position attributes. This means that CVs can't be keyframed as accurately as clusters can.

  Unlike CVs, you can add selection handles to clusters. This will speed up your animation workflow.
- Parent the clusters to the skeleton if you want them to move with the character.

Joint orientations

Before running the IK through the back joints, you will want to verify that the local rotation axes are pointing in a consistent direction. When creating

joint systems using **Auto Orientation** set to **XYZ** you may see that some of the Y and Z axes are pointing in different directions.

The Auto Orientation method will always force the X-axis to point at the child joint, but it has to make a decision about how to align the YZ axis depending on the direction of the child joint. In some cases, to ensure that the X-axis points correctly down the joint, the YZ axis will be flipped 180 degrees. This behavior is most noticeable when joints are drawn in the shape of an S.

In most cases this will not change how the system performs with IK or basic keyframing. But if you decide to at some point use an *expression* or *Set Driven Key*, you may have problems rotating the joints predictably. The remedy for this—and just generally good skeleton building practice—is to align the **Local Rotation Axes** so that they all point in consistent directions.

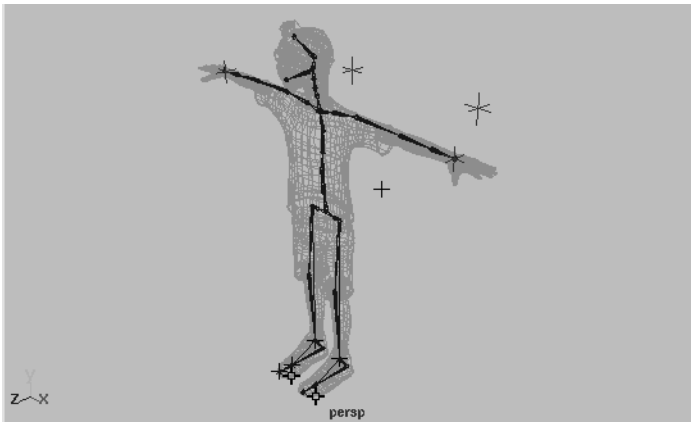
You will go into more detail concerning the importance of joint orientations in Lesson 6.

Aligning local rotation axes

Since the X-axis is already pointing down the joints, you might have to rotate the local rotation axis 180 degrees so that the Z-axis points towards Melvin's right hand. This will require that you rotate some of the local rotation axes 180 degrees around the X-axis.

1 Open an existing file

- Open the scene file *Melvin_04_arms.mb*.
- Template the skin to aid in the selection.



Melvin with joints in place

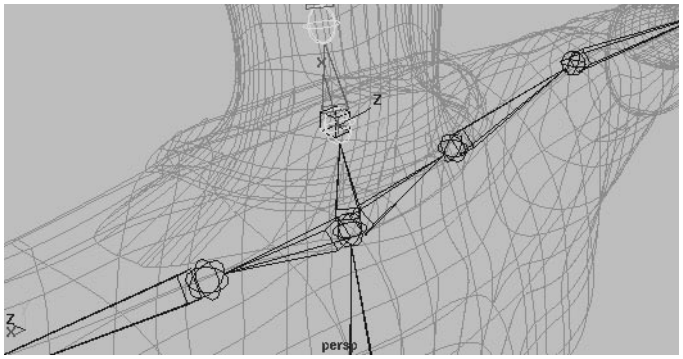
2 Verify the local rotation axes in Melvin's back

Confirm that all the local rotation axes in Melvin's back are pointing in the same direction. X should be pointing up the back, Y should be pointing out the back of Melvin, and Z should be pointing towards his right arm.

- Select back_a, the joint above the root joint.
- Press **F8** to change to Component mode.

- In the Selection mask tool bar, **RMB-click** the **?** button and select **Local Rotation Axes**.

  After you selected **Local Rotation Axes** from your selection mask, every joint in the selected's hierarchy will display its own local rotation axis.



Selecting the Local rotation axis

3 If an axis is flipped, rotate it 180 degrees in x

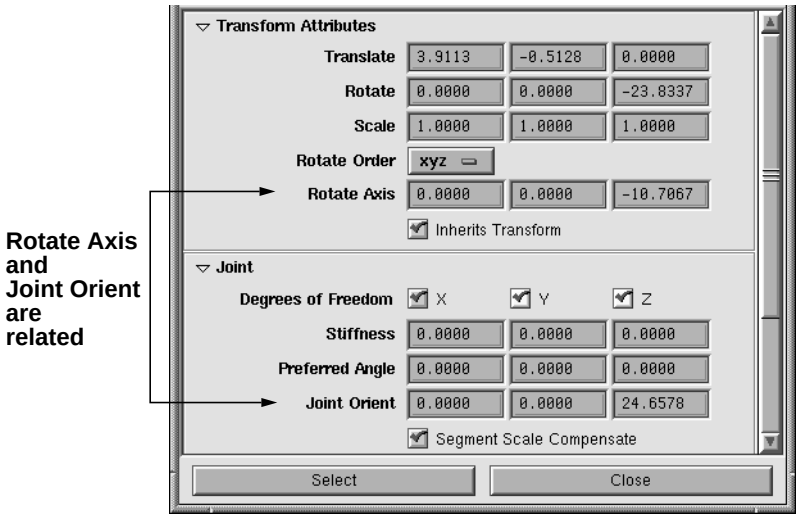
- Select the local rotation axis.
- Using the **Rotate** tool, rotate the axis 180 degrees in X.

  You can use the *coordinate entry field* to help you enter this value accurately. To change the rotate axis by command, select the joint, then enter:

  ```
  rotate -relative -objectSpace 180 0 0
  ```

  This command will rotate the axis exactly 180 degrees in X.

Note: If you change the rotation of the axis in the Attribute Editor you will have to change the **Rotate Axis** value and then change the corresponding **Joint Orient** value to be the inverse of the Rotate Axis value. If you change the **Rotate Axis** in the interface using the rotate manip, the **Joint Orient** will be changed automatically.



Attribute editor

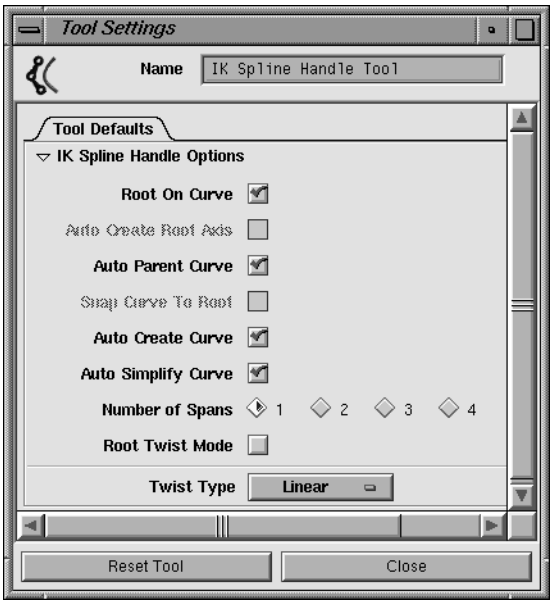
4 Save your work

Adding the IK Spline solver

The IK Spline solver allows you to control a chain of joints like the back bone with only a few control points. To animate a flexible back with forward kinematics requires you to keyframe the rotation of each joint individually. With IK Spline you will control all of the back joints with 3 control points.

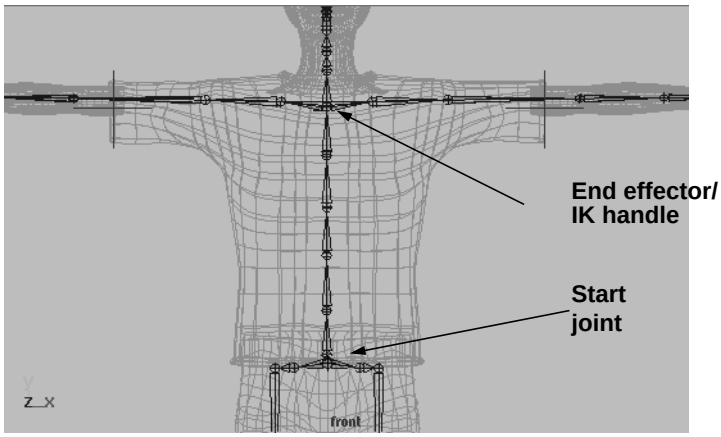
1 Add an IK Spline handle to the backbone

- Select **Skeleton** → **IK Spline Handle Tool** - ❐.
- Click **Reset** to set the tool to its default settings.



IK Spline option window

- **Select** the joint above the *back_root* joint, *back_a*, to place the start joint of the chain.
- **Select** the *back_shoulder* joint to place the IK handle.



IK Spline joints

An IK system is created with a curve running through the selected joints. You can then control this joint system by selecting the control vertices of this curve and translating them.

2 Name the new nodes

- Rename the new IK chain *Back_splineIK* and label the curve *back_curve*.

3 Save your work

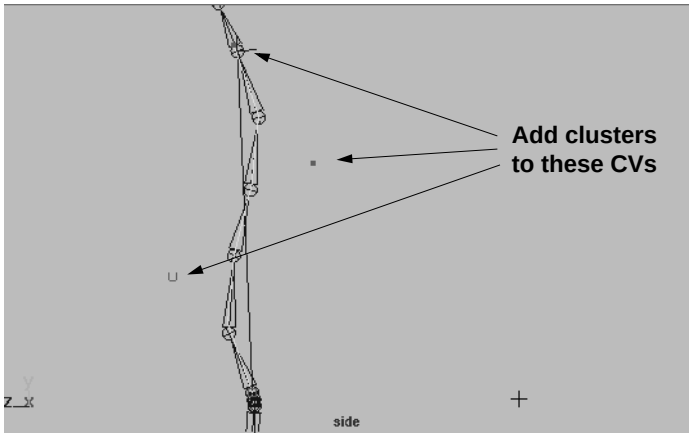
Test the IK Spline

There are two ways to operate the IK Spline. You should try both methods.

- Select the IK handle and change the value of the **twist** attribute. You may want to experiment with the **twist type** as well. It can be accessed through the Attribute Editor.
- Select a CV and move it.

Clusters for the IK Spline CVs

The curve used by the IK Spline solver has several CVs depending on the complexity of Melvin's back joints. Currently, the only way to select these CVs is in Component mode. To make selection easier, you will add clusters for the top three CVs of the back spline. Because you want these clusters to move relative to the root and *back_curve*, you will need to parent these clusters so that they will obey their parents' transformation predictably. For this reason, you will create them in **relative** mode.



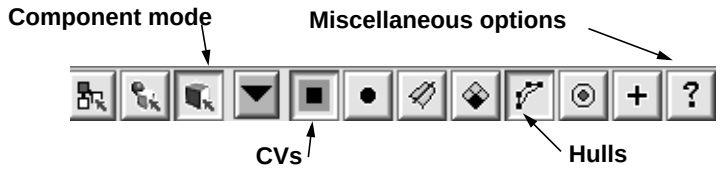
CVs to be clustered

Tip: Use the Pick Masks to select the curve in the work area. In Object mode, Select **Everything Off**, then toggle on the **Curve** icon.

---

1 In Component mode, select the top CV

- Select the *back_curve* that was created with the IK Spline handle.
- Change to **Component** mode.
- Set the **Pick Masks** to **Display Hulls** to help in selection.



Selection masks buttons

- **Select** the top CV.

2 Create a cluster in relative mode for this CV

- With the CV selected, select **Deform** →**Create Cluster** - ❒, and set the following:

  Mode to **Relative**.

- Press **Create**
- Rename this cluster *back_clusterA*.

3 Cluster the next two CVs

- Repeat steps 1 to 3 for the two CVs below the top CV. Label these *back_clusterB,* and *back_clusterC.*

4 Test the skeleton

- **Move** the clusters for proper movement in the back.

  When you translate the root, wrist, and ankle locators, the clusters remain in world space. You need to parent the clusters to the curve so they move along with Melvin.

5 Parent the three back clusters to the back_curve

- Select *back_clusterA*, then **Shift-Select** *back_clusterB* and *back_cluster_C*, then *back_curve.*
- Press **p** to parent.

  Now when you translate the root, wrist, and ankle locators the clusters should move with Melvin.

Note: When you parented these clusters, a node was inserted between the cluster and the *back_curve.* This is necessary because the clusters in relative mode will only be transformed by the node directly above them, so as to avoid any unpredictable behavior like double transformations.

Creating selection handles for clusters

To make selection of the clusters even easier, you'll display selection handles for them.

1 Add a selection handle to the clusters

- **Select** the *back_clusterA* node.
- Select **Display** → **Object Components** → **Selection Handles**.
- Press F8 to go into Component mode.

- Turn off the Points selection mask and turn on the handles selection mask.
- Click-drag a selection box around the node's new selection handle to select it.
- Move the handle so that it lies in back of Melvin's body. This will make it easier to select later.
- Repeat for the other two clusters.

2 Save your work

Driving the splineIK.twist attribute

When *back_splineIK* is selected, notice that there are several attributes such as **Offset**, **Roll**, and **Twist**. If you change the value for **Twist**, notice that the shoulder and head rotate. This rotation is tapered down to the beginning of the IK chain at the *back_root* joint.

To reduce the number of keyframed nodes, you can create an attribute on *back_root* to drive this **twist** attribute. Now, instead of selecting the IK handle to adjust the twist, you will be selecting one node – *back_root* – instead. Determining on which object you'll add attributes onto is really the users preference - the *back_root* is a common place to put these types of attributes, since it is the node that will be selected most often while animating.

1 Add a twist attribute to the back_root node

- Select *back_root*.
- Select **Modify** → **Add Attribute**...
- In the Add attribute window set the following:

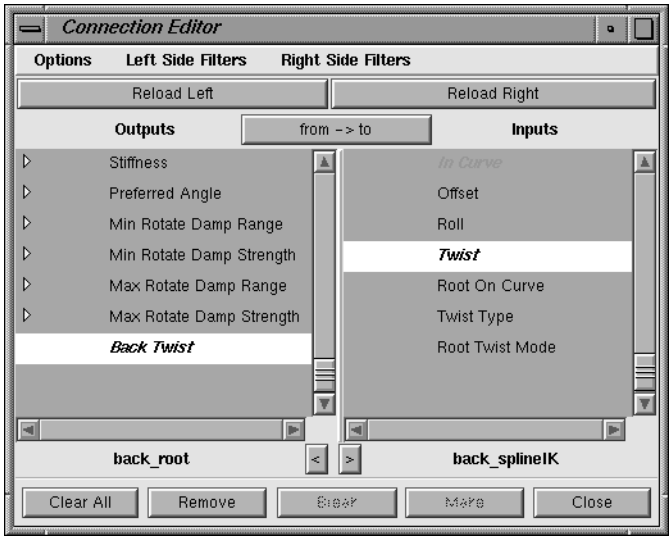
  Attribute name to **backTwist**;

  Data Type to **Float**.
- Press **OK.**

2 Connect the new twist attribute to IK spline's twist attribute

- Select **Windows** → **General Editors** → **Connection Editor...**
- Select *back_root*.
- In the Connection Editor, click **Reload Left**.
- Select *Back_splineIK*.
- In the Connection Editor, click **Reload Right**.
- In the Connection Editor, select **backTwist** as the Output and **Twist** as the Input.

  These two nodes are now connected. The *back_root* node's *backTwist* attribute will now drive the IK spline's *Twist*.



Connection editor

3 Save your work

Summary

You have now created some rudimentary control for Melvin's back making it easier to pose his back in a bent or swayed position. You've learned:

- How to add IK Spline
- How to create clusters
- How to use the connection editor

There are other techniques that animators have used for controlling the back:

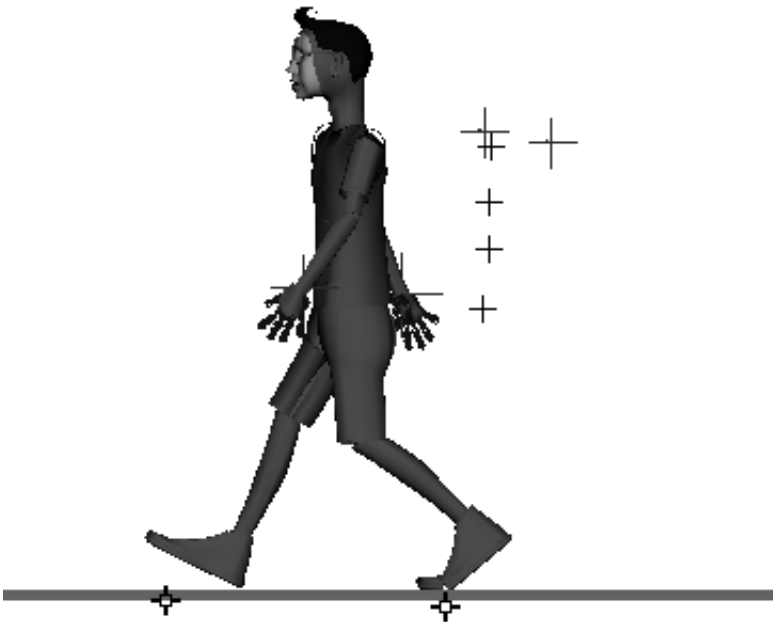
- **Set Driven Key** - create some attributes on the *back_root* joint possibly called (*backTwist, backSide, backForward*) and use Set Driven Key to drive the rotations of the back joints.
- **Expressions** - this technique would be similar to that of the Set Driven Key except it uses an expression to drive the rotation of the back joints instead of a Set Driven Key curve.
- **Forward Kinematics** - this is the hands-on approach. Consider using less joints in the back, and directly rotate the joints with the **Rotate** tool.

Note that this is not the only way to use Spline IK. Spline IK can also be used as a motion path to make characters bend smoothly along a path.

Lesson 5

5 Animating a walk cycle

Now that you have Melvin's controls set up it is time to start animating him. You will start with a walk cycle where you animate Melvin walking forward. This involves planting his feet and animating the pelvis area so that it follows accordingly.



Melvin walking

To help animate the pelvis, you will create expressions that keep it positioned in relation to the two foot controls. This means that you only have to key the two feet in order to start animating the walk.

In this lesson, you will learn the following:

- How to set up low resolution geometry
- How to organize your keyable attributes into character sets.
- How to create expressions for the pelvis joint
- How to animate the walk cycle
- How to set breakdown keys
- How to edit animations in the Graph Editor with buffer curves

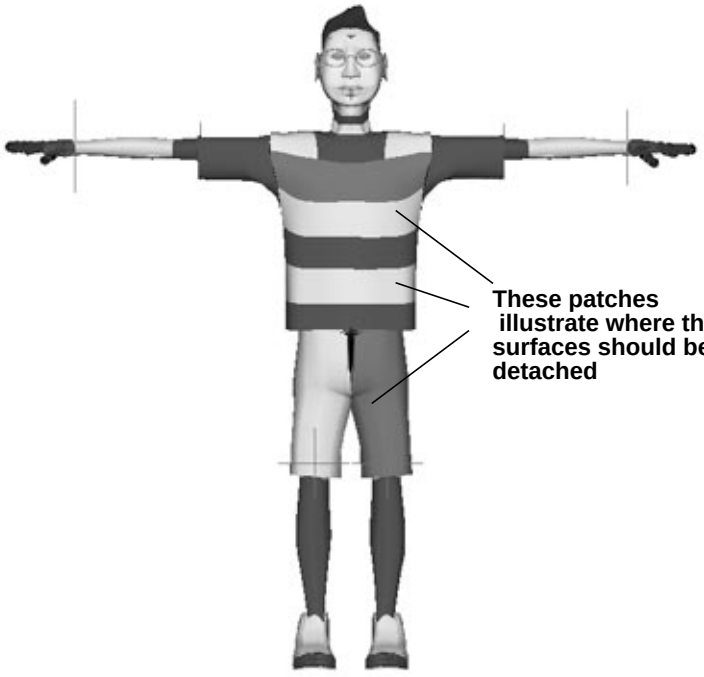
ANIMATING MELVIN

In this lesson, you will animate Melvin walking. This is where you begin to test out the controls you have worked on over the last few lessons. To assist you, low resolution geometry has been grouped into the skeleton and saved as a pre-made file. Below is a description of this process place so that you can apply it to your own work.

Working with low resolution geometry

You want to be able to see how the geometry will look as you animate the skeleton. Since a fully bound character would be a little heavy for previewing the motion, you can break down the geometry and parent low resolution pieces into the skeleton. This will speed up the workflow for setting keys while still allowing you to get a general feel for how Melvin will look down the line.

The goal of setting up detached surfaces is to speed up interactive performance by using grouped geometry instead of bound geometry. This gives the animator a good general idea of the position and orientation of the surfaces.



Detached surfaces

1 Open an existing file

- Open the file called *Melvin_05_spline.mb*.

Note: If you would like to just review this section and move on to making Melvin walk, a file has been prepared that already has the low resolution geometry.

---

2 Duplicate the geometry and add to a new layer

Place the existing geometry on a layer then duplicate it onto a new layer that will be reserved for low res geometry. You can then switch between the two layers.

- **Select** all of Melvin's surfaces.
- Create 2 new layers by selecting **Edit** → **Create Display Layer**
- **RMB** click on the *layer1* box and select **Assign Selected.**
- **RMB** click on **Layer Attributes** and rename the layer *melvin_hi_res.*
- Select **Edit** → **Duplicate**.
- **RMB** click on the *layer2* box and select **Assign Selected.**
- Change the name of *layer2* to *melvin_lo_res.*
- **RMB** on the *melvin_hi_res* layer and uncheck the Visibility box.

3 Detach the surfaces for the low-res layer

- **Select** one of the duplicated surfaces.
- Click on the surface with the **RMB** and select **Isoparm** from the marking menu.

  This puts you into component mode and lets you select isoparms on the surface.
- Click on one of the visible isoparms to select an isoparm.
- Drag with your **LMB** to select an isoparm.
- Press the **Shift** key and add more isoparms where they will be detached.
- Select **Edit Surfaces** → **Detach Surfaces.**

  This will divide the surface using the selected isoparms as cut lines. You will get one cut line for every isoparm you have selected.
- Continue with this process to break down the surface as outlined in the *detached surfaces* diagram shown at the beginning of this lesson.

4 Rebuild the detached surfaces

To further speed up interactivity as you work with these surfaces, you should rebuild them so that they are as simple as possible without losing too much detail.

- **Select** one of the detached surfaces.
- Select **Edit Surfaces** → **Rebuild Surfaces** - ❐. In the option window, set the following:

  Rebuild Type to **Uniform**

  Set Number of Spans according to the amount of detail required then click on the Rebuild button.
- Rebuild the remaining surfaces to simplify the low resolution geometry.

Tip: Rename the simplified surfaces with a short prefix. It can make it easier to pick them later. You can also use the *melvin_low_res* layer to pick this geometry.

5 Parent the reduced skin to the neighboring joints

To make the reduced surfaces move with the geometry, each piece needs to be parented to its neighboring joints.

- **Select** one or more of the reduced skin pieces.
- **Shift Select** the joint that you want to use to control the chosen surfaces.
- Press **p** to **parent** the surface to the skeleton.
- Repeat for all the other surfaces.

6 Save your work

EXPRESSIONS

Expressions are MEL scripts and mathematical functions that can automate the animation procedure. An expression can be used to connect attributes from one node to another node.

Keeping the Pelvis between the feet

Generally, when a character moves the pelvis stays between the feet. You will create an expression to do this.

1 Open an existing file

- Open the file called *Melvin_05_lowRes.mb*.

2 Create a new node for the expression

- **Group** the *back_root* node.
- Rename the new node *pelvisShift*.

  You are going to place the expressions on this new node. Later in this lesson, you will use the *back_root* joint to let you offset the position of the pelvis. This gives you flexibility to animate Melvin in positions other than walking.

3 Create the Expression

- **Select** the *pelvisShift* node.
- Select **Window** → **Expression Editor...**
- In the name field, enter *centerPelvis*.
- Enter these two lines:

```
tx = (L_foot.tx + R_foot.tx)/2;
tz = (L_foot.tz + R_foot.tz)/2;
```

- Click on the **Create** button.

The create button will become an edit button and you should see the two lines update to something like this:

```
centerPelvis.translateX = (L_foot.translateX +
R_foot.translateX)/2;
centerPelvis.translateZ = (L_foot.translateZ +
R_foot.translateZ)/2;
```

Note: You are not writing an expression for the Y translation of the pelvis since it is best to work with this attribute using keyframing.

---

4 Test the results

- Select the *L_foot* control and **Move** it.

  The pelvis should be moving to stay between the two feet.

5 Save your work

The expression works but what does it mean?

Now that you have seen the expression working, it is a good time to review how it is working. The mathematical function has been written to find the average value between the right and left feet.

If you look at the first line:

```
tx = (L_foot.tx + R_foot.tx)/2;
```

or

```
centerPelvis.translateX = (L_foot.translateX +
R_foot.translateX)/2;
```

In this equation, you are making the X translate attribute equal to the sum of the X translate position of the two foot controls:

```
(L_foot.translateX + R_foot.translateX)
```

And dividing the sum by two to get the average:

```
/2
```

The TranslateZ expression works the same way.

Note: The pelvis will get shifted forward slightly due to the expression. This can be compensated for later when you begin working with the *back_root* node.

The rotation of the pelvis

You will now create a second expression to control the twisting of the pelvis based on the rotation of the feet.

1 Create another new node.

- **Group** the *back_root* node.
- Rename it *pelvisTurn*.

2 Add the expression

- **Select** the *pelvisTurn* node.
- Select **Window** → **Expression Editor...**
- In the **Expression Name** field, enter *footHeight*.
- In the **Expression:** field, enter the following:

```
ry = (L_foot.ry + R_foot.ry)/2;
```

3 Save your work

Adding more control to the pelvis

Expressions can make animating easier but there is a risk of the animation becoming mechanical looking. There are two easy ways to deal with this:

- Use the expressions to get the general motion accomplished and then use the bake function to convert the expressions to animation curves to fine tune.
- Use group nodes to create a hierarchy so that animation can be layered.

The grouped nodes

When you created the expressions you put them on a new node above the back. A hierarchical animation is being setup. When working with hierarchical animation, you want to layer the nodes from the top downward to cover *Translation, Scaling* and then *Rotation*. The same thing is being done here with Melvin's pelvis.

- The top node controls translation.
- The second node controls the rotation.

This leaves the next node below, *back_root*, available for more animation.

If that node is translated, Melvin can shift his weight forward or backward or from side to side. That node can be used to make Melvin sit down or bounce as he walks.

If that node is rotated Melvin can lean back (feeling confident) or bend forward (carrying a heavy load).

Other nodes can be put in this hierarchy to do similar things. If Melvin had to walk up or down a hill, another node could be added with the pivot at his feet level.

MAKING MELVIN WALK

It is now time to begin animating Melvin. You are going to use the skeleton set-up you created throughout the day and animate a simple walk cycle. Before you start, you will use the Channel control window to simplify the keyframing process.

Setting up for keyframing

- Before you begin to set keys for the walk cycle, it is a good idea to use the Channel Controls to help restrict the animatable

channels on the various control nodes. The Channel Control window lets you make some channels non-keyable so that the **Set Key** function does not affect them.

1 Open the scene

- Open the scene file named *Melvin_05_expression.mb.*

2 Make some of the channels non-keyable for the left wrist locator

- Select **General Editors** → **Channel Control**.
- **Select** the *L_wristLocator*.
- In the Channel Control Editor, select **scaleX, scaleY, scaleZ,** and **visibility** from the **keyable** column.
- Click **Move >>** to transfer these attributes to the **non-keyable** column.

  You will now see that these attributes no longer appear in the Channel box. When you use the **Set Key** function, only the channels shown in the Channel box will be keyed.
- Repeat for the *R_wristLocator* node.

3 Restrict the keyable channels for the other control nodes

- Repeat this for some of the other nodes. Decide for yourself which channels are required to animate the following nodes:

  L_foot;

  R_foot;

  back_root

4 Save your work

Defining Characters

A **Character** in Maya is a collection of objects and attributes that are organized in a central place. This allows you to animate things as a single entity, rather than a group of separate objects. Characters don't have to be actual physical characters like Melvin, they only need to be objects and attributes that you want defined in one set. The benefit of working with characters is that you don't have to worry about keyframing each individual attribute in the character. Once the character node is selected, simply pose Melvin and put a keyframe on one of the attributes. By doing this, you will be setting a keyframe on all of the attributes in that character set.

You have created Melvin so he is easily controlled by only a few locators and selection handles. Now you are going to organize those locators and selection handles into two places so it is even easier to produce an animation.

1 Adjust the Pick Masks

The first step to creating a character is to select all of the objects that are going to be defined in that character. To do this, manipulate your pick masks and make the selections through the interface.

Menu set hotkeys

When working with menus, you will need to change menu sets. Remember the following hotkeys:

- **F2** - Animation
- **F3** - Modeling
- **F4** - Dynamics
- **F5** - Rendering

- Turn everything **off** in your pick masks.
- Turn on **Locators** and **Selection Handles**

  The main reason you are adjusting your pick masks is because you don't want unnecessary objects to be included in your character sets.

2 Make the Selections and Create the Bottom Character

To make animating easier, you are going to divide Melvin into two character sets; a top half and a bottom half. Dividing the character up like this will allow you to block in the animation with the bottom character (the root and the feet), and then refine the top half of the character (the arms, head, and back) after the timing has been worked out.

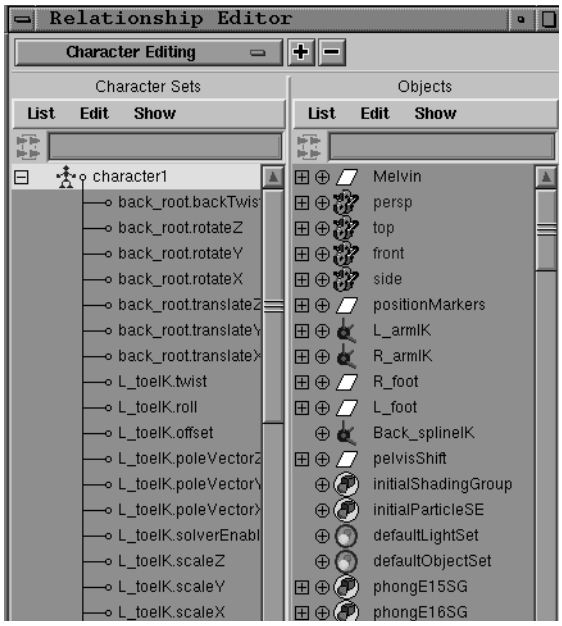
- Select the *L_foot* and the *R_foot* selection handles.
- Select the *back_root* joint.
- Select **Character** → **Create.**

3 Create a Character for the bottom half of Melvin

You will be using the Relationship Editor which holds all of the attributes that belong to a character.

- Select **Window** → **Relationship Editors** → **Character Sets...**

  Rename the character *melBottom*. You will now see *melBottom* to the right of the range slider. By clicking on the arrow next to the range slider, you can switch between different characters that you have created.



The Relationships Editor

4 Create another Character for the top half of Melvin

Now create another character involving the top half of Melvin.

- Select the wrist locators, the back clusters, and the pole vector locators.
- Create a Character out of the selections.
- Rename the Character *melTop*.

Note: You can also create sub characters by selecting the character node in the Outliner along with the other objects and selecting **Character** → **Create.** The sub characters will get keyframed when their parents get keyframed, but will have more control when only the sub character is the active character.

---

5 Edit the Characters

Although you have substantially reduced the amount of animated attributes in the character sets, there still seams to be a lot of attributes in the *melBottom* and *melTop* sets. You will now further reduce the amount of attributes in the set by transferring unneeded attributes.

- Select the *back_cluster's* scale and rotate attributes from the *melTop* character in the Relationship Editor.
- On the left side of the Relationship Editor, select **Edit** → **Remove Highlighted from Character.**
- Remove any other attributes that you don't want in the character set using the steps above. Pay particular attention to unneeded visibility and scale attributes.

Animating the walk

A walk cycle involves animating a character in several key positions. You want to start with both feet on the ground then animate one leg lifting as it shifts forward. The first part of this process is the animation of the feet sliding on the ground. The lifting of the feet will be added in another section.

1 Create a floor for Melvin to walk on

To help with the placement of Melvin's feet, it is a good idea to create a floor for Melvin to walk on.

- Select **Create** → **Polygon Primitives** → **Cube**.
- **Move** and **Scale** the cube to create a surface area in front of Melvin for walking.
- In the Y direction, **Scale** the cube down so that you can see a little thickness. This will help when working with Melvin in orthographic views.
- **Template** using **Display** → **Object Components** → **Templates**

Manipulator tip

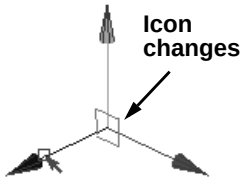
In the perspective view, you may want to be able to restrict interactive movement of Melvin's arms or legs along either the XY, YZ or XZ planes.

By default the move manipulator works either along a single axis using the axis handles or along the view plane using the center of the manipulator.

Manipulate along YZ

- Select the **Move** tool.
- Press the **Ctrl** key and click on the X axis handle. The center of the manipulator changes to show that it is aligned with the YZ plane.

- Click drag on the middle of the manipulator to work along the chosen plane.



To choose a plane to work with, press the **Ctrl** key and click on the axis handle that is perpendicular to the desired plane. To return to the default mode, press the **Ctrl** key and click on the center of the manipulator

2 Change keyframe interpolation type

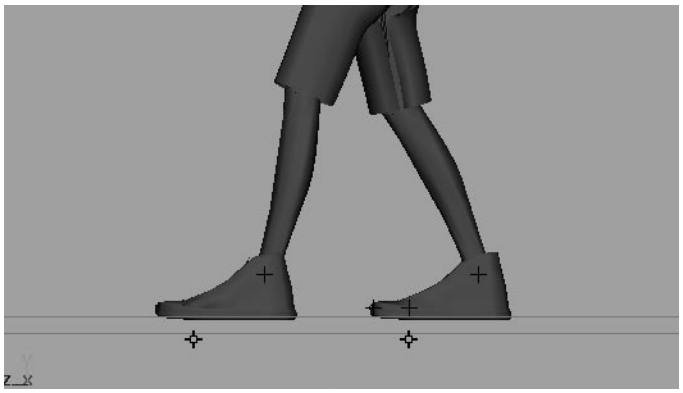
Before you begin, it is a good idea to change the set key interpolation type to **clamped**. This type of interpolation creates good ease-in and ease-out transitions between keys. This setting avoids the problem of overshoots associated with spline type interpolations.

- Select **Options** → **General Preferences**.
- Under the Animation tab, select **Clamped** as the **Default In Tangent** and the **Default Out Tangent.**

  This key type will help keep the feet from going through the floor when you plant the feet.

3 Position the foot controls

- Select the *back_root* node.
- **Move** this joint down a little until Melvin's knees are bending slightly.
- Select the *R_foot* node using its selection handle.
- **Move** this foot control slightly behind Melvin.
- Select the *L_foot* node using its selection handle.
- **Move** this foot control slightly in front of Melvin.



The start position of Melvin's feet

4 Select the Active Character

- Select *melBottom* as the active character.

  A box to the right of the time slider displays the name of the active character. Setting *melBottom* as the active character ensures that setting a keyframe on one attribute in the Character set will set a keyframe at that particular pose on every attribute in the Character set.

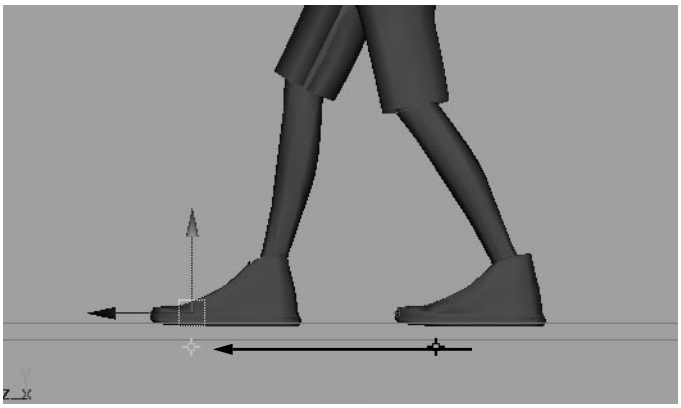
5 Set your animation preferences

- Click on the animation preferences button at the right end of the time slider.
- Enter a range of **120** frames in the preferences window.

6 Set keys for frame 1

- Go to **frame 1**
- Select one of the foot control nodes using its selection handle.
- Press **s** to **Set** keys for all of the attributes in the Character set.
- Set keys at **frame 20**
- Slide the time slider forward to frame **20**.
- **Select** the right foot and **Move** it in front of the left foot.

  Melvin's pelvis will also move forward because of the expressions.
- Select one of the foot control nodes then press **s.**



Moving the right foot in front of the left foot

7 Set keys at frame 40

- **Move** the time slider forward to frame **40**.
- **Select** the left foot and move it in front of the right foot.
- Press **s** to **Set** keys for the entire Character set.
- **Repeat** these steps until frame **120**

At intervals of 20 frames, set keys on the feet as you place one foot in front of the other in the following order.

- At frame **60** put the **right** foot in front of the **left** foot.
- At frame **80** put the **left** foot in front of the **right** foot.
- At frame **100** put the **right** foot in front of the **left** foot.
- At frame **120** put the **left** foot in front of the **right** foot.

8 Save your work

9 Playback the animation

Lifting the feet with Breakdown Keys

You will see Mevlin's feet sliding forward. The keys you have set establish the extreme positions of the feet and they also rough out the timing of

Auto Key

Auto Key is a useful way of quickly animating in Maya. To use Auto key, you must first set an initial key on the channels that you want animated. Auto key has the following conditions:

- Auto key will only work on channels that have already been keyed.
- If a Character is selected, a key will be set on each attribute in the Character set at each pose on the timeline.

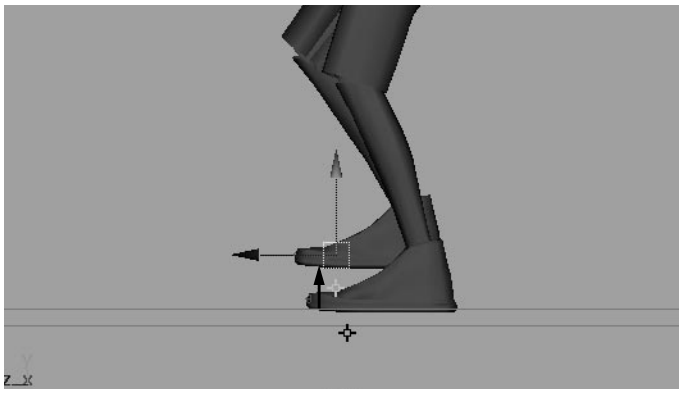
your animation. You now want to raise Melvin's feet at the mid-step to create more of a walking motion.

To do this, you will use Breakdown keys. Breakdown keys are a different type of keys from the keys that you have already set. They don't signify an absolute value in time, but rather a proportional relationship between adjacent keys. This means that breakdown keys always maintain their positional relationship between absolute keyframes. Each time a keyframe is moved, the breakdown keys surrounding it will also move to maintain its positional relationship.

For Melvin, you have already set keyframes on the feet to roughly define the timing of the animation. The rest of the keys that you set can be breakdown keys, so you'll always have the ability to change the timing by adjusting those original keys.

1 Animate the foot raising at mid step

- Go to frame **10**.
- Select the *R_foot* node and **Move** it up along the **Y** axis.



The passing position for the right leg

- Select **Animate** → **Set Breakdown**.

  This will set a breakdown key for every attribute in the *melBottom* character set.
- Go to frame **30**.
- Select the *L_foot* node and **Move** it up along the **Y** axis.
- Select **Animate** → **Set Breakdown**.
- Repeat for all of the mid-steps up until frame **110**.
- **Playback** the animation.

  Notice that the breakdown keys are displayed in the timeline as a different color tick compared to a traditional keyframe. They are also displayed in the graph editor and the dope sheet as a different color just so they are easily recognizable between keyframes.

2 Adjust the Timing

You now have keys (both breakdown keys and keyframes) that define the timing of Melvin's walk. Look at the animation and see if anything needs adjusting. This is a good opportunity to adjust the basic timing if you see something that you don't like.

- Play the Animation.
- Select a keyframe in the timeline by holding **Shift** then **LMB** dragging on the keyframe. The selected region of the timeline will turn red.
- **MMB** move the keyframe to adjust the timing. The breakdown keys will move proportionally with the moved keyframe.

3 Save your work

Adding the roll of the feet

Now that the positions of the feet have been keyed, you can key the **roll** of the feet to take advantage of the heel to toe motion that you established earlier. Again, you will use breakdown keys to put the roll on Melvin's foot.

1 Set the Character to None

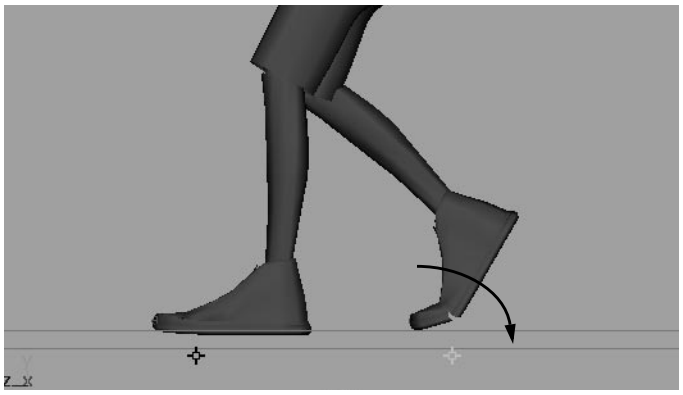
Since you will only be setting breakdown keys on specific attributes, you will not need to have the entire character node selected.

- Set the active character to **none** using the menu in the timeline

2 Use the roll attribute for the right foot

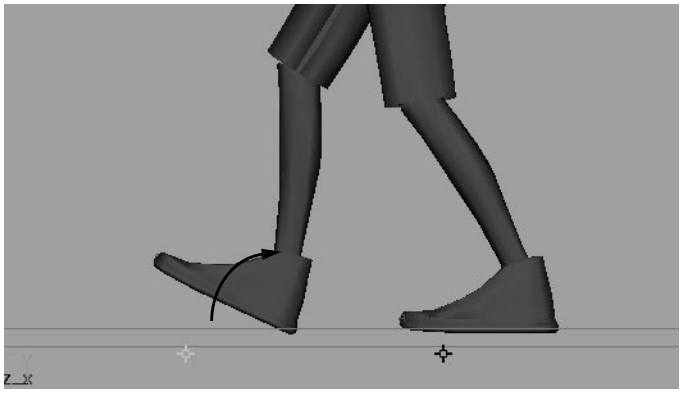
Earlier you set up the **roll** attribute on the two feet to create the heel to toe motion. A **roll** value of -5 rotates the heel up as if Melvin is placing his heel on the ground while a roll value of 10 rotates the toe forward at the point where Melvin is pushing off. You will now animate this attribute to establish the foot motion by using breakdown keys.

- Go to frame **1**. This is where the right foot is about to lift off of the ground.
- Select the *R_foot* node and set the **roll** attribute to **10**.
- Use the **RMB** in the Channel box to select **Breakdown Selected** on the **roll** channel.



The foot roll at frame 1

- Go to frame **20**. This is where the right foot is again touching down on the ground.
- Set the **roll** attribute to **-5** and **Breakdown Selected**.



The foot roll at frame 20

- Go to frame **40**.
- Set the **roll** attribute to **10** and select **Breakdown Selected**.
- Finish keying the **roll** attribute for the right foot then complete the same steps for the left foot. Always set the **roll** to -5 when the foot comes into the step and 10 when it is pushing off.

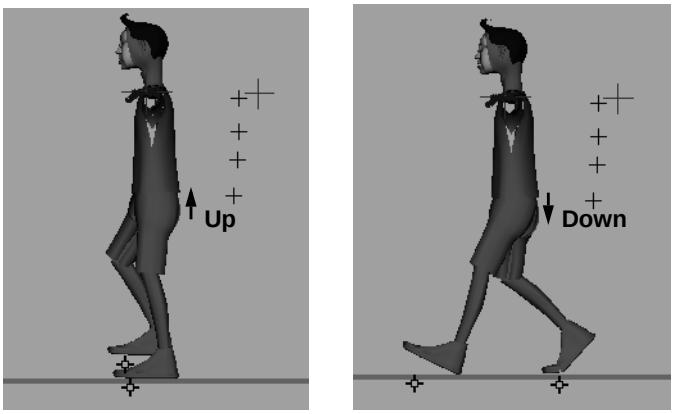
3 Save your work

Make the pelvis react to the feet

The animation of Melvin's walk is starting to look better but the body is reacting in a rigid manner. You can add animation to Melvin's pelvis so that it reacts in a natural manner to the walk.

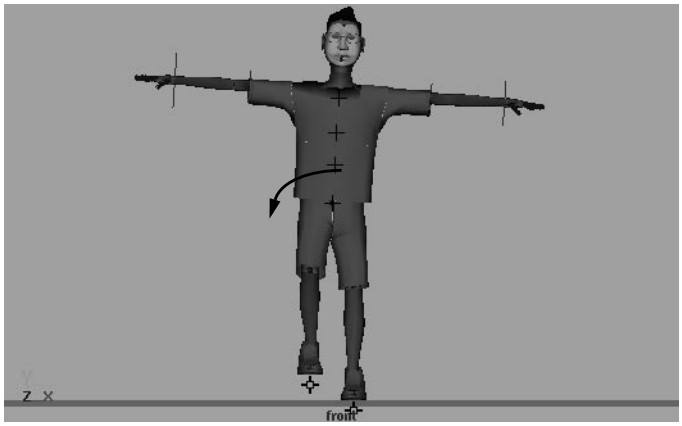
This section will not cover the steps, but will outline the things to look for so that you may use your talents to achieve the style you are looking for. It is suggested to use **Breakdown Keys** since these poses don't affect the overall timing of your animation. Shown below are some of the effects that you can add to the pelvis.

- Add some up and down movement to the pelvis. The pelvis will be lowered when both feet are planted and raised at the passing position.



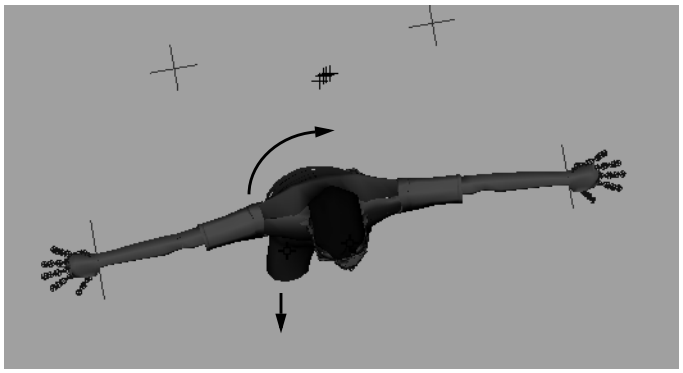
The raising and lowering of the pelvis

- Add rotation to the pelvis so that it rotates up and down with the legs. As the right leg raises, the pelvis rotates up on the left side; as the left leg raises, the pelvis rotates up on the right side.



Front view of pelvis rotations

- Add rotation to the pelvis so that it rotates back and forth with the legs. As the right leg goes forward, the pelvis rotates forward on the right side; as the left leg goes forward, the pelvis rotates forward on the left side.



Top view of pelvis rotations

Tip: Don't forget to edit the tangents on the keys for the pelvis to improve the quality of the motion. Setting the default tangent to clamped can give a jerky or mechanical type of motion to the pelvis.

Animate the arms

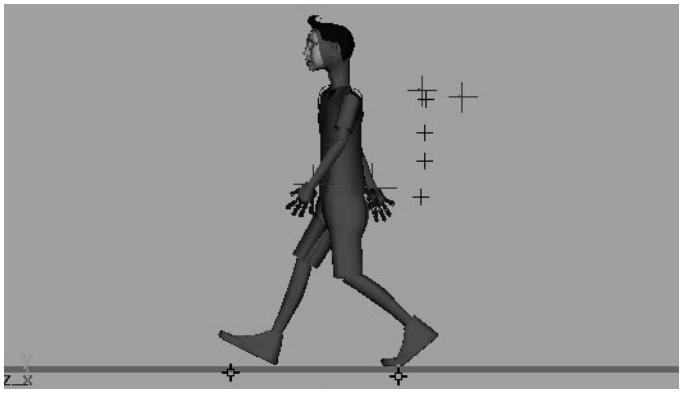
Like the pelvis, you can animate the arms to enhance the walk. As the right leg animates forward, the left arm moves forward to balance it. Similarly if the left leg is forward, the right arm is forward.

When animating Melvin's arms, make the *melTop* character the active character set to set keys on all of the arm and back attributes at any given pose.

To work on both characters at the same time, put the following command in the **Script Editor**.

```
setCurrentCharacters ({"melBottom", "melTop"});
```

It will be easier to make a shelf button out of this so you can always select both, *melTop* and *melBottom*, characters if you want to.



The arm motion

Tip: You may want to set your default tangent type back to spline in order to get a more fluid motion in the hands.

GRAPH EDITOR UPDATES

You have now roughly assembled the main pieces of Melvin's walk cycle. If you play back the walk, you will notice that something isn't quite right. The timing of the various actions needs refinement. You can refine the results using the Graph editor.

All of Melvin's motion is represented as animation curves in the Graph Editor. By default, the type of curves created between the keyframes is the same. Within this editor, you have the ability to edit the shape of this curve to add more character to Melvin's walk.

MODIFYING FOOT ROLL

To demonstrate how the shape of the curve can be used to refine the walk, you will start by focusing on the roll of the foot. This is a very complex motion that is not completely expressed in the keys you have set so far.

1 Change window layouts

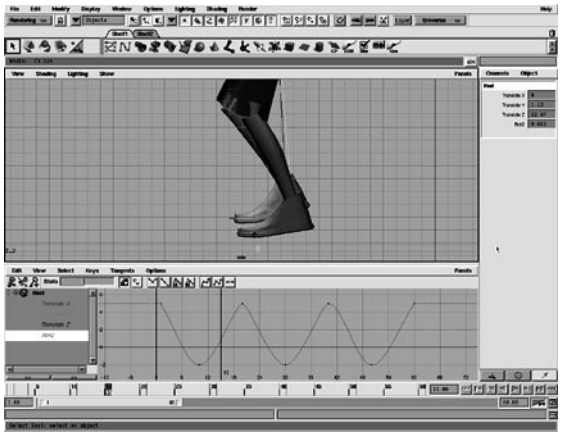
- Select **Window** → **View Arrangement**→ **2 Stacked.**
- Change the top panel to the side view.
- Change the bottom panel to the **Graph Editor**.

2 View the left foot's Roll curves

- In the Graph Editor select the *melCharacter* to see all of the attributes belonging to that set.
- Scroll down and find the *L_foot.roll* attribute.
- In the **View** menu toggle on **Auto Frame**.

Tip: You can also select the *L_foot* selection handle and bring up the Graph Editor. You will find the roll attribute listed under both the *L_foot* group and the *melCharacter* node. Editing one curve will update the other curve.

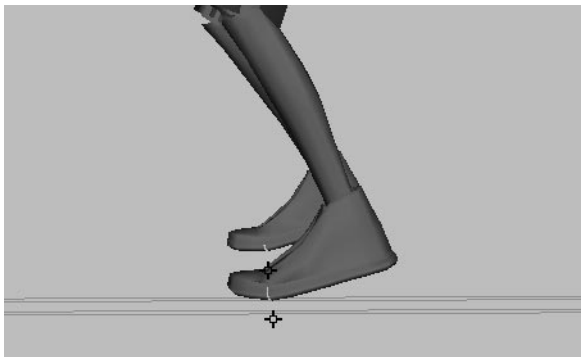
This automatically centers the selected curve in the window.



Stacked side window and Graph Editor

Adding keyframes in the Graph Editor

You may have noticed that when Melvin is in mid-step, the right leg is elevated but the left foot is not planted on the ground. Since you know that the foot is supposed to be flat on the ground at this point, you will add a keyframe to keep the roll of the feet flat at this important point.



Left foot not flat on ground

1 Move the animation to mid-step

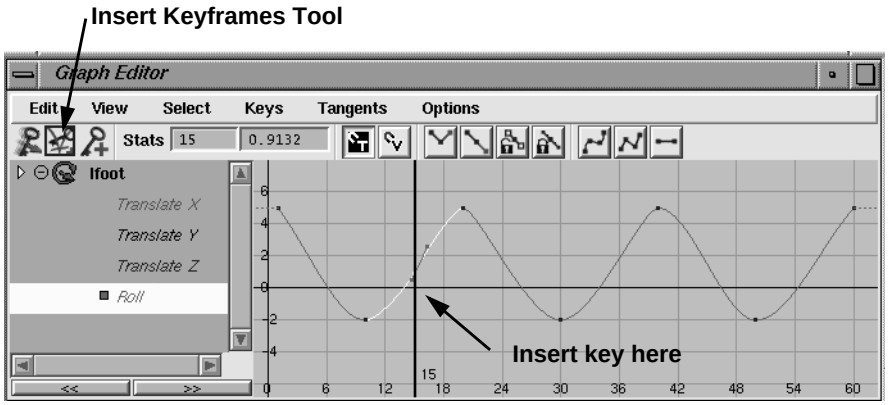
The problem is that the foot roll should have a value of **0**, halfway between the two main foot keyframes.

- Set the time slider to around frame **15**.

2 Insert a keyframe

- In the Graph Editor, select the **Insert Keyframes Tool.**
- **LMB**, select the animation curve.
- **MMB**, click on the curve at the current frame to insert a keyframe.

  A keyframe with tangent handles has now been added at this frame.



Inserting a keyframe

3 Change the value of the keyframe

- Click drag a selection marquee around the new keyframe with the **LMB**.
- In the Graph Editor, change the second **Stats** field to **0.**

  This sets the value of the roll channel to **0**. The frame stays at **15**. You will now notice that the foot roll is flat on the ground.

4 Switch to a breakdown key

Since Melvin's foot roll is not involved in the overall timing of the walk, you are going to convert that keyframe to a breakdown key. It is a good idea to only create timing poses as keyframes and while everything in-between can be breakdown keys.

- **Select** the new keyframe.
- In the Graph Editor, select **Keys** → **Convert to Breakdown.**

5 Repeat for mid-step frames on the left foot

Move through the rest of the animation and repeat the previous steps to correct the roll at mid-step for all the cycles of the walk.

6 Repeat all steps for the right foot

7 Playback the animation

When you play back the walk cycle, notice that Melvin's feet roll a little better now. It would be more realistic if Melvin, after stepping on his heel, would put his foot down more quickly and remain on his foot a little longer before pushing off again. You will achieve these refinements by editing the slope of the animation curve.

8 Save your work

Working with the animation curves

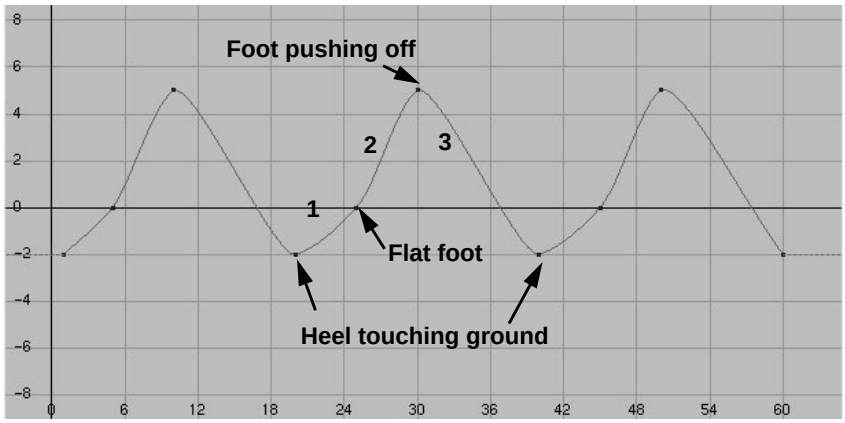
You have keyframes set when the foot is in three important positions:

- The foot's heel touching the ground.
- The foot flat on the ground

- The foot pushing off

You will now refine the timing of these events by editing the *slope* or *speed* of the curve. To speed up the motion, increase the slope of the curve. To slow down the motion, decrease the slope of the curve. Think of a skier going down a hill. Steep slopes increase speed while flatter sections slow the skier down.

The following two diagrams show you before and after curves of edits that are required:



Before modification

Slope 1 - Between the heel touching the ground and the flattening of the foot

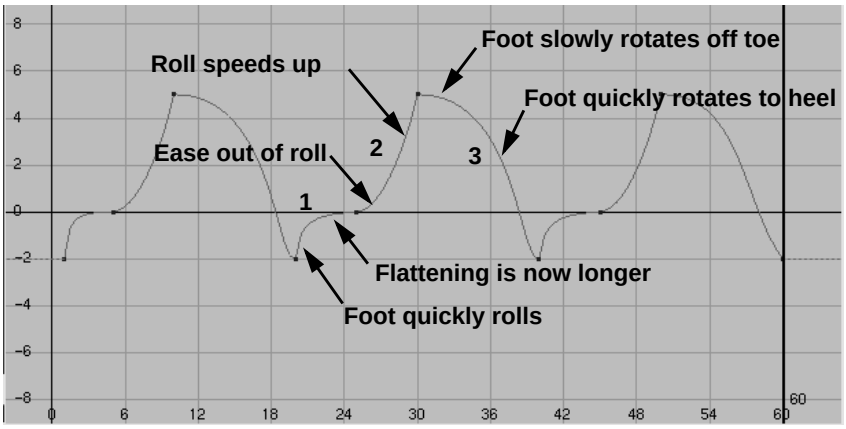
Currently, the slope of this curve is relatively constant. You want the heel to quickly lift off the ground then gently flatten. You will therefore first increase the slope of the curve to speed up the motion then flatten the curve to make the foot to remain on the ground a little longer.

Slope 2 – Between the flattening of the foot and the foot pushing off

In this section, you want the foot to slowly roll onto the toe then quickly move to the push-off position. Start with a flat slope that transitions into a steep slope.

Slope 3 – Between the heel touching the ground and the flattening of the foot

This final cycle illustrates how the foot will rotate from the push-off position and back to the heel-to-ground. You will change this to make Melvin point his toe downward until his toe quickly snaps upward before touching the ground again.



After modifications

In the next section, you will learn how to make these changes in the Graph editor.

Modifying Curves

You are now going to edit the shape of the curve as shown above. Your goal will be to achieve a more complex motion by reshaping how the slopes of the curve are drawn. You will also create buffer curves to compare the changes made between animations.

1 Create a buffer curve

When adjusting curves, it is sometimes difficult to tell how much improvement an animation is making. There are many times when it is just easier to compare two curves against each other to see which one looks better. In Maya, you can create buffer curves in the graph editor that will allow you to toggle between the active curve and the buffer curve so you can see the difference in the animation.

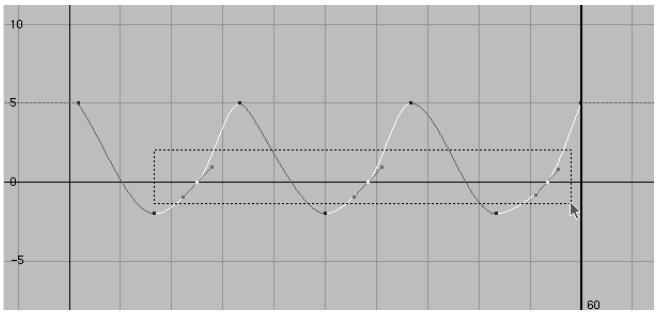
- Select the *l_foot.roll* curve in the graph editor.
- Select **Curves** → **Buffer Curve Snapshot** or press the **Buffer Curve Snapshot** button at the top of the Graph Editor.

  By creating a buffer curve, you are basically creating a reference curve that you will look at later after you have altered the existing curve. You will not notice any difference in the Graph Editor until you have displayed the buffer curve at the end of this lesson.

2 Flatten the tangency of the curve for slow movement

To create a quick heel-to-flat foot and hold motion, you will start the curve from a steep slope and ease into a flat slope.

- **Click-drag** to select the all the keyframes on the curve with a *Roll* attribute to **0** using the **LMB**.



Click-drag select keyframes

- In the Graph Editor menu, select **Tangents** → **Flat.**

  Your selected keyframes' tangent handles are now positioned horizontally, creating a very slow motion at each point.

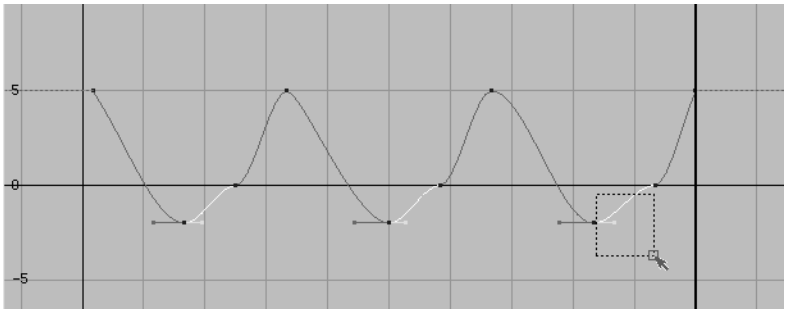
3 Break the tangency of the curve for abrupt movement

- **Click-drag** select all the bottom keyframes.
- In the Graph Editor menu, select **Keys** → **Break Tangents.**
- In the Graph Editor menu, select **Keys** → **Free Tangent Weight.**

Tip: Setting the tangents to a free weight will only work if your **General Preferences** are set correctly. You can verify that you can free the tangent weight by going into the Animation tab of the General Preferences and setting **Weighted Tangents** to **on**

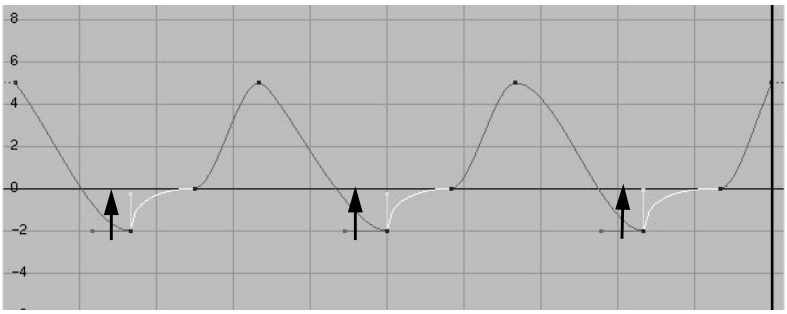
This allows you to modify each keyframe's tangent handle individually, as well as changing the scale of the tangent handle.

- **Shift-select** the right tangent handle of the selected keyframes.



Shift-select tangent handles

- Press **w** to get the **Move** tool.
- With the **MMB**, click-drag the handles over their respective keyframes.

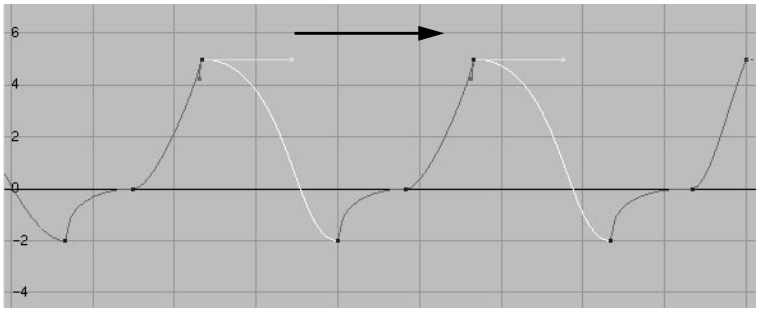


Moved tangent handles

4 Correct the speed of the toe-to-heel position

To make Melvin's foot gradually speed up from the push-off position to where the heel contacts the floor, the curve must start from a flat slope and change to a steep slope.

- Repeat the previous steps using the **Break Tangents** and the **Free Tangent Weight** tool to achieve the following curve shape.



Slow to fast curve

5 Switch buffer curves

After adjusting the curves with the above steps, its time to see what a difference you have made. Play the animation and concentrate on the foot roll.

- In the Graph Editor, select **View → Show Buffer Curve.**

  This should display the original curve that you created for the roll.

- Select **Curves → Swap Buffer Curve.**

  Now the curves for the *l_foot.roll* have been switched. Play the animation again and concentrate on the changes. Switch the curves back after your through.

6 Repeat the steps for the right foot

7 Save your work

Refining more animation curves

The techniques for refining the animation curves on Melvin's feet can be easily applied to any of his keyable attributes. Try experimenting with Linear curves as well.

Summary

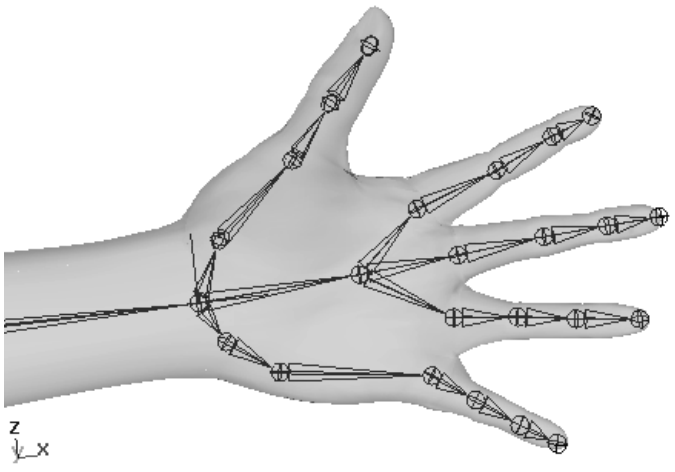
In this lesson, you have completed the following tasks:

- Created a low resolution version of your character for faster animation playback.
- Written expressions to make animating easier.
- Simplified the Channel Box by making unkeyed channels "non-keyable".
- Organized Melvin's attributes into one central Character set.
- Created keyframes and breakdown keys which hold a proportional relationship between keyframes.
- Refined animation in the Graph Editor.
- Tested animations by switching between buffer curves.

Lesson 6

6 The Hand Joints

In this lesson you will build the skeletal structure for Melvin's left hand. You will design the hand to have sophisticated articulation while still being simple to animate.



Melvin's Hand

As part of this lesson you will verify the local rotation axis of each joint since the hand will be animated using **Forward Kinematics.** The correct orientation of the axes will be crucial to your success in the next lesson when you use Set Driven Key to control the fingers.

In this lesson, you will learn the following:

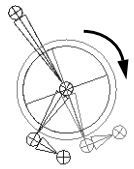
- How to add joints to the end of an existing joint hierarchy
- How to verify and correct a joint's local rotation axis
- How to build a hand skeleton
- How to place the hand joints for proper skinning and motion
- How to test the motion using Forward Kinematics

Editing joints: Rotating and Scaling

After you create joints, you can reshape the skeleton by using a combination of rotating and scaling:

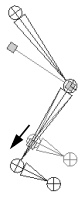
Rotate method

- **Select** the desired joint.
- Select the **Rotate** tool.
- Rotate the joint to effect the selected joint and all the lower joints.



Scale method

- **Select** the desired joint.
- Select the **Scale** tool.
- Scale the joint to stretch the joint's bone and reposition all the lower joints.



Using these two editing techniques, you can reposition your joints without affecting the joint orientations in the chain.

JOINT ORIENTATION

Each joint has a Local Rotation Axis that defines how the joint will react to transformations. For most parts of a character, the default orientation will be fine. But when setting up more complex situations, like Melvin's hands, it is important to make sure that all the joints' axes of rotation are aimed in a consistent manner.

Why should you use Auto joint orient?

By using **Auto joint orient** you make sure that the Local Rotation Axes of the joints are all aligned with the bones that follows. This will help control the joints transformations if you choose to use Forward Kinematics. It can also have an effect on how your clusters work later when you skin your character.

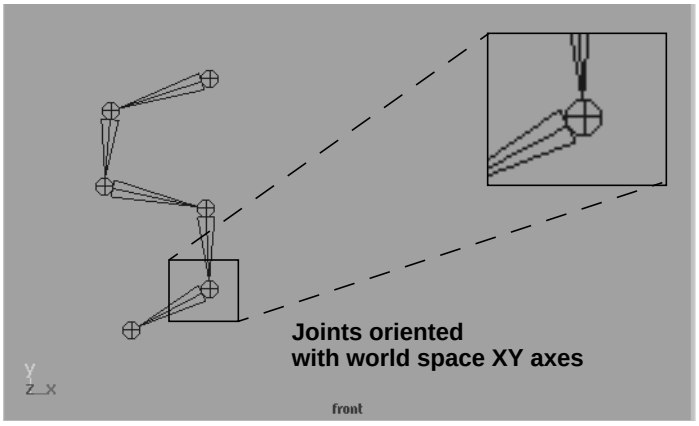
1 Draw joints with Auto joint orient turned off

- Select **Skeleton** → **Joint Tool** - ❐ and set the following:

  Auto Joint Orient to **none**.

- In the front window, draw several joints in an S pattern.

  You will see that the round joint icons are all aligned with the X and Y axes, not with the bone that follows the joint.



S pattern joints

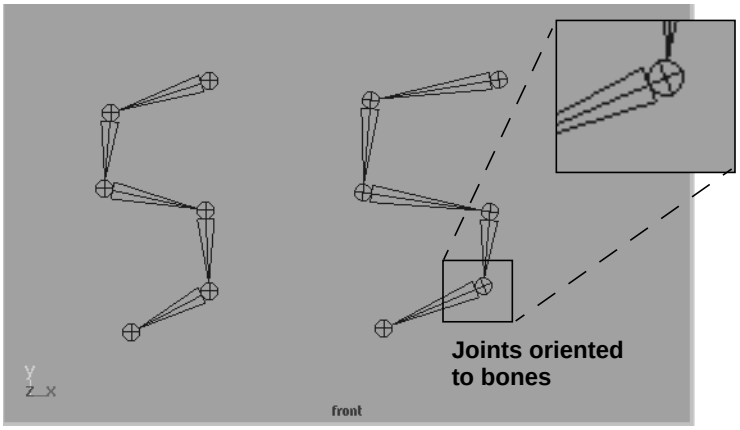
2 Draw joints with Auto joint orient turned on

- Select **Skeleton** → **Joint Tool** - ❐ and set the following:

  Auto Joint Orient to **XYZ**.

- In the front window, draw several joints in an S pattern.

  The round joint icons are all aligned with the bone that follows it. Because there is no bone to align to, only the last joint is aligned with the world space.

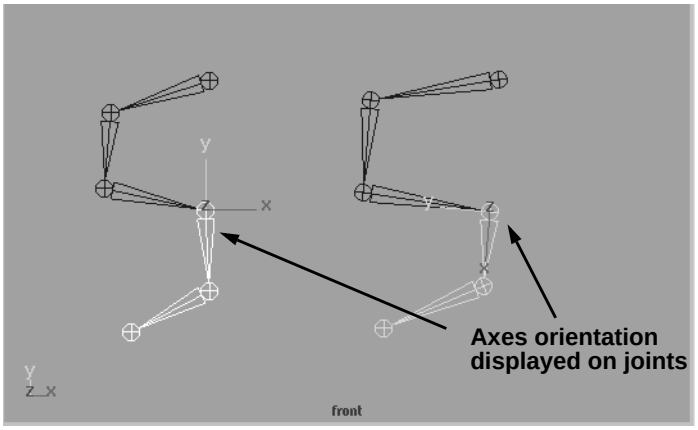


Auto oriented joints

3 Display the joint axes

- **Select** the fourth joint in both of the hierarchies.
- Hit F8 to go into Component mode.
- Check the **?** box in the Status Line. This should display the local rotation axes of the selected hierarchies.

  The first skeleton has its axes aligned with world space while the second skeleton has its axes pointing down the bone.



Local rotation axes displayed

4 Rotate joints around the X axis

- In the channel box, click on the *Rotate X* channel.
- With your **MMB**, click drag in the front view. This lets you edit the *Rotate X* channel value interactively.

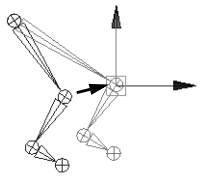
  The second skeleton rotates nicely around the bone while the first skeleton is rotating in world space with no relation to the bone at all.

Editing joints: Move and Pivot

You can also edit your joints either using the Move tool or by moving the pivot of the joint:

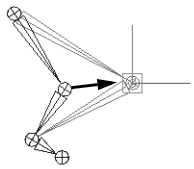
Move method

- **Select** the desired joint.
- Select the **Move** tool.
- Move the joint to position the selected joint and all the lower joints.



Pivot method

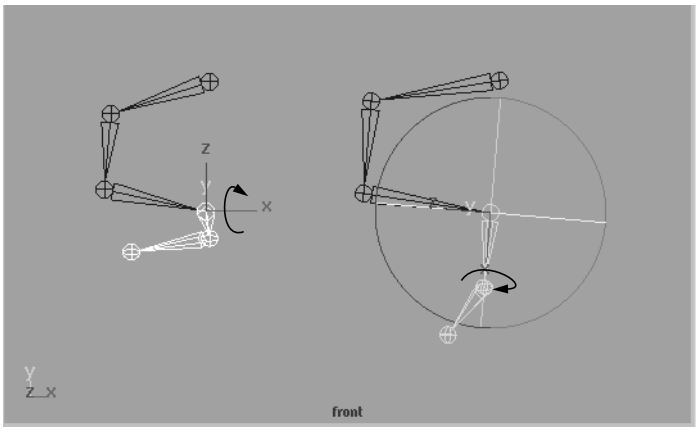
- **Select** the desired joint.
- Select a transform tool then press the **Insert** key.
- Move the pivot to position only the selected joint.



If you use this method then the orientation of the joint above the edited joint will be broken. You can correct this using the following script:

```
joint -e -oj xyz
```

This will re-orient the selected joints as if they were created with the **Auto Orient** option.



Joints rotated around their local X axes

- When you are finished, press the **z** key to undo the rotations.

How editing joints affects joint orientation

In the side bars on the two last pages, you can see that there are four ways of repositioning joints. The first two – scaling and rotating – will always maintain your chosen joint orientation while moving and pivot location will prevent your local rotation axes from being aligned properly with the bones.

You can use all four methods for editing joints but move and pivot will require that you re-orient your joints as outlined in the side bar.

How to correct flipped axes

In addition to making sure that one axis always points down the bone, you also want to make sure that the other axes relate to the skeleton in the same way.

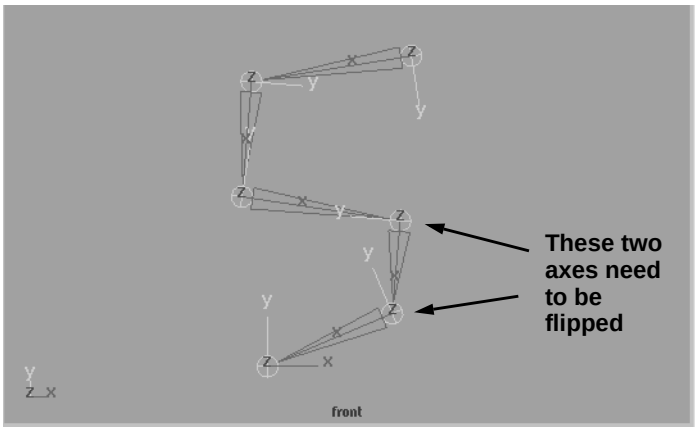
1 Display the joint axes as a selectable component

You will now learn how to correct the orientation of a joint axes by rotating it into the proper position. You must first display the local axes as components.

- **Select** the root joint of the second S shaped skeleton.
- Press **F8** to go into component mode then click on the **?** selection mask button.

  You can see the local axes for all the joints. The first two axes are pointing to the right side of the skeleton while the next two are pointing towards the left. You can now select and edit the two flipped axes.

Note: Don't worry about the last joint. Its local axis isn't oriented towards anything and doesn't affect how the skeleton works.

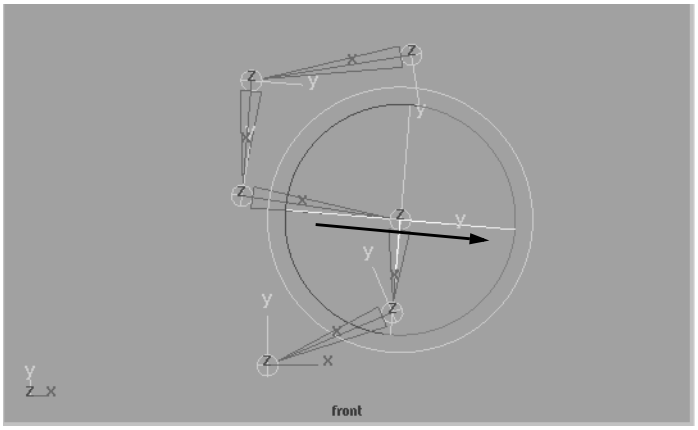


Joints rotated around X axis.

2 Rotate the joint axes interactively

One way of rotating the local axes is to select them and rotate them interactively.

- **Select** the local rotation axis on the fourth joint down.
- **Rotate** the axes to flip it around 180 degrees.



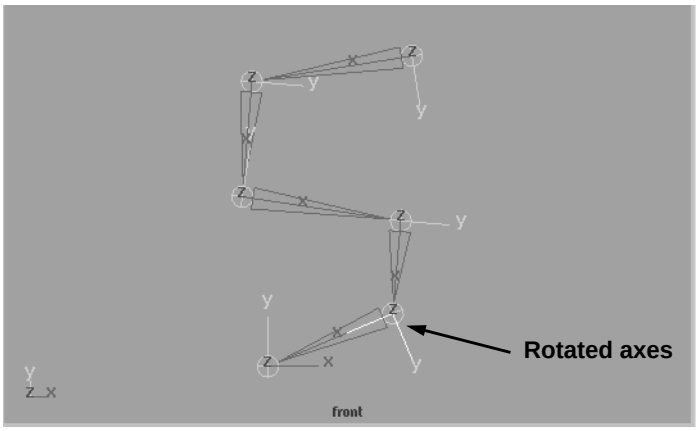
Rotated axis

3 Rotating the joint axes using a script

You can also rotate the axes more accurately by entering a simple line script in the command line. Because the joints were created with Auto Orient set to XYZ, the flipped axis only needs to be rotated 180 degrees.

- **Select** the local rotation axis on the fifth joint down.
- In the command line, enter the following command:

  rotate -r -os 180 0 0



Corrected axes

When do you worry about the local rotation axis?

To determine the proper axis for your joints, you need to understand what you are going to do with the joints. Read the following text to explore some of the possibilities. This topic will be discussed throughout the course in more detail. You may want to return to this list when you are more familiar with the options available:

- **Forward kinematics** - for forward kinematics, it is important that joints which are not aligned with the world axis be able to rotate correctly local to their direction. This is very apparent in the fingers and you will be looking at this in much more detail in this chapter. How a finger joint rotates is directly linked to the orientation of its local rotation axis. The finger joints should only rotate around one axis, so it is important that the axis be aligned directly down the bone.
- **Expressions** - if you write an expression to rotate all the joints of the back around the z-axis, you would want them to rotate in a consistent direction. If one of the axes was flipped, the task of writing an expression would be much more difficult.
- **Set Driven Key** - the reasons for having consistent orientation is similar to the reasons outlined for expressions.
- **Inverse Kinematics** - the differences are not as apparent when IK goes through the joints, but in many cases joints will only need to rotate around the axis, and having the local rotation axis set before adding IK will help with this - especially in the areas of hinge joints such as knees and elbows.
- **Skinning and flexors** - the orientation of the local rotation axis will affect the default flexor attributes as well as the skinning behavior in certain cases. This will be discussed later in this course.
- **Constraints** - with orient constraints, the result will depend on whether the two objects involved in the constraint have similar or the same orientation.

You may recall that you created the joints with **Auto Orient** set to **XYZ**. This forces the X-axis to point down the bone toward the child joint. If the joints are translated when adjusting their placement you will need to adjust their orientation so they still point down the bone. Note that if their positions are adjusted by rotating and/or scaling, the joint orientation will still point down the bone.

Changing Joint Orientation

When moving joints after they are created it is possible that the move will change the orientation of the local rotation axis. It is important to note that if a joint is either rotated or scaled, that the local rotation axis will be transformed with it and the X-axis will still point down the bone.

Notice though what happens if a joint is translated - either with the translate tool or the translate-insert tool. The joint orientation stays the same and no longer points down the bone. Joints can be re-oriented with the following command:

```
joint -e -oj xyz;
```

It is important to note what is happening when the joint orientation is changed with the "joint" command and/or by rotating the axis directly in the interface. Open the Attribute Editor and notice what changes and what doesn't change:

- **Rotate** doesn't change - you want this value to stay the same.
- **Rotate Axis** changes so the joint will rotate around the correct axis.
- **Joint Orient** - changes by the inverse of the value of the Rotate Axis so that the Rotate attributes don't have to change. This value compensates for the change in the Rotate Axis.

Notice what happens if the values are changed by typing them in the Attribute Editor. For example, type in a new value for the Z Rotate axis and note that the joint rotates because the Joint Orient has not been set to the inverse/negative of the Rotate Axis value.

BUILDING MELVIN'S HAND

There are a number of ways to create a skeletal system for a hand. Your goal is to set up a hand that is both simple to animate and articulate in how the joints can be positioned.

Following is a diagram of the hand that shows you the joints that you will be creating in this lesson.

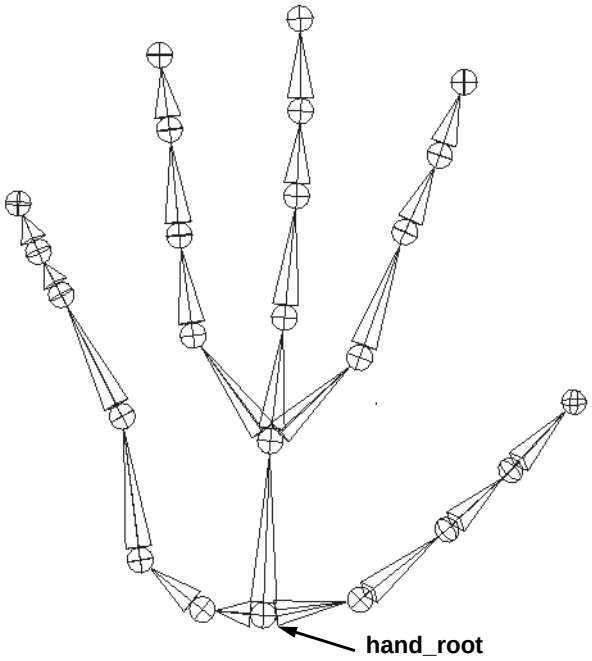
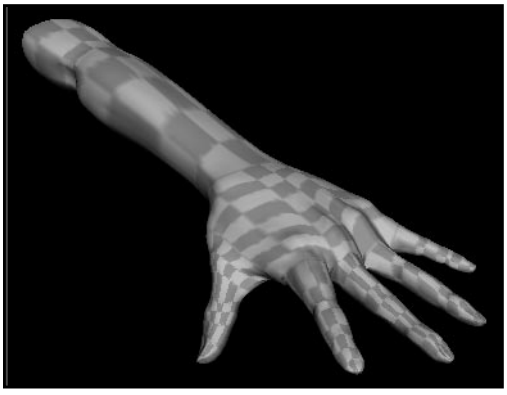


Image of hand joints

These bones don't look exactly like the bones in an anatomy book but they will give you good control over the surfaces that make up Melvin's hand. You will even control the two bones coming off to the left and right of the *hand_root* joint so that they rotate inwards to a semi-cupped shape.

You are going to build the hand skeleton as an integrated part of the whole skeletal structure. This means that you will continue building joints off of the existing skeleton hierarchy.



The arm and hand

Joint orientation

To set-up the hand for Forward kinematic motion, you will draw all the joints with **Auto Joint Orient** set to **XYZ**. Once this is complete, you will then confirm that all the rotation orientations for the hand joints are set

properly. You should note that some of the thumb joints will need to rotate in slightly different directions compared to the other fingers therefore you will set up their local axes differently.

With the joint orientations set correctly, you will be able to bend the fingers by rotating them around their local axis instead of in world space.

Creating the middle finger

You will begin building the hand based on both the templated Melvin geometry, and the placement of the current wrist joint. The first joint you create (*hand_root*) will start from the *left_wrist* joint.

Most of the other fingers will stem from the middle finger, so you will create it first. When creating the joints for the hand you won't need to use Grid Snap because the joints will not always be in a straight line.

To add new joints to the end of an existing joint chain is really simple. Basically there are only 2 things to remember:

- Selection masks are setup so joints *can* be selected
- You must LMB click the first *new* joint on the end joint of the existing chain.

1 Open an existing scene file

- Open the scene file called *Melvin_06_readyForHands.mb.*
- Template the geometry *or* toggle off the selection mask for geometry to avoid picking the skin. Make sure that you can select joints since you'll need to start creating them from the wrist joint.

Selection mask buttons

2 Create new joints for the middle finger

In a top view, starting from the wrist joint, create a skeletal chain consisting of 6 joints: 2 joints will serve as the wrist and palm, and 4 as the middle finger. Auto orient the joints to XYZ.

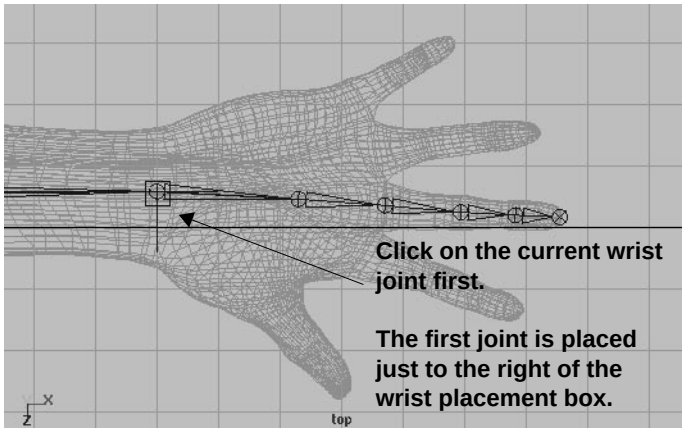
- Click on the *left_wrist* joint to continue the current hierarchy
- Place joints as shown in the following figure.

  The first joint created will become the root joint and eventually be the wrist joint that is attached to the *left_arm*. It should be placed at the edge of the hand geometry where it meets the arm geometry.
- Label the joints:

  hand_root, middle_palm, middle1, middle2, middle3, middle4.

Knuckle Placement

Proper knuckle placement is important to define good deformations later on. The knuckles should be closer to the top of the finger. The first knuckle should be set back in the hand. Look at your own hand for reference and notice where the knuckles are.

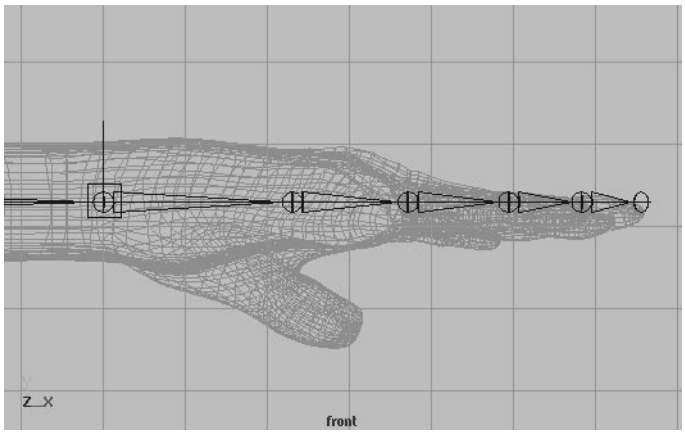


Wrist joints with middle finger

3 Verify the joint alignment in the hand

From side and perspective views, make sure the joints line up in the middle of the hand and are scaled appropriately.

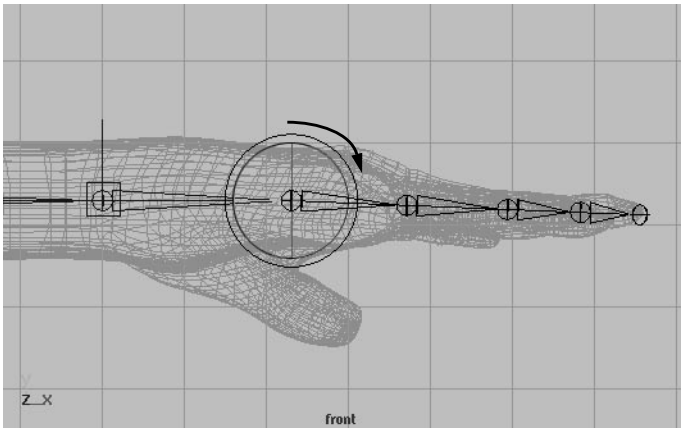
- Select the *hand_root* joint of the *middle_finger* and **Move** it to the middle of the hand and finger.



Side view of hand

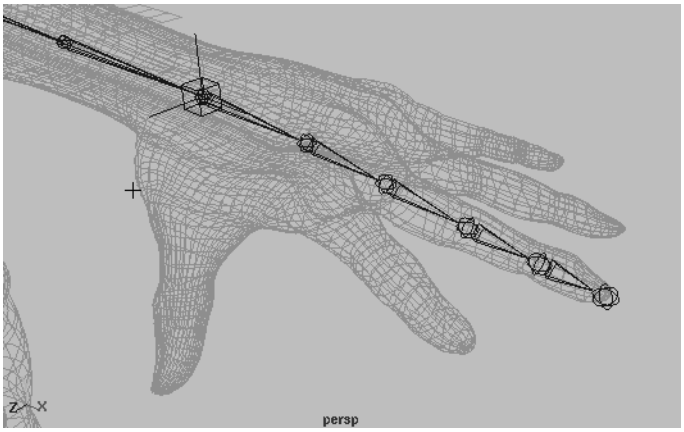
Tip: In the perspective view, select **View** → **Frame Selection** to center the camera around the selected finger joints.

- **Rotate** and **Scale** the joints so that they line up as shown in the following figure.



Rotated joint

Tip: Make sure the **Rotation** manipulator is working in **Local Space**. Check this in the Tool Settings window, accessed by double-clicking the Rotate manipulator icon in the Tool shelf.



Aligned joints

Creating index and ring fingers

The index and ring fingers will stem from the *middle_palm* joint.

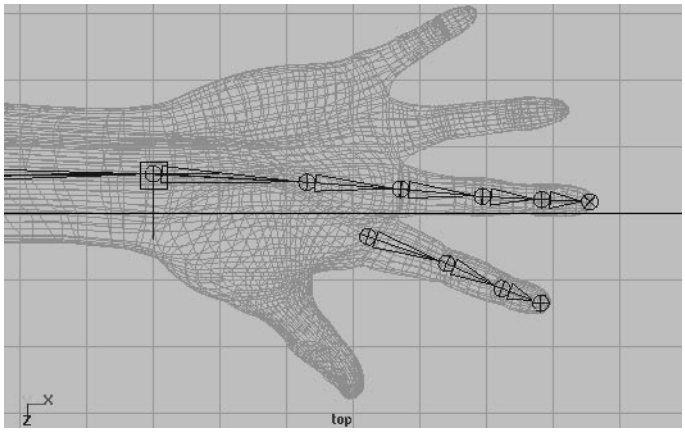
1 Create the index finger joints

- In a top view, create 4 joints for the index finger.

  The first joint should be placed at the knuckle of the index finger. Observe your own hand for clues about its placement.

Knuckle Placement

Proper knuckle placement is important to define good deformations later on. The knuckles should be closer to the top of the finger. The first knuckle should be set back in the hand. Look at your own hand for reference and notice where the knuckles are.

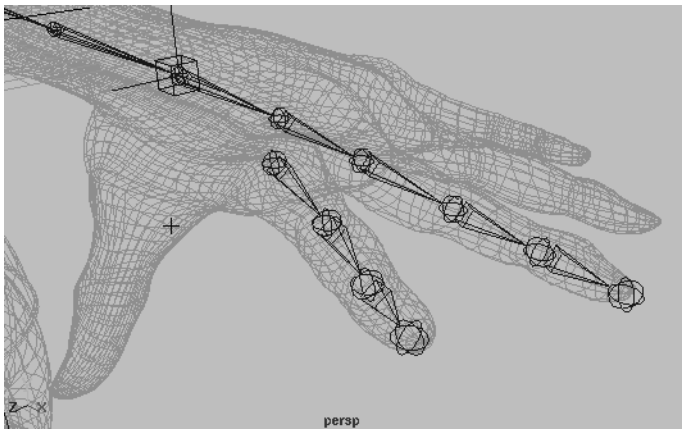


Index finger joints

2 Verify the joint alignment

From side and perspective views, make sure the joints line up in the middle of the hand.

- **Select** the index finger root joint and translate it to the middle of the hand and finger geometry.
- **Rotate** and **scale** the joints so that they line up as in the following figure.
- Label the joints *index1, index2, index3, index4.*



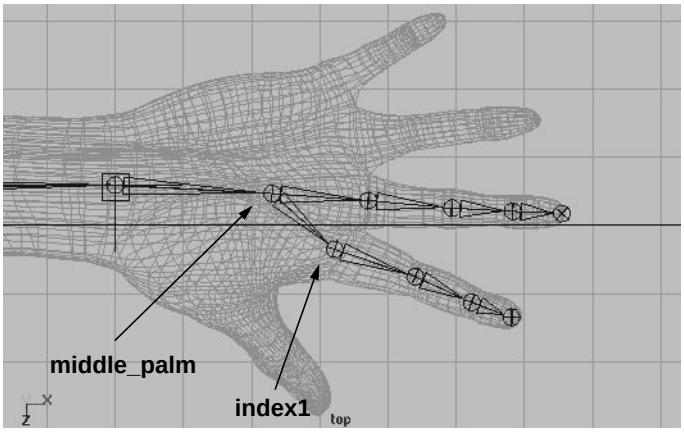
Aligned index finger joints

3 Connect the index finger to the middle palm joint

You will now parent the *index1* joint to the *middle_palm* joint. This will add the new joints to the existing hierarchy.

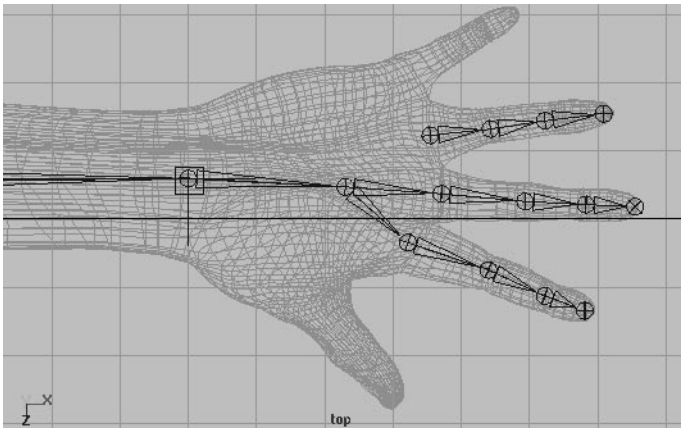
- Select *index1*, then **Shift-Select** *middle_palm*.
- Select **Edit** → **Parent**.

  The index finger is now attached to the *middle_palm* joint.



Connected joints

- Repeat these steps to create the ring finger.
- Name the ring finger joints: *ring1*, *ring2*, *ring3*, and *ring4*.



Ring finger joints

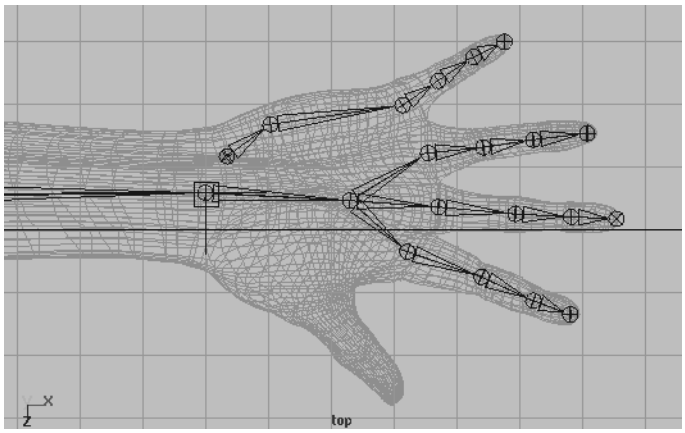
- Parent the ring finger into the *middle_palm* joint.

Creating the pinky finger

The pinky finger will be a little different than the index and ring fingers in that it will start down near the base of the hand. You are not stemming the pinky finger from the *middle_palm* is because you want it to work independently of the three middle fingers, allowing you to pose the hand into a cupped position.

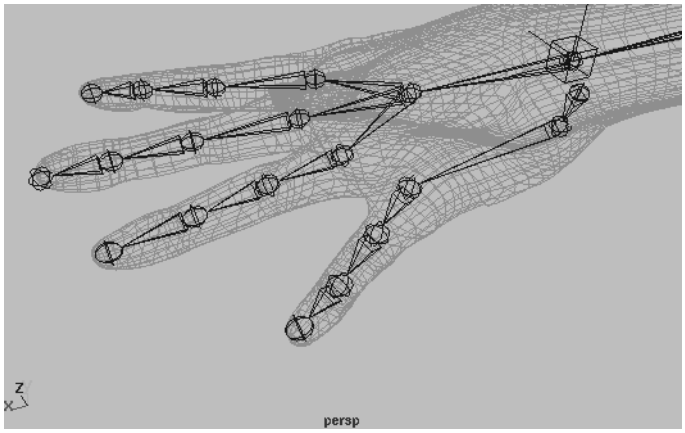
1 Create the Pinky finger joints

In a top view, create 6 joints for the pinky side of the palm as well as the pinky finger.



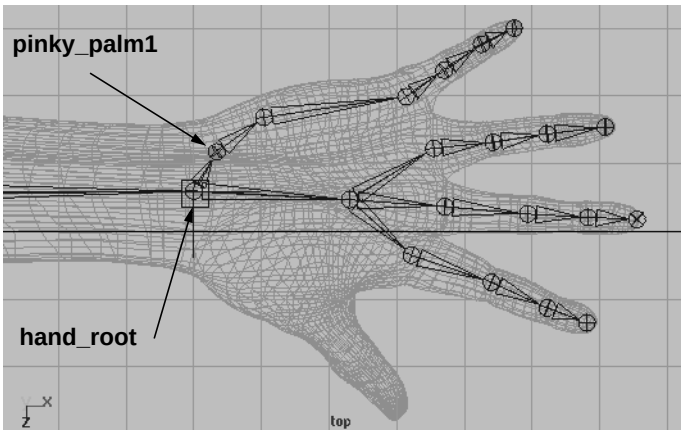
Pinky finger joints

- From side and perspective views, make sure the joints line up in the middle of the hand.



Alignment of joints

- Name the joints: *pinkyPalm1, pinkyPalm2, pinky1, pinky2, pinky3, pinky4.*
- Parent *pinkyPalm1* to the *hand_root* joint.



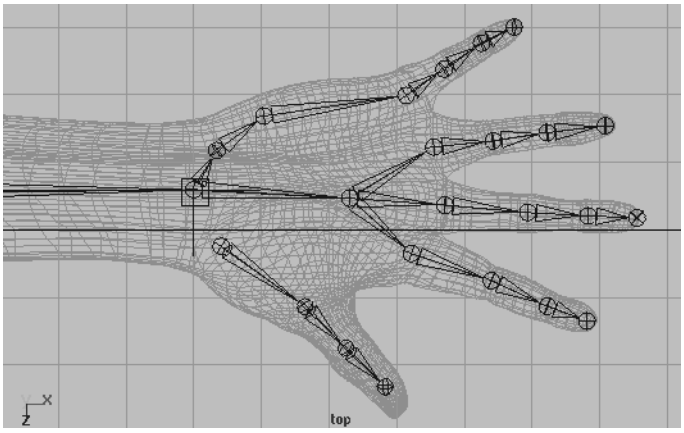
Pinky finger

Creating the thumb

The thumb will be created similar to the pinky but the way it articulates will be different. The first couple of joints stemming from *hand_root* will be ball joints (they can pivot around more than one axis). Later, when checking the rotation axes, you'll need to decide how these first few joints rotate to create realistic articulation of the thumb.

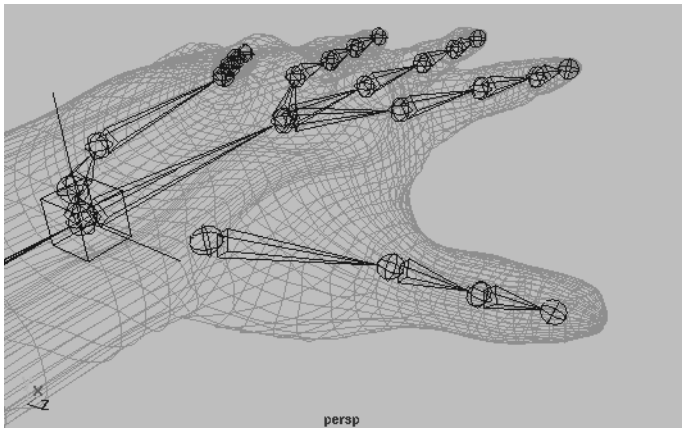
1 Create the thumb joints

- In a top view, create 4 joints for the thumb.



Thumb joints

- From side and perspective views, make sure the joints line up in the middle of the hand.
- **Rotate** the parent joint so all of the joints line up along the thumb.
- **Scale** some of the joints so they extend the length of the thumb.

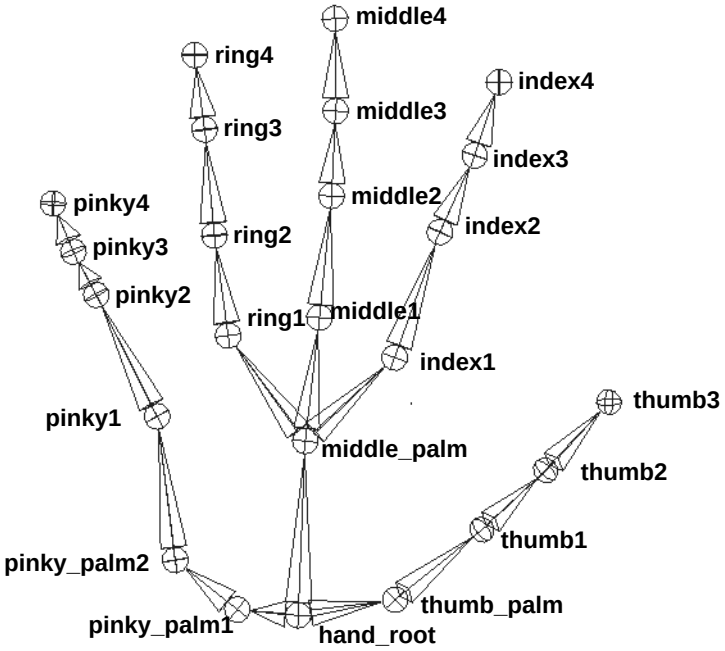


Alignment of thumb joints

- Rename the joints: *thumbPalm*, *thumb1*, *thumb2*, *thumb3*.
- Parent *thumbPalm* to the *hand_root* joint.

Joint names

You should now have a hand constructed and labeled as follows:



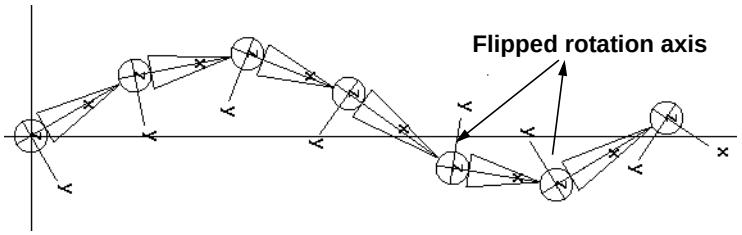
Hand joints named correctly

JOINT ORIENTATION

At the beginning of this lesson, you explored the concept of orienting your joints properly using the joints local axes. The joints for the hand were created with **Auto Joint Orient** set to **XYZ** so, by default, the x-axis will point down the bone. As you've already seen, some of the Y and Z axis might be

flipped. You will verify the local joint rotation axis to make sure there are no reversed axes of rotation. This is visually apparent if you select the joint and toggle Local Rotation Axis from the Component mode Pick Masks.

The following figure shows what you'll see if there is a reversed axis of rotation.



Flipped axes

If you find that a rotation axis is flipped, rotate it to corresponds with the other joints. Having all the joints rotating in the same direction will help you in the next lesson when you will use **Set Driven Key** to set up the finger controls.

Re-AutoOrienting joints

If you use the translate tool while you are editing the locations of the hand joint positions, the orientation axis of the translated joint stays at the orientation in which it was created. This means that it will no longer be aligned with the bone. You must therefore re-orient the joints.

1 Re-orient the joints

- **Select** the joint(s) needing to be re-auto oriented in Object mode
- Enter the following in the Script Editor

  ```
  joint -e -oj xyz;
  ```

  This will re-orient the joint(s) as if they were created with auto orient set to XYZ. The `-e` flag is for edit and `-oj` is for orient joint.
- Verify their orientation by displaying the local rotation axis in Component mode.

A couple other flags can be added to the command above. The `-ch` flag will re-orient all of the children joints below the selected joint. The `-zso` flag will reset the scale axis so it too points down the bone. The full command would look as follows:

```
joint -e -oj xyz -zso -ch;
```

Note: If the joint positions were changed by rotating or scaling, their orientations will be OK. Only when they are translated (the joint moved or the joint pivot moved) should the above workflow be considered.

Adjusting joint orientation

In the first lesson you adjusted the local rotation axis by rotating with the rotate manipulator. In many cases, the axis will only need to be rotated exactly 180 degrees in a certain axis.

Looking at the palm of the hand from the arm, the joint orientations should be as follows:

- X pointing down the joint
- Y pointing towards the pinky
- Z pointing up

1 Hide the geometry.

- Select **Display** → **Hide** → **Hide Geometry** → **All**.

2 Display the rotation axes for all the hand joints

- Select *hand_root* in object mode.
- Enter and execute `select -hi` in the script editor.

  This will select all the joints down the hierarchy from the current selected joint.
- In Component mode, click the Selection mask **?** button. This lets you add **Local Rotation Axis** to the selection mask.

  The local rotation axis is displayed.

3 Rotate flipped joints

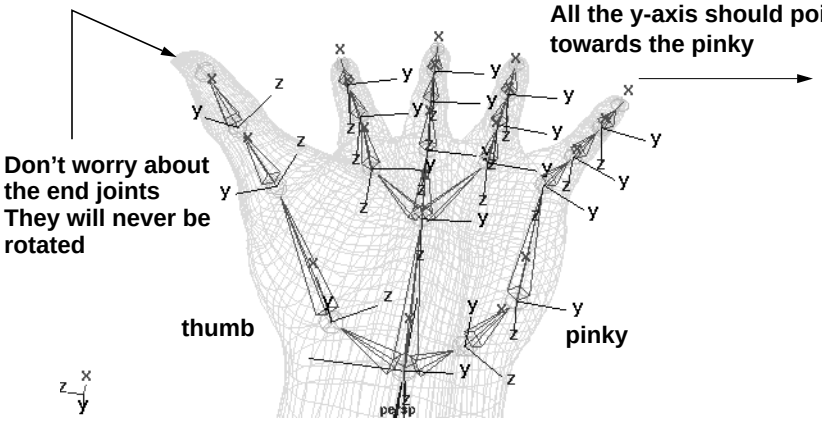
Select any joints that appear flipped by selecting the flipped joint's rotate axis. These are any axes where y does not point in the direction of the pinky except for the thumbs.

- **Rotate** the selected local rotation axes by entering and executing the following in the command line:

```
rotate -r -os 180 0 0
```

Command flags

To view the flags for a command you can type `help command_name` in the script editor.

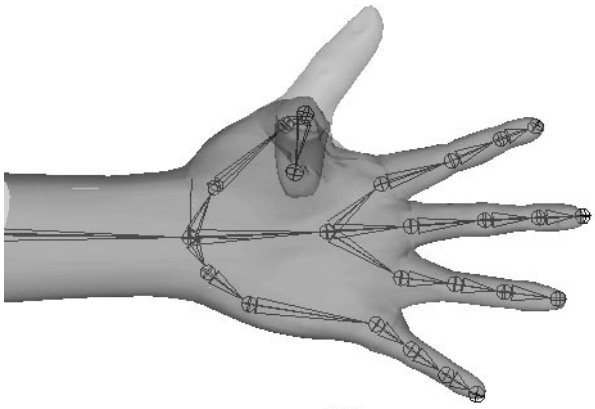


Rotating the local axes on joints

Tip: Use X-ray mode to help see inside the geometry for accurate bone placement. Access X-ray mode in the View panel by selecting **Shading** → **Options** → **Xray**.

Thumb orientation

The rotate axis of the thumb joints will differ from the other fingers. You will need to adjust each of these individually. One of the best references is to look at how your thumb articulates. Notice the thumb1 and thumb2 joints are hinge joints.

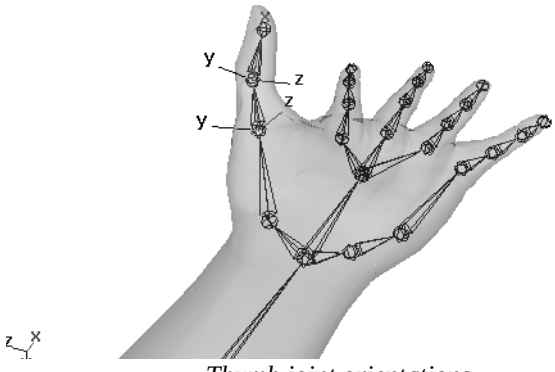


Rotation of thumb

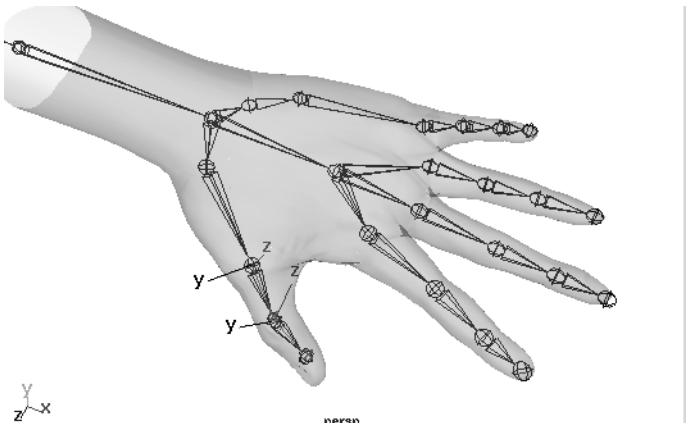
1 Set up the thumb orientation

Align the *thumb1* and *thumb2* pivots so the y axis is perpendicular to the side of the thumb. Both joints will have similarly aligned rotate pivots.

- In Component mode, use the **Rotate** manipulator to adjust the rotate pivots as shown in the following two figures:



Thumb joint orientations



Thumb joint orientations

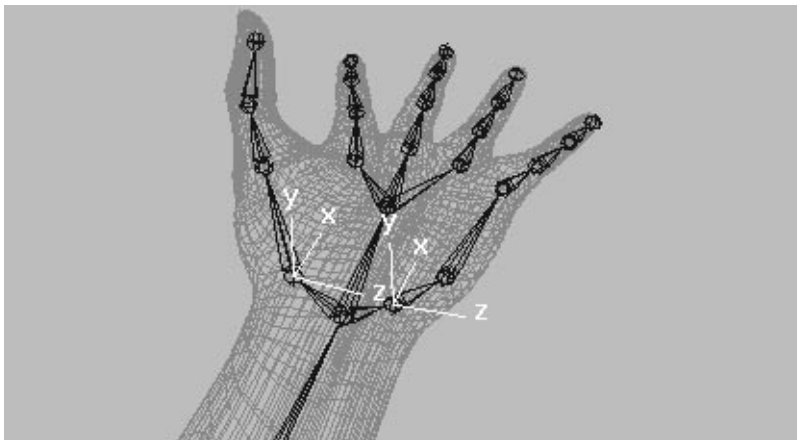
The *thumb_palm* joint should work like a saddle joint - meaning it will rotate in 2 directions.

2 Thumb palm and Pinky palm orientation

To later facilitate the cupping of the hand, the two palm joints need to have their axes aligned with the middle finger. This way the two outside fingers can be easily rotated around the X-axis to bring the fingers in.

- Align the x-axis of the *thumb_palm* joint with the direction of the middle finger.
- Display the geometry to see how the thumb will be oriented at bind pose.
- The *pinky_palm1* joint orientation needs to be aligned similarly to the *thumb_palm* joint (aligned in the direction of the middle finger).

Tip: You can go back after the skin has been bound and fine tune these orientations.



Palm orientation to pinky finger

Note: Some people like to set the local axis of the *pinky_palm1* joint and the *thumb_palm* joint to both point into the center of the hand. This means a positive rotation will point them in and a negative rotation will point them out.

Setting the preferred angle

Now you have created all the joints for our hand. Assuming everything positioned correctly, you need to tell Maya what all the joints' preferred angles are. This will allow you to return to the preferred position later in the lesson when you start changing the joint rotations.

1 Set the preferred angle for the hand joints

- Select the *hand_root* joint.
- Select **Skeleton** → **Set Preferred Angle** - ❒ and set the following:

  Recursive: On

  This will set the preferred angle for all the joints below the *hand_root* joint.

Tip: Confirm the preferred angle by rotating some of the joints, then selecting the hand_root and **Skeleton** → **Assume Preferred Angle**.

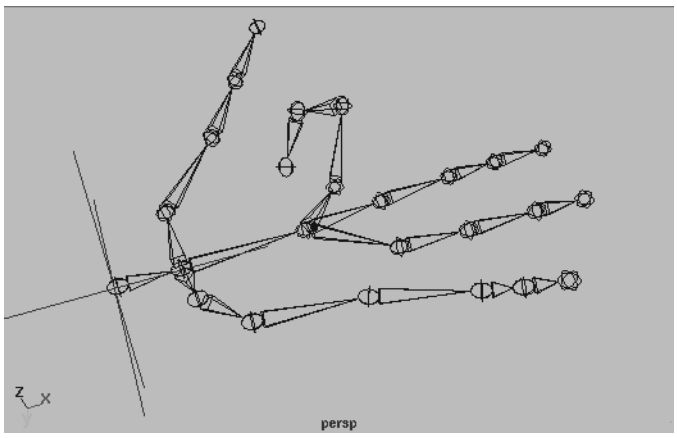
Summary

You have now completed and are aware of the following tasks:

- Adding joints to the end of an existing joint hierarchy
- Verifying and aligning the joint's local rotation axes
- Building a hand skeleton
- Joint placement for proper skinning and motion
- Testing the motion using Forward Kinematics

7 The Hand Controls

In the previous lesson you set up all the joints for Melvin's left hand. At this stage the hand could be animated by rotating each finger joint using forward kinematics. Because this could be a tedious task, this lesson will focus on developing a set of controls that will drive the rotations of all the hand joints. You will use Set Driven Key to control the fingers and movement of the hand.



Hand controls

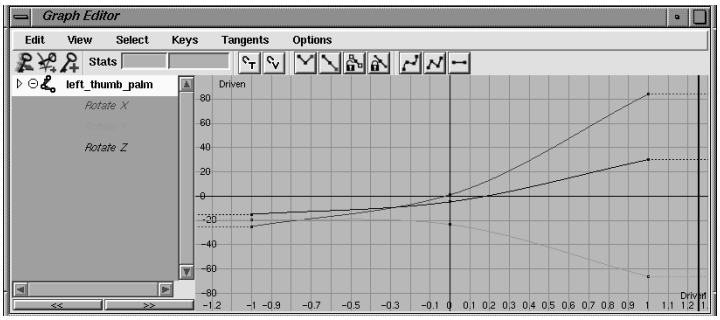
In this lesson, you will learn the following:

- How to add finger control attributes to the wrist locator
- How to use Set Driven Key to control the finger joints
- How to make connections with the Connection Editor

Lesson 7

Set Driven Key Basics

It is important to understand that Set Driven Key is a curve relationship between two attributes in Maya. In the Graph Editor, the horizontal axis represents the driver attribute values and the vertical axis represents the driven attribute values. The curve represents the relationship between these two attributes.



Graph editor view of set driven key

Because Set Driven Key is a curve relationship, it is possible to adjust the tangents of this curve and add additional keys. This will allow for some interesting behavior - for example if the rotate attribute of an elbow is driving the size of a bicep muscle, the curve could be edited so that when the elbow reached its maximum bend, the bicep shakes a little as it flexed.

LOCATORS AND ADDING ATTRIBUTES

You can now add a locator to the hand to use as a manipulator and a control center for articulating the fingers. Creating a single place for the control of the hand means the animator does not have to search around for the necessary controls—they will be on the manipulator that is controlling the placement of the hand (a logical place to put the finger controls). You will be adding attributes to this locator so you can keyframe and control all the hand motion by selecting one object.

Creating the hand control

You'll start by creating a locator for the hand control. You will add attributes to it so you can keyframe and control all the hand motion by selecting this locator.

1 Open an existing file

- Open the file *Melvin_07_hands.mb.*

2 Add attributes to the Locator

You will add attributes to the *L_wristLocator* to control the fingers.

- **Select** *L_wristLocator.*

- Select **Modify** → **Add Attribute**. Set the following:

  Attribute Name to *indexCurl*;

  Make Attribute Keyable to **On**;

  Data Type to **Float**;

  Attribute Type to **Scalar**;

  Minimum Value to **0**;

  Maximum Value to **10**;

  Default Value to **0**.

- Click the **Add** button.
- Repeat the steps outlined above to add the following attributes:

  middleCurl, ringCurl, pinkyCurl, *pinkyCup, thumbCurl.*

- Add the following attributes to *L_wristLocator*:

  fingerSpread, thumbRotX, and *thumbRotZ* and set the **Min**, **Max**, and **Default values** to **-10**, **10**, and **0** respectively.

  These are the attributes that you will control with Set Driven Key. They now show up in the Channel Box for the *L_wristLocator*.

SET DRIVEN KEY TO CONTROL THE FINGERS

Now that you have all the attributes to eventually control the articulation of the fingers, you need a tool to connect the two. Set Driven Key is perfect for this task because it sets up a relationship between two attributes.

Bending fingers with Set Driven Key

In the case of bending the index finger, you can have its joints rotate when you change the value for the indexCurl attribute. When indexCurl is set to 0, none of the index finger joints will be rotated, but when you change indexCurl to 10, the joints will rotate. For motions like spreading the fingers, the Min and Max should range from -10 to 10, where -10 moves the fingers closer together and 10 moves them spread apart.

When setting up the Set Driven Key, you can do all the selecting and manipulating in the work area or use the Outliner or Hypergraph if selecting becomes tedious. The objects can be selected from inside the Set Driven Key editor as well.

Driving the index curl

The techniques you use to set up the index curl are the same for all the Set Driven Key set ups. You will set the controls so you can curl the finger by changing the value of the indexCurl attribute.

1 Open the Set Driven key window

- Select the **Animate** → **Set Driven Key** → **Set** - ❐.

The Set Driven Key window appears. It is divided in two parts: Driver and Driven. The attributes you just created will be the *drivers* and the joint rotations on the hand will be the *driven* objects.

2 Select the Driver node and attribute

- Select *L_wristLocator*.
- Click **Load Driver**.

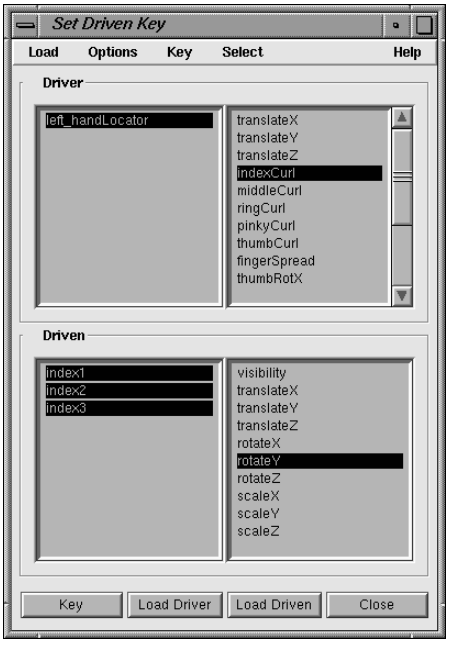
  Notice that *L_wristLocator* appears in the list of Drivers. You should only see one driver object.
- Select **indexCurl** from the list of keyable attributes.

3 Select the Driven node and attribute

In the Set Driven Key Editor, you will use the **rotateY** attribute of the index joints as the driven attribute. To rotate the joint around one axis, you only need to drive that rotation attribute.

- Shift select the 3 index joints (**index1, index2, index3**).
- Click **Load Driven**.

  Notice that the selected objects appear in the Driven Objects list.
- Select the driven objects, then select **rotateY** from the list of attributes (refer to the following figure). Note that the local rotation attributes are set up so that the fingers (which are hinge joints) only need to rotate around one axis.



Set Driven Key window

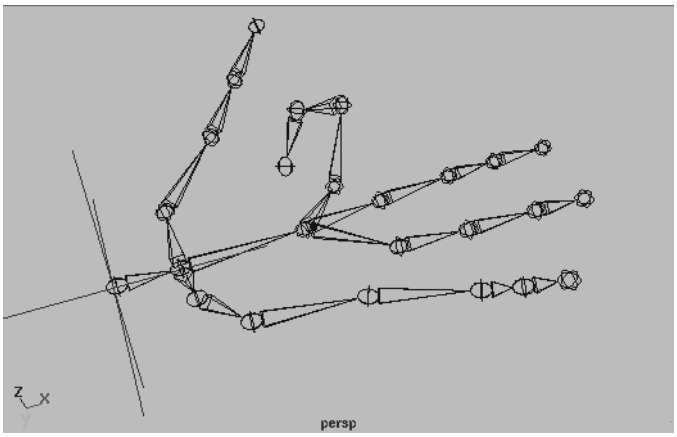
4 Set an initial key position

- Select the *L_wristLocator* to verify that **indexCurl** is set to **0**.

- Click **Key** in the Set Driven Key window.

5 Set a second key position

- In the Set Driven Key window select **L_wristLocator** by clicking on its name and in the channel box, set **indexCurl** to **10**.
- Curl all the joints so that they rotate about 90 - 100 degrees. Check the limits of the rotation of your own finger for reference.
- Press **Key**.



Rotating finger joints

Tip: It is better to "over-rotate" the joints. If you don't rotate the joints enough, you may have to edit them later. If you over-rotate the position, however, you don't have to move the attribute to its full range.

6 Test the values

Select the *L_wristLocator* and test different values for indexCurl between 0 and 10.

- In the channel box, click on the name of the attribute (it should highlight to designate that it is selected).
- In the camera viewport, drag with the **MMB** to change the value of the selected attribute in the channel box.

7 Use Set Driven key for the other fingers

- Repeat steps 1 to 6 for the middle, ring, pinky, and thumb joints.

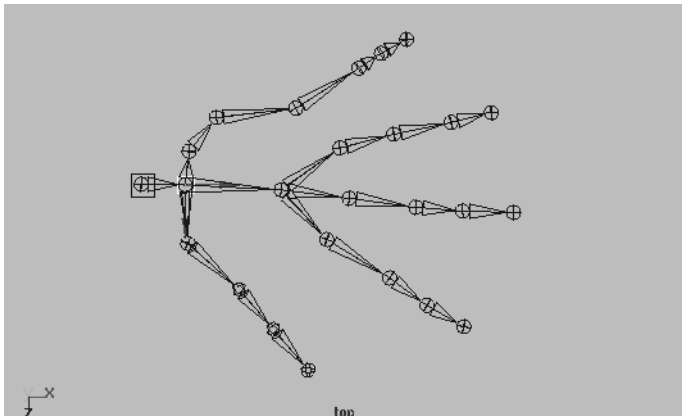
Note: The thumb has 2 joints to curl and the fingers have 3.

Driving the finger spread

You also want the hand to be able to spread its fingers. Use Set Driven Key again to control the action. This time you'll use attributes that have a range between -10 and 10, with 0 being the rest position or preferred angle.

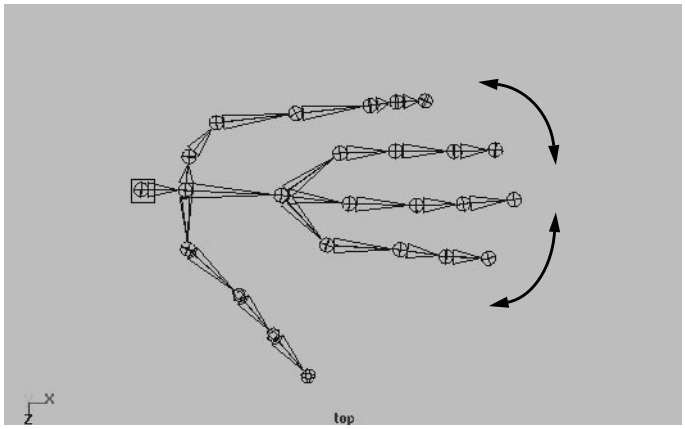
1 Use Set Driven Key to drive the finger spreading

- Load the *L_wristLocator indexCurl* as the **Driver** attribute
- Shift select *index1, middle1, ring1, pinky1*.
- Click **Load Driven**.
- Select the joints and the corresponding keyable rotation attributes.
- Set a key with **fingerSpread** at **0** and the selected finger joints in their rest position.
- Set a key with **fingerSpread** at **10** and the selected finger joints spread to their widest position.



Open finger spread

- Set a key with **fingerSpread** at **-10** and the selected finger joints in a closed position.



Closed finger spread

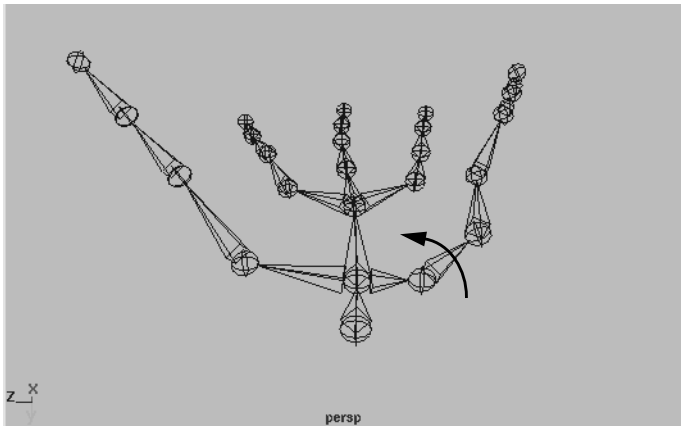
2 Test the results

- Test the range of motion between **-10** and **10** by changing the **fingerSpread** attribute.

3 Save you work

Driving the pinky cup

Another function of a real hand is the ability of the palm to form a cup. This is achieved by rotating the bones of the pinky side of the hand. You'll again control this action using Set Driven Key to rotate the pinky_palm joint around the axis that is parallel to the direction of the middle finger.



The cupping of the pinky

1 Use Set Driven Key

- Load the *L_wristLocator pinkyCupl* as the **driver**.
- Load the **pinkyPalm** rotation attributes as the driven objects.
- Set a key with **pinkyPalm** set to **0** and all the joints at their preferred angle.
- Set a key with **pinkyPalm** set to **10** and the *pinkyPalm* joint rotated up.

2 Test the results

- Select the *L_wristLocator* and test different values for pinkyPalm between **0** and **10**.

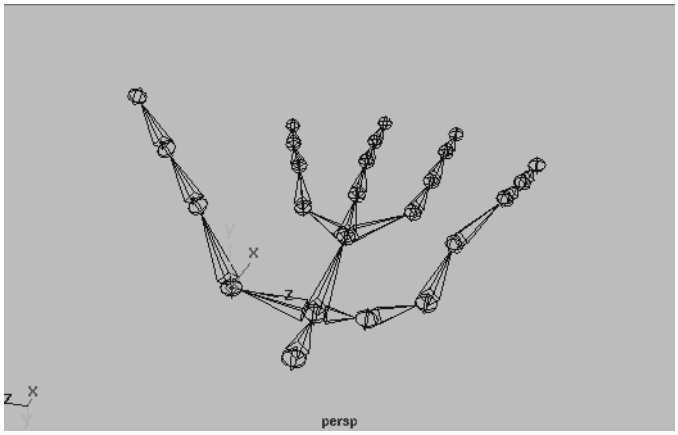
You now have the ability to cup one side of the palm. The other side will get its motion from the movements of the thumb

THE THUMB

The thumb is different from the other fingers in that its base pivots on a saddle joint and has much more freedom of movement than the finger joints. Where the fingers rotate around one hinge joint, the saddle joint will rotate around 2 axis.

When you set up the thumb motion, you need to allow for flexible articulation that mimics the orbiting provided by a saddle type joint. This is broken down into RotateX and RotateY. It also requires you to change

the rotation axis so that it is not aligned with the direction of the joint, but with the direction of the hand, as shown in the following figure.



Thumb Axis

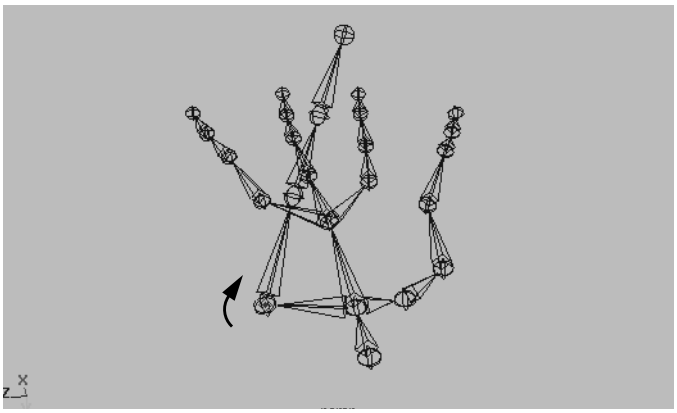
Driving rotation of the thumb

The following exercise is the same as all the Set Driven Keys for the other fingers. Because this thumb joint articulates differently, it is important to make sure that the local rotation axes are set correctly before creating any Set Driven Keys.

- **x-axis** is parallel to the direction of the middle finger
- **z-axis** is normal to the palm

1 Drive the rotation of the thumb

- Select **Animate** → **Set Driven Key** → **Set -** ❐.
- Select *L_wristLocator* and click **Load Driver**.
- Select *thumbPalm* and click **Load Driven**.
- Select **thumbRotX** as the Driver attribute and **rotateX** as the Driven attribute.
- Set a key with **thumbRotX** at **0** and the **thumbPalm** joint at its rest position / orientation.



Thumb rotation

- Set **thumbRotX** to **10**.
- Rotate the *thumbPalm* in the x-axis, to the point where the thumb crosses the palm towards the pinky.
- Click **Key**.

2 Set the second key position

Set a key with **thumbRotX** at **-10** with the **thumbPalm** joint x-rotated out towards the back of the hand.

- Set **thumbRotX** to **-10**.
- Rotate the thumbPalm in the x-axis, so that the thumb is rotated to where the hand is in a flat pose.
- Click **Key**.

3 Test the operation of the thumb in this direction

Driving thumbPalm joint with thumbRotZ

Combined with the motion above, setting up this motion will allow the thumb to touch the base of the index finger.

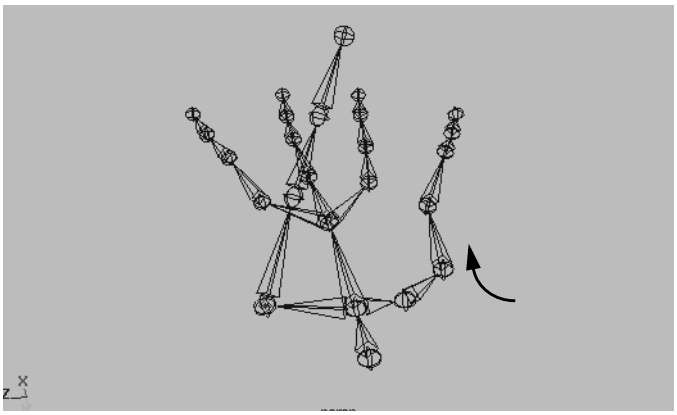
1 Load the Driver and the Driven attributes

In the Set Driven Key window load the *L_wristLocator* as the driver and the *thumbPalm* joint as the driven object.

- Select **Animate** → **Set Driven Key** → **Set -** ❐.
- Select *L_wristLocator* and click **Load Driver**.
- Select *thumbPalm* and click **Load Driven**.
- Select **thumbRotZ** as the Driver attribute and **rotateZ** as the Driven attribute.

2 Set Keys for these attributes

- Set a key with **thumbRotZ** at **0** and the *thumbPalm* joint at its rest position.
- Set a key with **thumbRotz** at **10** and the *thumbPalm* joint rotated in z towards the index finger.



Rotations of thumb

- Set **thumbRotZ** to **10**.
- **Rotate** the thumbPalm in the z-axis, where the thumb crosses the palm towards the base of the index finger.
- Click **Key**.

3 Set another key

Set a key with **thumbRotZ** at **-10** with the *thumbPalm* joint z-rotated out towards the wrist.

- Set **thumbRotZ** to **-10**.
- Rotate the thumbPalm in the z-axis, where the thumb is rotated in the direction of the wrist. The thumb should be almost perpendicular to the palm.
- Click **Key**.

4 Test the operation of the thumb in this direction

5 Save you work

Test the Controls

One of the true tests of articulating the hand is to form the fingers into a cupped shape. Here are a couple poses to try out by adjusting all the finger attributes:

- Try to make the pinky and the thumb touch
- Shape the fingers into a cup
- Create a sphere and place it in the palm of the hand then wrap the fingers around the ball

Controlling the wrist with Attributes

The *wristLocator* will be used to position of the hand.

By adding 3 new attributes to this locator, you can drive the rotation of the wrist. Two of these attributes, wristUpDown and wristSide will drive the y and z rotation of the wrist joint. For the wristTwist attribute, the motion

should not be driven from the wrist joint, but from the forearm joint. Assuming all the local rotation axes are set correctly (x axis points down the bone), the wristTwist can drive the x rotate attribute of the forearm joint.

1 Create new attributes for the wrist locator

On the *L_wristLocator*, create 3 new attributes: *wristTwist*, *wristSide*, and *wristUpDown*.

2 Make a connection

In the connection editor you will set up the connection: *L_wristLocator.wristTwist* → *left_forearm.rotateX*

- Select **Window** → **General Editors** → **Connection Editor...**
- Select *L_wristLocator* and **Reload Left.**
- Select *left_forearm* and **Reload Right.**
- Select *L_wristLocator.wristTwist* and then select *left_forearm.rotateX*.

  This will make the *L_wristLocator.wristTwist* attribute control the *left_forearm.rotateX*.

3 Make two more connections

- Repeat the previous step to make the following connections:

  L_wristLocator.wristSide → *left_wrist.rotateY*

  L_wristLocator.wristUpDown→ *left_wrist.rotateZ*

4 Save your work

Now all the hand articulations (fingers, position, and rotation) can be controlled by selecting one object: *L_wristLoctor*. When animating you should be able to use your selection masks so you can select locators and selection handles only to avoid picking and animating the wrong object.

Summary

This lesson you focussed on the workflow of adding attributes to drive the rotation of the joints. By now you should be comfortable with the general Set Driven Key workflow. This is a very powerful tool that is used constantly in all areas of Maya. You should also be comfortable with using the connection editor to make more direct relationships.

Connection Editor

The Connection Editor sets up direct relationships between channels. The input channel will have the same value as the output channel.

More complex relationships can be created by adding one of the utility nodes to the stream.

Lesson 8

8 Smooth Skinning

In this section, you will explore Maya's Smooth Skinning function. Smooth skinning provides smooth deformations around joints by allowing multiple joints to have influence on the same CV.

In later lessons, you will examine other methods of skinning geometry to skeletons such us rigid binding and indirect binding with lattices and wrap deformers.

In this lesson, you will learn the following:

- How to Smooth Bind surfaces to bones.
- Editing weights with the Paint Skin Weights Tool
- Tips and Tricks for weighting smooth skinned surfaces.

SMOOTH SKINNING MELVIN

Bind Skin is probably the most common technique of attaching geometry to skeletal joints. With smooth bind, the bound skin points are weighted across many different joints, with each joint having a different weight of influence depending on the joint's hierarchy and / or distance from the particular skin point.

A bound geometry point (CV's, poly vertices, lattice points) can be thought of as "skin points" where they are put into a skin cluster node after they are bound. These points all have weight of 1.0, but the weights can be shared between many different joints and influences.

1 Open an existing file

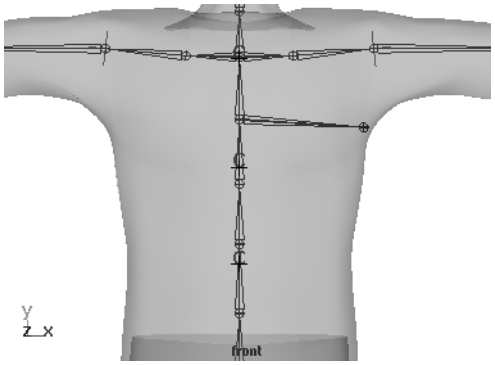
- Open the scene file *Mlevin_08_handControls.mb.*

2 Add ribs to the skeleton

When smooth binding a skeleton, it is a good idea to add a few more joints in areas that might need more influence when you deform the character. These extra joints help smooth out deformations and help to set up the character for less weighting later on. They also help the geometry maintain volume under extreme deformations. For Melvin, you are going to give him some rib bones. These new rib bones will help smooth out deformations around the shoulder and the torso area.

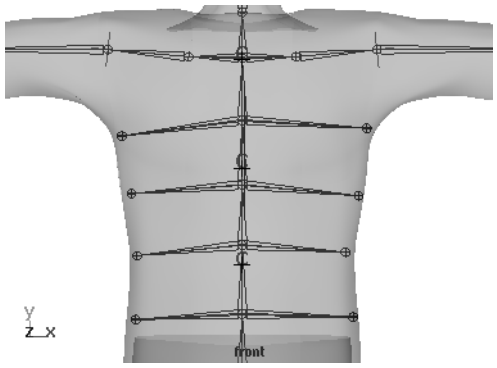
- Select the **Skeleton → Joint Tool.**
- Select the *back_e* spine joint.
- Add a rib bone by clicking to the side of Melvin's body.

  Try to make the rib joint parallel to the arms. Make sure that you are in the front view when you are adding rib joints.



Melvin with one rib added.

- Continue to add rib joints until there are a total of 8 rib joints, 4 on each side.



Melvin's completed rib cage.

3 Hide all of the surfaces that don't need to be bound

There are many surfaces in the scene that don't need to be bound right now.

- Select the *Head, PositionMarkers*, and *Hair* groups in the Outliner or the Hypergraph.
- Hide the surfaces by Selecting **Display** → **Hide** → **Hide Selection**.

  The head will be dicussed in a later chapter.

4 Select the surfaces to be bound

- Turn all of your pick masks **off** in the status line.
- Turn on only the **Surfaces** pick mask.
- Select all of the surfaces in the scene by drawing a selection box around Melvin.

The status line with the Surfaces mask turned on.

Note: Double check that you are binding the correct surfaces. Do not have the *low_Res_Melvin* layer visible when you are binding skin. If those surfaces are displayed, they will have double transforms on them since they are also parented to the joints.

---

5 Select root skeleton

- Control select the *back_root* joint in the Outliner

6 Smooth Bind Melvin

- Select **Skin** → **Bind Skin** → **Smooth Bind -** ❐**,** and set the following:

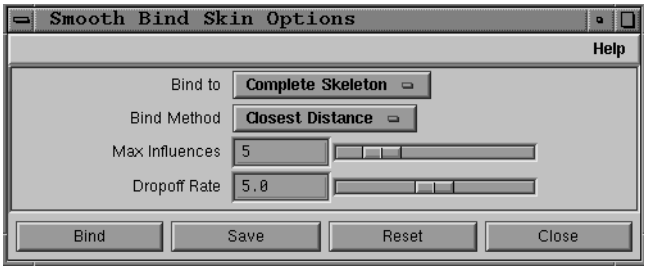
  Bind to to **Complete Skeleton**

  Bind Method to **Closest Distance.**

  Max Influences to **5**.

  Dropoof Rate to **5.0.**

- Press **Bind** to attach (bind) the skin to the skeleton and establish weighting.



The Smooth Bind options window.

Binding by **Closest Distance** specifies that joint influence will be based *only* on the distance between the skin points and the joints. This method ignores the hierarchy of the skeleton.

Max Influences is the number of joints that will have influence on a individual skin point. Setting the **Max Influences** to 5 means that each skin point will not have more than 5 joints affecting it.

Setting the **Dropoff Rate** is another way to determine how each skin point will be influenced by different joints. The Dropoff Rate controls how rapidly the influence of each joint on individual skin points are going to decrease with the distance between the two. The greater the dropoff, the more rapid the decrease in influence with distance.

Tip: The Dropoff Rate can be adjusted on individual joints after the character is skinned up. The Max Influences can also be adjusted after the character has been skinned except the new setting takes effect only on the selected surfaces instead of the entire character.

---

7 Test the Results

- Test the results of the smooth bind by moving Melvin's arms and legs. Pay particular attention to the shoulder, elbow, and pelvis areas.

8 Save your work

EDITING WEIGHTS

Weighting a character has traditionally been a long and tedious task. Maya's smooth bind function eases the burden of this process by setting up good deformations by default.

When Melvin was bound with the smooth bind function, a *skin cluster node* was created for each of the surfaces that was bound to the skeleton. Each of those bound skin points' weights are shared between multiple joints, and those are weighted accordingly to the distance between the joints. When you weight a character, you are actually reassigning those weights to different joints to achieve better deformations.

After moving Melvin around a little bit, you may notice that the default settings of the smooth bind give very good deformations, but it does leave some problem areas such as the pelvis, shoulders, and torso. These areas can be improved by editing the weighting of the skin points for the different joints that they are influenced by.

Smooth the Hips

The first area that you are going to smooth out is the hips. You will put Melvin in a pose that illustrates problem areas and smooth out the deformations using the **Paint Skin Weights Tool**.

1 Switch Display Layers

To increase interactive performance you can switch between a hi and low res version of Melvin. Use the *lo_res_Melvin* layer to position Melvin and *hi_res_Melvin* layer to examine the quality of the skinning.

- Show the **Layer Bar** by selecting **Options** → **Layer Bar**.

The Layer Bar with hi and lo res display layers

- **RMB**-click on the *hi_res_Melvin* layer and uncheck **Visibility.**

  This hides all of the high resolution surfaces on Melvin. Melvin's bones will still be visible in the viewport.

- Right click on the *lo_res_Melvin* layer and check **Visibility** to **on**.

  This displays the low resolution version of Melvin. This geometry is parented to the various bones in the skeleton.

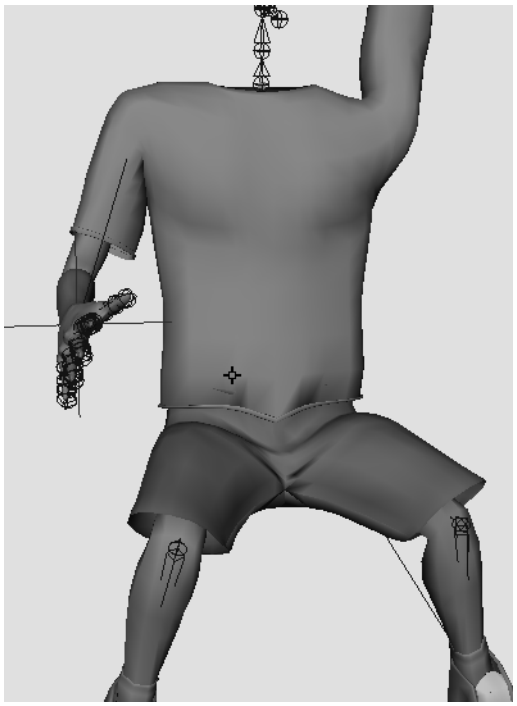
2 Pose Melvin to show problem areas

Positioning Melvin in somewhat of an extreme position makes it easier to determine which areas of the skinning need to be adjusted.

- Rotate the shoulders so the left arm is oriented vertically.
- Bend the hips and knees.

3 Display in Shaded View

- Press **5** to display the shaded view.
- Press **3** to increase the resolution of the view.



Melvin posed with default weighting

4 Show the High Resolution Geometry

After Melvin is in a pose that you like, display the high resolution geometry so you can see details in the skinning.

- Turn the visibility **off** for the *lo_res_Melvin* layer.
- Turn the visibility **on** for the *hi_res_Melvin* layer.

  This pose allows you to see a few problem areas when the arms and legs are rotated. The hips, shoulders, and torso are the most obvious areas that need to be re-weighted and smoothed out.

5 Hide the shirt, arms, legs and shoes

To avoid painting weights on other surfaces, hide the parts of Melvin that you are not going to work on.

- Select the **shirt, arms, legs, and shoes.**
- Select **Display** → **Hide** → **Hide Selection**

6 Select the surfaces around the hip

Melvin is built from many different NURBS patches. When working in a particular area, it is best to select all of the surfaces connected to it so all of the seams will stay tangent and together.

- Select all of the patches on the shorts.

- Select **Skin** → **Edit Smooth Skin** → **Paint Skin Weights Tool - ❐**

Note: The **Skin**→ **Edit Smooth Skin** → **Paint Skin Weights Tool** is exclusive to smooth bound skin. The **Deform** → **Paint Weights Tool** only works on rigid bound skin.

7 Adjust options in the Paint Weights Tool

There are a number of options that need to be changed in order for Artisan to work properly across multiple surfaces.

- Within the **Seam** tab set the following:

  Averaging Seam to **on**

  Averaging Pole CVs to **on**

  These options enable Artisan to keep multiple surfaces and CV's together at their seams.

- Within the **Misc** tab set the following

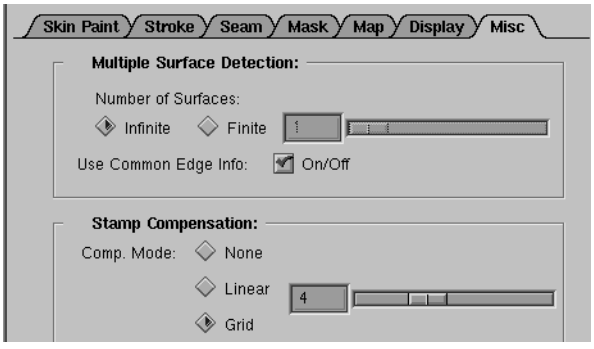
  Number of Surfaces to **Infinite**

  Use Common Edge Info to **on**

  Stamp Compensation to **Grid**

  These settings for **number of surfaces** and **use common edge info** ensure that the brush strokes across multiple surfaces behave as though they are being applied to one contiguous surface.

  Setting **stamp compensation** to **grid** provides a more consistent brush shape when "painting" over rough surfaces.



The Paint Skin Weights window with the Misc tab displayed.

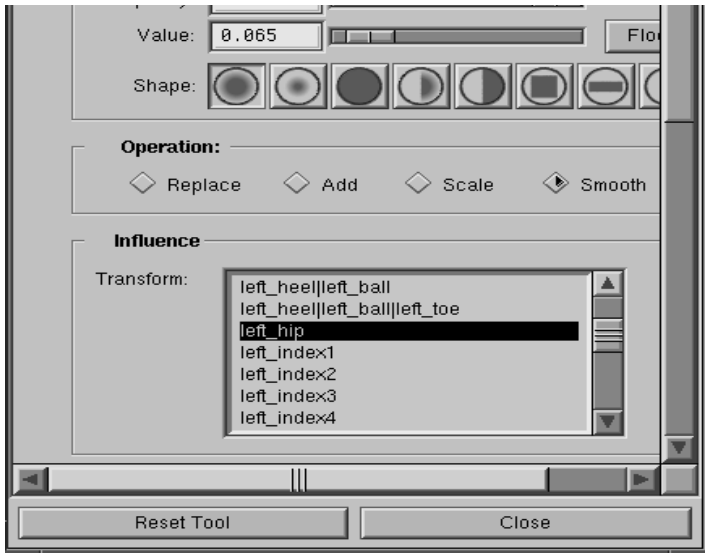
- Within the **Display** tab set the following:

  Color Feedback to **on**

  This allows you to see a greyscale representation of the weighting values associated to the bound surface being painted. White corresponds to a value of **1**, black a value of **0**. The shades of gray represent a value between 0 and 1.

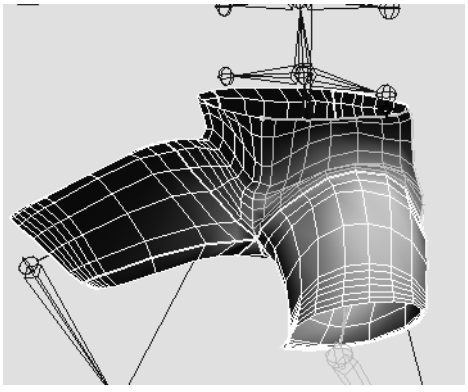
8 Select the left hip joint as the influence

- Within the **Skin Paint** tab in the option window, select *left_hip* as the **influence** that will be painted on.



Selecting left_hip as the Influence to be painted

You may also notice yellow borders displayed between individual patches. Artisan recognizes these borders as seams between different patches and attempts to keep multiple surfaces together even if they are not stitched. The shirt and shorts are made up of a number of different patches, none of which are actually stitched together.



The default weighting of the left hip with color feedback enabled

Tip: Another way to select and paint weights on the *left_hip* is to select the **Paint Skin Weights Tool**, **RMB-**click over the joint itself, then select **Paint Weight** from the resulting marking menu.

9 Adjust the Brush Size and Smooth the Weighting

In the **Skin Paint** tab of the Tool Settings menu, set the following:

Radius(U) and **Radius (L)** to **~1.5.**

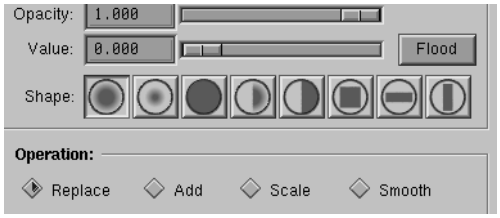
Operation to **Smooth**

- Paint on the surface so the area around the hip has more of a evenly deformed appearance.

10 Flood the Root Weighting to zero

The root joint was bound like every other joint in the hierarchy and was assigned weight to it. For this character set-up, the root joint is only used to translate and rotate the entire character, not for deformations. Therefore, the weighting can be removed for any skin points associated with the root joint.

- Select *back_root* as the **influence**.
- Set **Value** to **0** in the **Stamp Profile** section **.**
- Set **Operation** to **Replace.**
- Press the **Flood** button.



Using the Flood button to set all weights to a common value

By pressing the Flood button, you are replacing all of the skin point weights with a value of **0**. If the operation selected was set to **Add**, the **Flood** button would add the specified value to all of the skin point weights.

11 Weight the back joints

- Select *back_a* as an **influence** to paint on.
- Select **Smooth** as the operation and smooth the weighting with brush.
- Repeat the above steps with *back_b*.

12 Smooth the Pelvis

The hip area has many influences affecting those skin points and the pelvis should not be forgotten. The *left_pelvis* should have its weighting smoothed out to help out the deformations around the hips.

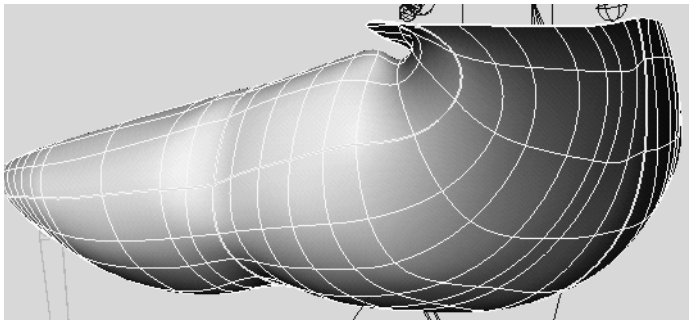
- Select the *left_pelvis* as an **influence**.
- Smooth the weights on the shorts.

Tip: Don't overwork a pose. It is easy to spend lots of time trying to get a pose into a perfectly weighted state while forgetting to test the skeleton in different poses. It's also important to consider that tiny imperfections may not be visible when a texture map is applied or if the camera is not far from the problematic region.

13 Replace Value at the Seam

Although Artisan will keep the different seams together, sometimes it has problems with the tangency of the CV's between the seams. To get around this, **replace** the values of the seam area with a common value. Then, smooth the results.

- With the *left_hip* selected, set the operation to **Replace** and adjust the **Value** to **0.967**.
- Paint the new value on the surfaces around the seam.



Painting weights across multiple seams while maintaining tangency

14 Smooth the Area

After replacing the weights on the hip, you will need to Smooth out the area again with the new weights. Try to avoid smoothing too much over the seam.

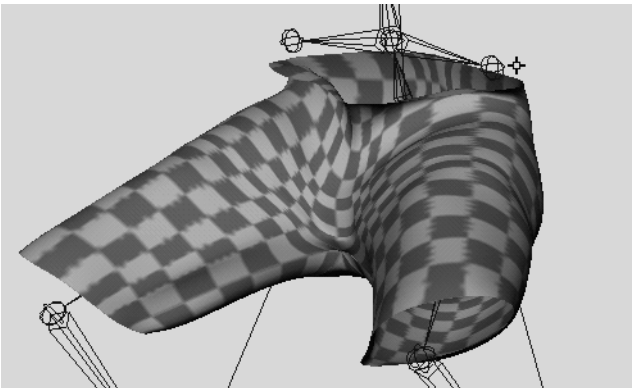
- Set the **Operation** to **Smooth** and smooth the weights in the hip area.

15 Test the Results

Move the leg around to see if there are any other areas in the hip region that need to be worked on. It is very common to have a surface look good in one pose but not have it work well in others.

16 Weight the other Hip

After you have got a successful results from the *left_hip*, try to repeat the procedures to the *right_hip*.



The hips with the skinning corrected

17 Save your work

Smooth the Torso (Optional)

After you've had some practice with the hips, its time to move on to the torso area. The shoulders and the torso have traditionally been one of the hardest areas to skin correctly, but Maya's smooth bind function makes the process easier and quicker.

In this lesson, don't worry to much about smoothing the actual shoulder bone. You will be learning methods in the next chapter to correct the deformations around the shoulder.

1 Open file

- Open *Melvin_08_hipsDone.mb.*

2 Hide Non Essential Surfaces

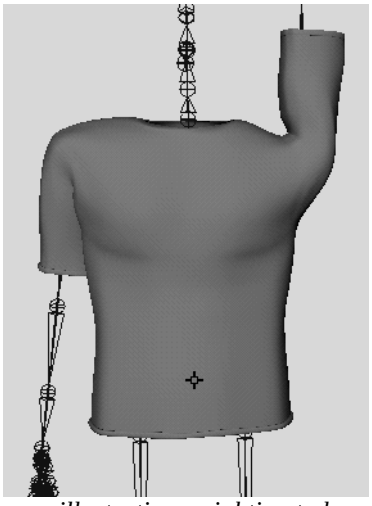
When you were weighting the shorts, every surface was hidden except for the shorts. Now, you are going to do the same thing except you will be working on the shirt.

- Select the *shorts, arms, hands, legs*, and *shoes*.
- Select **Display** → **Hide** → **Hide Selection.**

3 Repose Melvin

- Pose Melvin so one shoulder is up and one shoulder is down so you can see the differences in the two arms.

4



One arm up illustrating weighting to be adjusted

5 Adjust weights around the sleeve using Paint Weights

- Select all of the surfaces that make up the **shirt**.
- Select **Skin** → **Edit Smooth Skin** → **Paint Weights Tool** ❐

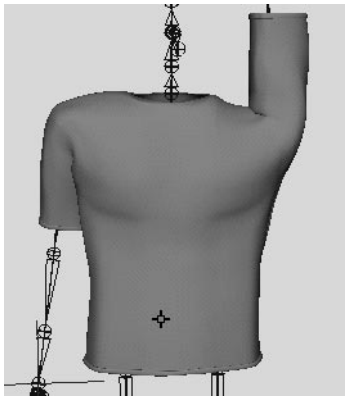
  With the default smooth bind, some of the back, neck, collar and elbow joints may hold some weight on the sleeve. To correct this, select each of these joints as an **influence** and **Replace** the **Value** on the sleeve with **0.00.**

6 Adjust weights for torso region torso

When Melvin lifts his arm, it would be nice if his whole shirt lifted up a bit instead of just the very top of the shirt. To do this, you are going to replace some of the weights on the torso and smooth them out so it looks natural.

- Set **Operation** to **Replace** and **Value** to **0.122.**
- Select **left_shoulder** as the **Influence** and paint the new values along the side of the torso.
- Smooth out the new values until you get a look that you are happy with.

Tip: Using **Scale** as the **Operation** can be useful if you don't get the desired results with **Smooth**. When **Scale** is selected the value in **Paint Mult** is multiplied by the current weight value under the stamp to produce the final weighted result.



The shoulder with adjusted weighting producing even deformation

7 Repose Melvin and test

Put Melvin in a different pose to test the results of the weighting. Adjust the weighting with Artisan as you see necessary. Paint weights as need if problems come up with different poses.

8 Smooth the other side

Repeat the above steps for the other side of the body.

9 Save your Work

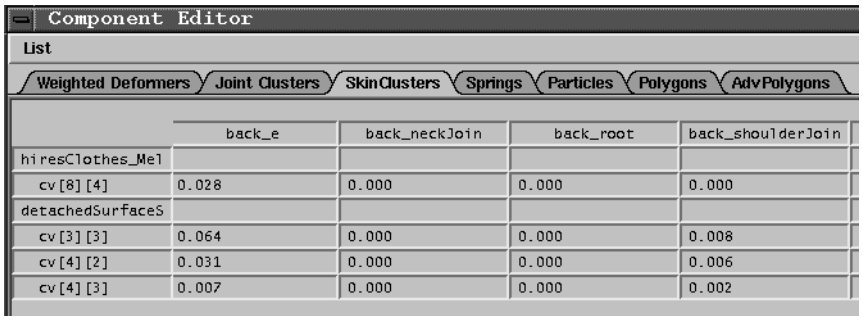
PAINTING WEIGHTS TIPS

Although the **Smooth Bind** function and the **Paint Skin Point Weights Tool** simplify the process used for weighting a character, you may still encounter some pitfalls depending on the character you are working with. The following section provides some general tips and guidelines for making the smooth skinnnig process more efficient and also summarizes some of the key points of the workflow you've just completed.

Check For Weighting From Other Influences

Each skin point has a total weight value of **1.0**, but that weight can be spread across many influences. If a group of skin points aren't behaving the way that you want them to, it is possible that they are getting weights from different (and perhaps unwanted) influences.

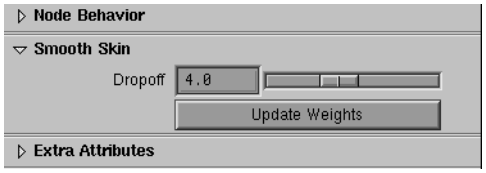
To check or modify the assignments of weights of each skin point, pick a CV then select **Window → General Editors → Component Editor.**



Adjusting weighting using the Component Editor

Adjust the Dropoff Rate

When you originally smooth bound the skin, you set the **Dropoff Rate** for each of the influences. The dropoff rate determines how much the weighting decreases as the distance between the influence and the skin point increases. Increasing the drop off rate helps localize the weighting for the selected joint. To adjust the **Dropoff Rate** after skinning, select the *trasform node* of the desired joint, and adjust **Dropoff** in the **Smooth Skin** section of the Attribute Editor.



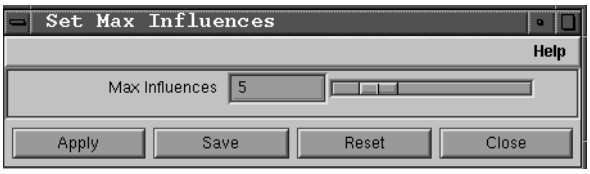
Adjusting the weighting dropoff rate

Adjust the Max Influences

You have an option to set the number of **max influences** on *each surface* that was bound. For Melvin, you set the **Max Influences** to **5**, which means that a total of 5 influences can participate in the weighting on a given skin point. This adds up to a lot of weighting and re-weighting since changing the weighting of one skin point has a "rippling" effect on the weights of the other skin points. As the number of max influences increases, so does this complexity of interdependent weighting.

In many cases it is easier to lower the **Max Influences** of each surface than trying to track down which influence controls which skin point. Lower Max Influence settings will help to localize the control of the weighting. Note that a Max Influence setting of 1 causes the surface skinning to behave like a rigid skinned setup.

To change the Max Influence setting, pick the surface(s) to adjust and select **Skin** → **Edit Smooth Skin** → **Set Max Influences...**



Adjusting Max Influences.

Equalize Weighting Between Multiple Surfaces

If the tangency between 2 patches is giving you problems, it is often easiest to set the same weighting value on the two surfaces to get a uniform weight across the seam then smooth out the weighting between the two surfaces. This technique is helpful because all of the values are set to a uniform state before the smoothing process begins.

Flood Values Across Surfaces

As you have seen, depending on the number of **Max Influences** set when the original **Smooth Bind** function is applied, there can be many joints affecting the same skin point. At times, it is easiest to select the surfaces and an influence and **Replace** all of those weighting values with a common value using the **Flood** button.

This is particularly useful for removing unwanted weighting applied to the root joint, or other joints that should not have any influence on the surface.

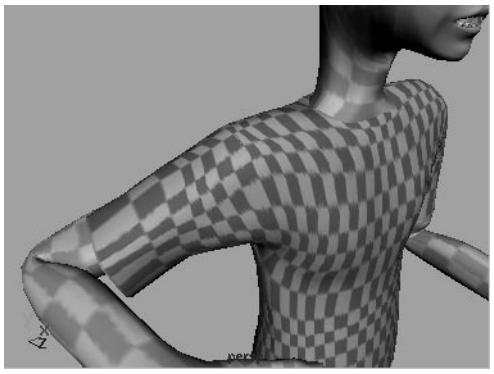
CONCLUSION

You should now have a good understanding of how to use smooth skinning to bind geometry to a skeleton and to edit the resulting skin weighting. Smooth skinning allows skin point weighting to be influenced by multiple joints. These weights can be edited in the Component Editor or painted using Artisan tools even if the surface contains multiple patches.

Maya also has a rigid skinning method which will be discussed later.

9 INFLUENCE OBJECTS

In this lesson, **influence objects** will be examined to aid the deformations of the smooth bound geometry. Their transforms can be used to manipulate the position of skin points to either smooth out deformations or to add effects to the skin. For this lesson, you are going to add influence objects to replicate an elbow, a bicep, and to correct deformations around the shoulder.



Lattice bound to the bones

In this lesson, you will learn the following:

- How to create influence objects.
- How to automate deformations with influence objects and SDK.
- How to weight influence objects and the surrounding skin to provide realistic deformations.

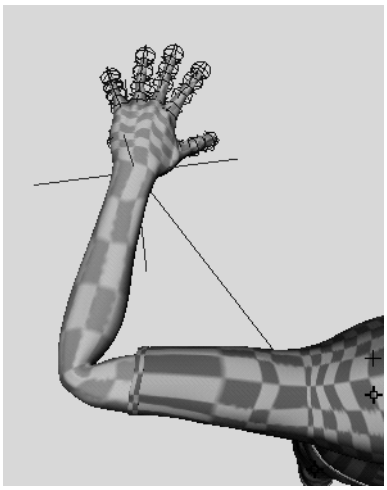
Lesson 9

INFLUENCE OBJECTS

Influence objects deform smooth bound skin by manipulating the transformations of the skin points. These objects can be animated to move as a muscle would deform, or they can be parented to a joint to act like a rigid bone. Influence objects can also be placed within the geometry to smooth out deformations and to maintain volume throughout the character.

Adding an Elbow

The default smooth bind gives very good deformations in parts of the character. However, it doesn't do a good job with replicating bones such as knees and elbows. For this exercise, you are going to add an influence object to create an elbow for Melvin.



Default elbow bend without any influence objects

1 Open file

- Open the *Melvin_09_smoothed.mb.*

2 Return to Bind Pose

A skeleton must be at bind pose for the influence objects to work. If a character is not in the bind pose, Maya will not create the influence object.

- Select **Modify** → **Disable Nodes** → **All.**
- Select **Skin** → **Go to Bind Pose.**

Tip: If you made a bind pose button with the Record Pose script, it will be easier to use that instead of disabling the nodes and returning to bind pose through the menus.

---

3 Use a polygonal sphere as elbow influence object

For Melvin's elbow, you are going to create an underlying object that will be deformed with the rest of the arm. This elbow will be created

from a polygonal sphere. You are going to use polygons instead of NURBS to make it easier to differentiate between *skinned surfaces* and *influence objects*.

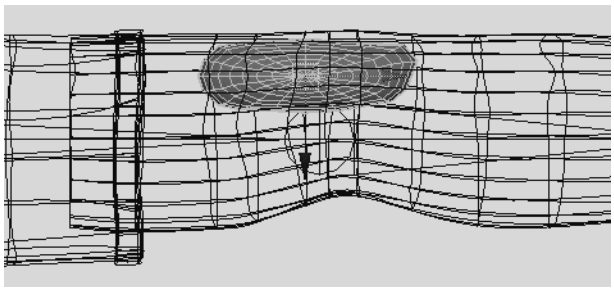
- Select **Create** → **Polygon Primitives** → **Sphere**.
- Rename the sphere *elbow_Influence.*

4 Move the sphere Intoposition

- **Translate** *elbow_Influence* so it is in the same location as *left_elbow* joint

5 Deform the sphere so it Resembles a Bone

- **Scale** the *elbow_Influence* so it resembles the shape of a bone.
- You may need to tweak individual vertices to get the look you are after. Try to shape and position the object so it fits between the *left_elbow* joint and the back edge of the *left_arm* geometry.



elbow_Influence positioned and deformed

6 Select appropriate geometries and create Influence

In order to create influence objects, you must select your objects in the proper order. The first selected is what is going to be influenced and the last selected is the influence object itself.

- Select *left_Arm* geometry
- **Shift-select** *elbow_Influence.*
- Select **Skin** → **Edit Smooth Skin** → **Add Influence.**

7 Parent ebow Influence to appropriate joint

You are going to parent the influence object to a joint so the influence object moves with the rest of the skeleton.

- Select *elbow_Influence* and **Shift-select** the *left_shoulder* joint. in the persp window.
- Select **Edit** → **Parent.**

  When you created the influence object, Maya created a **base** object and hid it from view. you will also parent this base object to the shoulder bone. The **base** object stores the component information of the influence object. Without the base object, you would not be able to manipulate the components of the influence object.
- Select *elbow_InfluenceBase* from the Outliner or the Hypergraph.

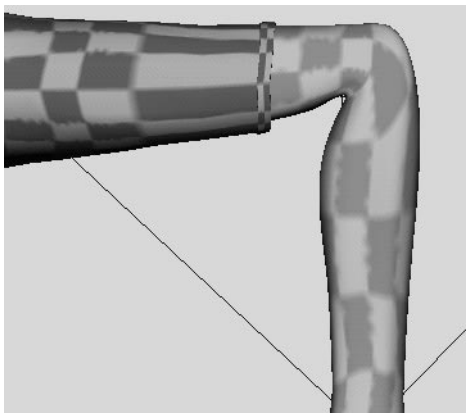
Disabling Nodes

Whenever IK or joints are driven by constraints or expressions the **Go To Bind Pose** button will not immediately work. To make it work, you can select **Modify** → **Disable All Nodes**. This toggles off the control of any nodes that may prevent you from reaching the *Bind Pose*. Don't forget to **Enable All Nodes** to animate with the controls you've created.

- Shift select the *left_shoulder*
- Select **Edit** → **Parent.**

8 Test the Results

- Rotate the elbow to see the deformations around the elbow



Improved elbow deformation after applying influence object

Note: Don't worry if the influence object is sticking through your geometry. When you create a influence object, Maya automatically turns **off** the **Primary Visibility** in the object's Render Stats. The influence object will not render unless you manually turn **on** the **Primary Visibility** in the Attribute Editor.

---

9 Save Your Work

- Select **File** → **Save Scene**

Add a Bicep Muscle Bulge

Melvin was bound with the smooth bind function and although it creates good deformations around most of the body, it won't allow you to create flexors the way that a rigid bound skeleton will. To create the bulge in the bicep when Melvin bends his elbow, you are going to create an influence object and animate it with Set Driven Key.

Because there are many steps that are the same when you create influence objects, many of these steps will be familiar to you.

1 Open file

(Or you can also continue working on the same file as before.)

- Open *Melvin_10_elbowInfluence.mb.*

2 Return to Bind Pose

Just like when you added the elbow influence object, you need to be in bind pose to add an influence object.

- Select **Modify** → **Disable Nodes** → **All.**
- Select **Skin** → **Go to Bind Pose.**

3 Create polygonal sphere to act as an Influence

- Select **Create** → **Polygon Primitives** → **Sphere**.
- Rename the sphere *bicep_influence.*

4 Position and deform the sphere

- **Translate** the *bicep_influence* to where the bicep muscle would be on the arm. Make sure that it is close to the front of the *left_Arm* surface.
- **Scale** the sphere so it looks roughly like a bulged bicep muscle. You might need to adjust some vertices in component mode.

5 Select the surfaces and create Influence

For the bicep deformation to look correct, the deformations must take place on the shirt as well. Influence objects can effect more than one surface as long as they were created with all of those surfaces selected.

- **Select** the *sleeve* surfaces and the *left_Arm* geometry.
- **Shift-select** *bicep_influence.*
- Select **Skin** → **Edit Smooth Skin** → **Add Influence -** ❐.
- Press **Reset** then **Add.**

6 Parent bicep_influence to left_Shoulder

You will now parent the influence object to a joint so it moves with the rest of the arm joints.

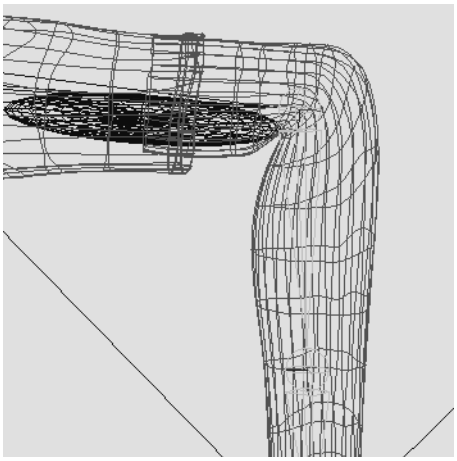
- Select *bicep_influence* then **Shift-select** *left_Shoulder* joint.
- Select **Edit** → **Parent**.

Also parent the influence object's **base** to the shoulder.

- Select *elbow_InfluenceBase* from the Outliner or the Hypergraph.
- **Shift-select** the *left_shoulder*
- Select **Edit** → **Parent.**

7 Rotate the elbow to test deformation

To check out the deformations of the influence object, rotate the elbow joint. Most likely, the deformations are not obviously noticeable. To make the bicep bulge when the elbow bends, you are going to set up a Set Driven Key relationship between the influence object and the elbow rotation.



8 Straighten the Elbow Rotation

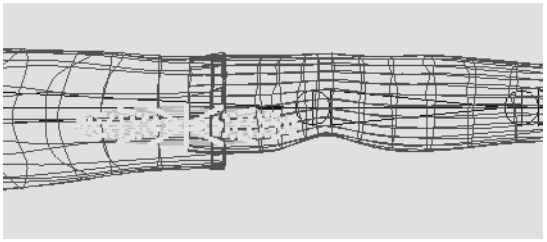
Because you are going to set up a **Set Driven Key** relationship, it is easiest to start with the elbow straightened out.

- Select the *left_elbow* joint.
- Set the **Rotate Y** value to **0.**

9 Select the Vertices

There is an attribute in each of the influence object's channel box that allows you to select between using the object's transform node or its components as the driving force to create the deformations. You are going to be using its component information to get the desired deformations from the bicep.

- **Select** the influence object.
- **Press F8** to switch to **component** mode.
- Select all of the vertices on the influence object.

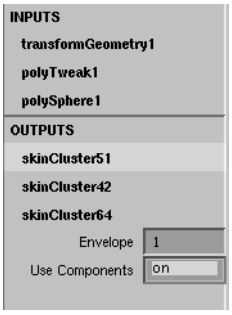


Selecting all the bicep vertices

10 Turn on Use Components

In the Channel Box of the influence object, a list of **Outputs** are connected to the influence object. Each one of those skinCluster outputs has an attribute called **Use Components** that needs to be switched to **on** for Maya to evaluate the influence's vertices rather than its transform.

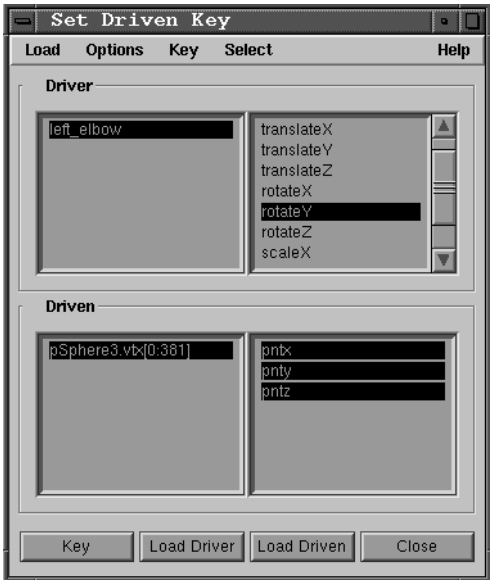
- Select each **skinCluster** Output.
- Switch the **Use Components** attribute to **on** for each output.



Enabling Use Components in the Channel Box

11 Prepare Set Driven Key

- Select **Animate** → **Set Driven Key** → **Set -** ❐.
- Load the selected *vertices* as the **Driven.**
- Load the *left_elbow* as the **Driver.**



Set Driven Key window:

12 Set a Driven Key for initial position

In order to set a driven key, you will need to set the desired attributes on the **Driver** and the **Driven.**

- Select the *left_elbow.rotateY* as the **Driver** attribute.
- Select the *vertex.pntx*, *vertex.pnty*, and *vertex.pntz* as the **Driven** attributes.
- Press **Key.**

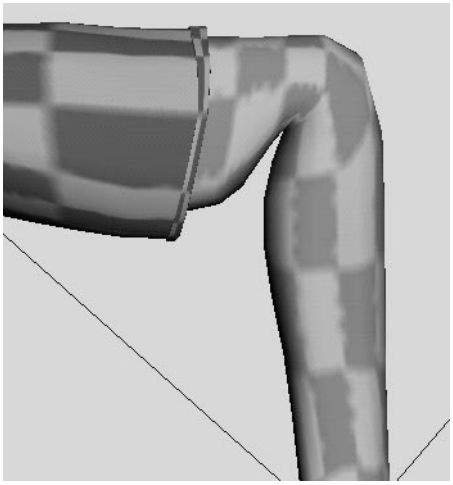
Note: Depending on how you selected the vertices, you may see a list of every vertex on the influence object. If you see this, just select them all and then select all of the attributes on the right side of the window.

13 Rotate the elbow

The best way to see the deformations of the influence object is to rotate the elbow so it is bent.

14 Deform the Influence Object

- Translate the vertices until you see a bicep deformation that you are happy with. The arm and shirt surfaces should update each time you move a vertex.



Rotating the elbow and positioning the bulge

15 Add a Set Driven Key for the rotated position

- Press **Key** in the **Set Driven Key** window to complete the relationship between the joint rotation and the influence object.

16 Test the Results

Rotate the elbow and test the deformations of the bicep. The bicep should now grow as the elbow rotates.

Add an Influence Object for the Shoulder

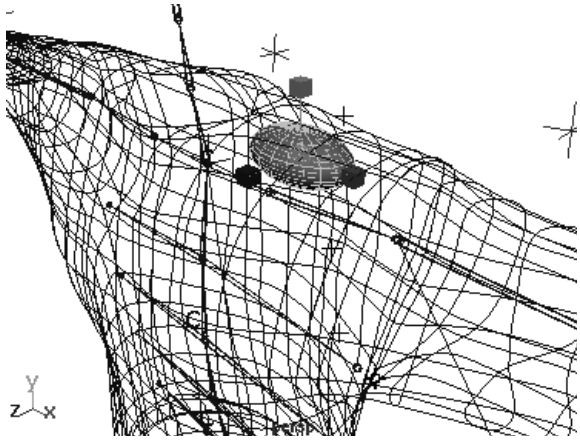
You probably could have just painted the weights of the shoulder to get the deformation that you wanted. However, adding influence objects to the shoulder will help maintain the volume in the shoulder area and is a quicker solution to getting good shoulder deformation.

1 Return to bind pose

- Select the *back_root* joint and select **Skin** → **Go to Bind Pose.**

2 Create an influence geometry for the shoulder

- Model a shoulder from a polygon sphere.
- **Rename** the sphere *left_Shoulder_Influence.*
- **Move** the *left_Shoulder_Influence* so it rests just to where you want Melvin's shoulder to be.
- Parent the *left_Shoulder_Influence* to the *left_collar* joint.



Positioning a modified sphere as a shoulder influence object

3 Create an influence object

- Select the following 6 surfaces around the shoulder:

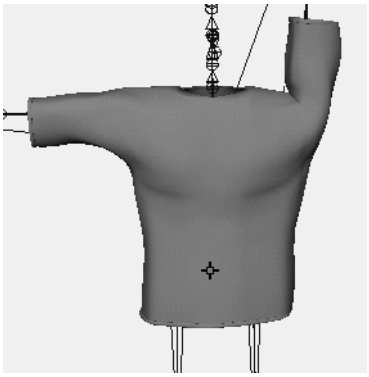
  shirt_left_middle, shirt_left_middle_back, shirt_left_top, shirt_left_top_back, shirt_left_sleeve_front, shirt_left_sleeve_back

- **Shift-select** the *left_Shoulder_Influence.*

4 Paint the weights

Influence objects can be weighted just the same as regular bound joints. Use the **Paint Weights Tool** to do this.

- **Select** the shirt surfaces.
- **Shift-select** the *left_Shoulder_Influence* sphere.
- Open the **Paint Skin Weights Tool**.
- **Select** *left_Shoulder_Influence* as the influence to be painted on.
- Paint around the shoulder to smooth the weights out.



Smoothed shoulder

5 Test the results

Place Melvin in different poses to test the deformations around the shoulder. You might need to repaint the weights again if the shoulder does not deform properly in other positions.

6 Save your work

Summary

In this lesson, you have learned how to:

- Add influence objects to smooth bound skin
- Manipulate skin points with influence objects and SDK
- Adjust weighting on influence objects to smooth out deformations

10 Kick the can

In this scene, Melvin will walk up to a discarded tin can and kick it. This exercise makes use of several tools and techniques that are common to character animation such as walking, interaction with another object, anticipation, and follow through.

Footage of kicking the can

In this lesson, you will learn the following:

- How to use a motion study as a guideline for the animation.
- How to use Image Planes.
- How to work with the Dope Sheet.
- How to work with the Graph Editor.
- How to Playblast your animation.

Lesson 10

Animation controls

You will begin by looking at how your character has been built up to this point. You have completed binding Melvin's skin and have several controls for animating him:

- left and right wrist locators (arm control, rotate hand and forearm, finger controls).
- left and right ankle locators (leg control, rotate ankle).
- back_root (translate and rotate the hips).
- back_clusters (fine-tune back motion).
- pole vector constraints for elbows.

By selecting the two wrist locators, the two ankle locators, you can move the entire character to different locations in the scene.

Animation workflow and tools

With this lesson, experiment with several techniques. Try using the following suggestions that *could* be used when animating.

This is an outline of a suggested animation workflow that you can follow—or you can take this any direction you choose. Whichever workflow you use, there are several helpful animation tools and techniques that you might want to keep in mind.

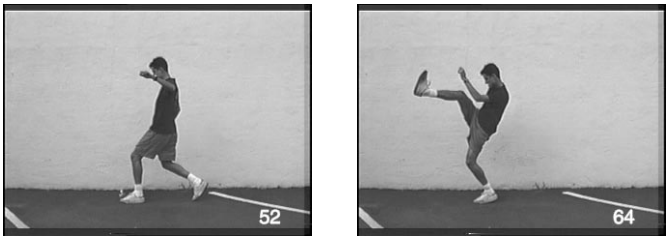
Storyboarding

The story board is where you hope to find as many problem areas and special requirements as necessary. Here is where you note and plan for timing issues that may occur.

Motion study

Once you have completed the story board to asses the basic timing and actions of the character, you need to see how a person kicks a can.

"When in doubt go to the motion study." has been said by many of the professionals at leading production companies. There is no substitute for learning character animation from real live examples. Included is a motion study of a person approaching, then kicking a can.



Motion study footage

IMAGE PLANES

You can use the digitized video as flipbooks and also bring them into Maya as image planes. *Fcheck* serves as a great method for quickly viewing the reference motion. While not as fast, image planes work really well as a frame-by-frame placement guide.

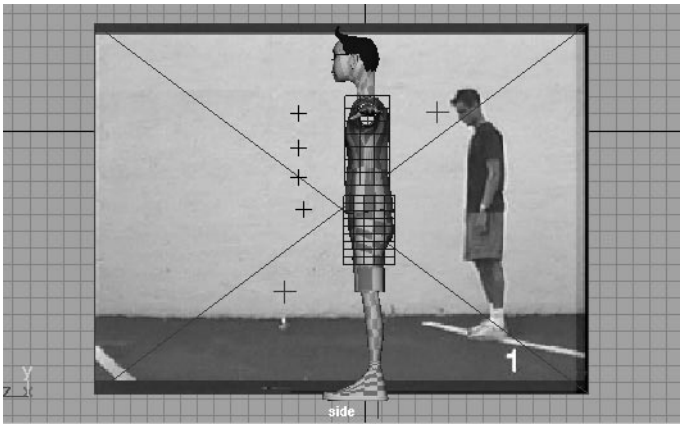


Image plane added to the side window

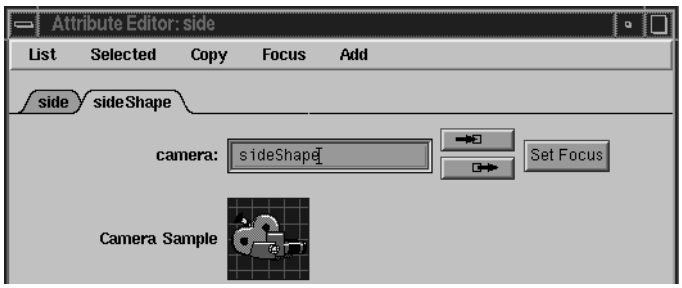
Because the motion tests are filmed from the side, you must set up the image plane for the side camera. This is the basic workflow for getting the image planes to work:

1 Open an existing file

- Open the file *Melvin_10_noImagePlane.mb*

2 Add an Image plane to the side camera

- Select side camera.
- In the Attribute Editor, click the **Sideshape** tab.



The sideShape tab

- Under the **Environment** folder, click **Create**. This will create an image plane.

The Environment folder

Using fcheck

You can use **fcheck** to display a single image or a series of images from a unix shell.

To see this motion test with fcheck:

- open a unix shell
- `cd ~/maya10/projects/Melvin/motiontests/`
- `fcheck Melvin9kick`.

To learn more about fcheck type `fcheck -h` in a shell.

- Under the **Image Plane** attributes, click **Browse** to read in the digitized video sequence into **Image Name**.

Note: If the image has no alpha, you should change the **Display Mode** to **RGB**.

- Toggle **Fixed** for Image Plane. If you zoom or pan the side view, the image plane will still be the same size relative to your scene.
- Under **Placement Extras**, adjust the following:

  Width and **Height** to scale the image plane to the size of the current scene.

  Center XYZ to position the image plane

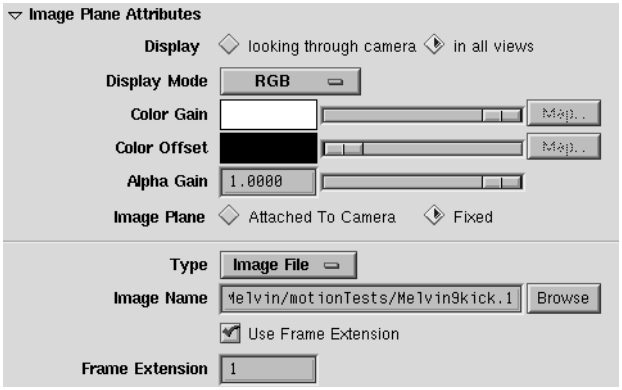


Image plane attributes

BLOCKING

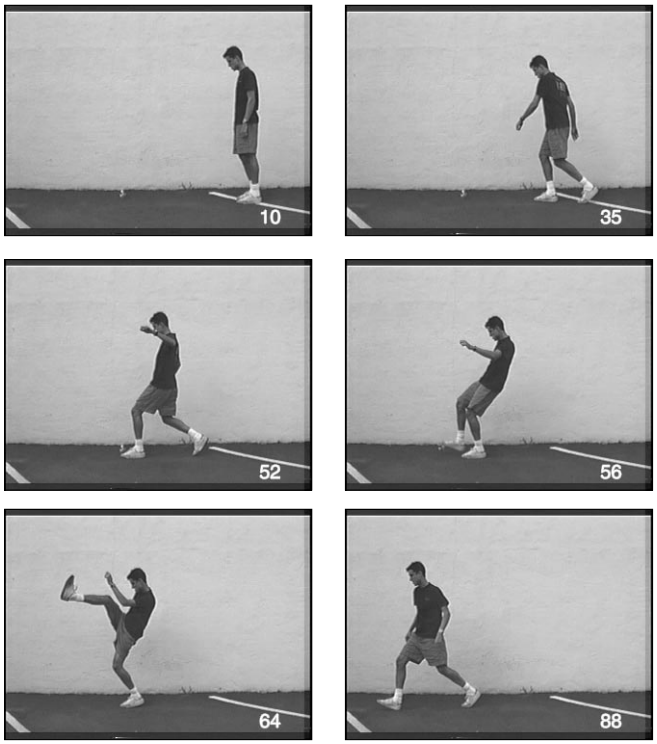
Before setting keys for the detailed motion, you'll *block* the shot. Blocking a shot consists of setting key poses every 5-10 frames to rough out the animation. Working with character sets is a good way to set general keyframes on all of the attributes in your character set.

For scenes where the motion is not repetitive, it is important to study the extreme positions the character gets into. For the kick, look for the back swing, the plant, and the follow through of the kick. These are the poses that really define the feel of the animation.

After blocking, you'll review this generalized motion asking the following questions:

- Does the motion and timing work in this scene?
- Is the motion too fast/slow?
- Is there continuity with other shots?

You don't want to worry about the details of the motion until you are comfortable with the generalized motion.



Main Keyframes

- Frame 10 -Just starts to move from passive position
- Frame 35 -First step
- Frame 52 -Plants feet for kick; notice arm position
- Frame 56 -Contact with the can; note weight distribution
- Frame 64 -Follow through; notice the extension
- Frame 88 -Landing after the kick; hips rotate

Note: This is only a guide of where the extreme poses could be. Take a look at the motion tests and set up a list of the poses you wish to block.

IN-BETWEENS AND BREAKDOWNS

Once you have finalized the blocked motion, it's time to start rounding out the motions. The *in-between* is responsible for creating the interpolations from one blocked key to the next. It shapes the motion away from the linear point to point motion you have established.

In-betweens can occur every 3 to 5 frames or as needed. When your character is moving really fast (during the kick), the in-betweens could occur on every frame. At this stage, you are not concerned with perfect motion. The resulting motion will look better than the blocked motion, but

it will still need some fine-tuning. Study the video, you may be surprised where and when these keys occur.

Consider using breakdown keys for your in-between keyframes. Breakdown keys are designed to be placed between blocked poses so they can later be moved in the timeline to maintain the relationship between the standard keyframes. Although adjusting overall timing may not be used as much if you are working straight from a motion test, it is still a good idea to get in the habit using breakdown keys. It will also be useful if you decide to change the timing of your animation later on.

Also, consider using sub-characters for your inbetween poses so you don't key all of the attributes in the entire character set. A main character set can be created to block out the animation, while sub-characters can be used to key specific parts of the character. For example, each leg can be its own sub-character that controls all of the motion for that leg and foot.

Try to avoid keying all of the attributes on an object like you did when you blocked out the motion for the animation. In some places, you may still want to set a key on all the keyable attributes. However, in other more focused places you will want to key only the selected locator or selected joint / cluster via the selection handle. Use the RMB in the Channel Box to key individual attributes by selecting **Key Selected** or **Breakdown Selected.** This will result in liner curves in the Graph Editor. The fewer keys you set in this phase the easier it is to make major changes later.

After you have completed a cursory in-between, save this file as your rough in-between. If you need to make major changes to the animation, this is where you will most likely start**.**

After the in-betweens are finished you will go back through and address the rough edges and start working on the details that make the animation interesting.

Adding in-betweens

The motion study is the chief guide for adding in-betweens. Some of the main movements that may escape the casual observer have been pointed out. This is a largely self-guided exercise based on motion study and your creative interpretation. Decide for yourself whether to use standard keyframes or breakdown keys.

1 Create a Character Set for Melvin

Create a character set that has all of Melvin's controls (locators and selection handles) in it. This character set will make it easier to block out the animation.

2 Key the Character in the rest position at frame 1

Confirm he is posed correctly on top of the image plane.

- At frame 1, set a **key** for the character set.

3 Create Melvin's first step

The character takes his initial step by first falling forward and then brings his foot forward to catch himself. Determine the mid-stride

position and keyframe that pose. Use the motion study to determine the frame. Work in the following sequence:

- Move *back_root*.
- Move the feet.
- Move the hands then key.

4 Position the stepping stone to where the first footstep lands

Carefully align Melvin's heel and Key the *passing pose*, again based on the motion study.

- Go through each locator and major body position.
- Note twist and turning of upper and lower body.
- The character's head is down focused on the can.

5 Key the next step

- Key the next mid-stride pose.
- Key the passing pose**.**

6 Key the planting of the foot in preparation for the kick

Note the lean of the body and the reach of the arms for balance.

- Use the same method of defining the mid-stride and the passing pose.

7 Kick the can

The kick's main movement besides the leg motion is the rotation that takes place in the upper and lower body. The mechanics of the rotation and balance influence the arm swing which is also pronounced.

8 Create a playblast of the animation

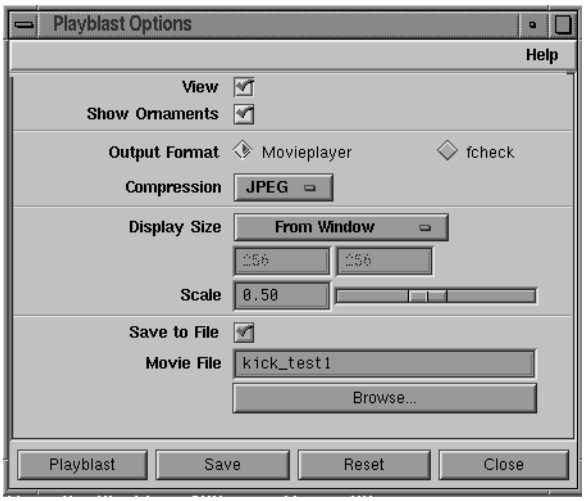
- Position the perspective camera in an advantageous position to see all the action.
- Select **Window** → **Playblast...** - ❐.

  View the playback, scrub through and note areas that need work.

Playblast

Window → **Playblast...** is a way to evaluate your animation quickly. It is a very fast screen grab of your animation. It will capture the animation as it is currently displayed, so you can see it in wire mode or shaded mode (which may take a little longer to calculate). The purpose of this tool is to get real time playback by using a compressed file.

At each stage of creating the animation, you should use Maya's Playblast function to create motion tests to evaluate your animation in real time.



Playblast Option Window

FINE-TUNING THE MOTION

Once you are comfortable with the in-betweens, you can start to fine tune the motion. This is the stage that never ends. This is where you can find yourself tweaking and adding keys on every frame. You want to avoid doing that as much as possible. Here are some things to keep in mind that will hopefully keep you on target.

Work on major keys and in-betweens first, then secondary and tertiary keys next, working in layers of refinement.

Get your main keyframes looking as good as possible first, then break down into the next layer of in-betweens. Once this layer looks good then go to the next layer. You will find you have intimate knowledge of these *milestone keys* instead of having keys scattered all over the timeline.

Adjust the animation curves in the Graph Editor

In the Graph Editor, you can get a lot of mileage out of a key by working with the tangency or method of interpolation. *Be sure to not drag keys off the frame* or create keys that do not fall on a whole frame. If you render on whole frames, any key that is not on the frame will be missed. The motion will be close but the key may not be as precise as you would like.

Remove superfluous keys

Remove keys that don't seem to be contributing or were made ineffective. This is best done in the Graph Editor where you can see the direct result on the curve by removing the key. If you make a mistake, simply undo the removal.

Test to Playblast

Test in the work area and to Playblast. It is often a good idea to take a break while you build a movie. Come back to the computer a little fresher to view the movie and plan the changes you will make.

Add subtle motions to major and minor joints and control points

You will often find that after the basic in-betweens are completed, it is time to look at parts of the character you have not keyed at all. The hands and head are very important, as are the shoulders and hip joints that will contribute to the motion of the attached joints. Rotations and translations in all dimensions are what make the subtleties of realistic movement.

Offset the motion of joints to achieve secondary motion

Offsetting is the act of delaying a joint's motion in relation to the surrounding joints. This is often seen as a *breaking* movement. When an arm, for example, moves toward an object that it wants to pick up, it does not move in unison at once toward its target. Rather, it will break at the main joint (elbow) first, then at the wrist, then the fingers.

Consider another example, the hand. When you make a fist, all of your fingers do not close at once. Some fingers may begin to close ahead of others while some may start late but finish first. These subtle movements and accelerations are at the heart of realistic motion.

Use the Dope Sheet Editor

The Dope Sheet Editor is a good place to view and offset the timing of keyframed motion when you only want to affect the timing and not the magnitude of the motion.

OPTIMIZATION

There are several options that can optimize feedback when setting up the animation of a character.

Display optimization

Geometry can be viewed at many levels of accuracy. By selecting the geometry and then pressing **1**, **2**, or **3** on the keyboard you can select between coarse, medium and fine display accuracy. This will not affect how the geometry is rendered only how it will display. There are several options under the Display menu that affect the performance of Maya's display. An overview follows:

NURBS smoothness

Under the **Display** menu you will find a sub-menu for **NURBS Smoothness.** These options control how Nurbs surfaces are displayed.

- **Display** → **NURBS Smoothness** → **Hull** displays the selected geometry in the crudest form. From the option box you can adjust the coarseness of the hull display. By default this display type is not mapped to a keyboard key but **Ctrl 1** is as good as any if you are setting your own hot keys.
- **Display** → **NURBS Smoothness** → **Rough, Medium, and Fine** are as you would expect and are selected by pressing the **1**, **2** or **3** keys on the keyboard. The option box for each of these allows you to decide whether you want this mode to affect the selected object or all objects.

- **Display** → **NURBS Smoothness** → **Custom** is a user-defined setting for display smoothness. There are many customizable settings in the options box allowing for almost infinite combinations to suit your needs.

Fast interaction

Display → **Fast Interaction** enables the user to interact with the scene more quickly by temporarily changing the resolution of the geometry while the scene is being manipulated, then switching back to the higher resolution after the scene has settled. This setting will also improve playback of animation in the timeline, but be prepared for some degraded looking geometry.

Animation preferences

In the **Options** → **General Preferences** window **Animation** section is a few settings that will influence the way Maya plays back your animation. In the **Playback** section you have options to change:

Update View

Update the Active panel or All panels

Looping

Determines the Looping method

Playback Speed

Free – Maya will play every frame regardless of frame rate settings

Normal – This setting forces Maya to play back at the frame rate that is set in the **Options** → **General Preferences Units** section. Video frame rate is 30fps and 24fps is for film.

Half /Twice – Maya plays back at half or twice the specified frame rate.

Other – Maya will play back at a user defined percentage of the specified frame rate

Performance options

None – Changes made in the Animation Editors are not reflected in the scene until the Transport Controls are used or the Current Time indicator is moved.

Delayed – Changes made in the Animation Editors are not reflected in the scene until the mouse action is completed by releasing the button.

Interactive – This is the default setting. Scenes update as changes are made to the keys and curves in the Editors.

Summary

You have now completed the following tasks:

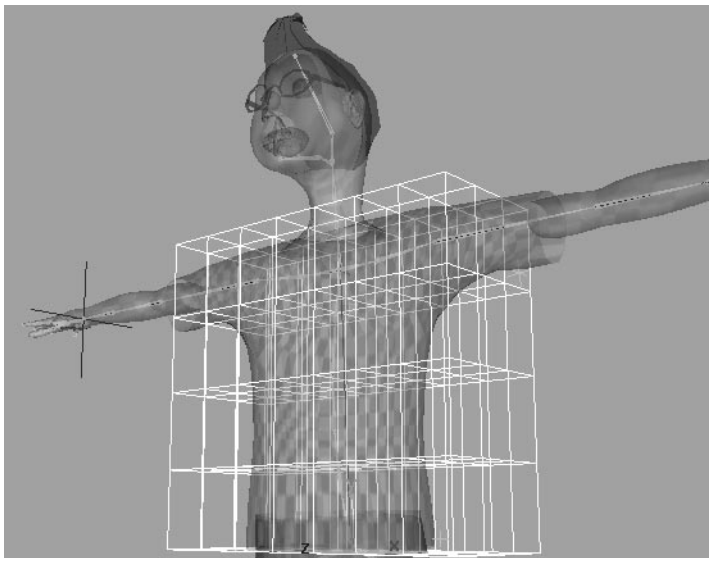
- Used image planes with sequenced images as guides
- Used Playblast for evaluating your animation
- Tuned your animation using various techniques

- Reviewed some topics for improving interactive performance

Lesson 11

11 Rigid Skinning

In this lesson you will explore rigid skinning techniques. While smooth skinning binds CVs or deformer components to multiple joints and influence objects, rigid skinning binds these components to only one joint. The components are then weighted to provide smooth interaction. The use of flexors and other layers of deformers then work to control the skin deformation. You will use a number of techniques in combination for skinning Melvin's skin to the skeletons.



Lattice bound to the bones

In this lesson, you will learn the following:

- Skinning surfaces to bones
- Binding surfaces through lattices to the bones
- Combining direct and indirect skinning
- Creating partitions and skin set organization

Before skinning Melvin, you will experiment on some practice joints. You will run through some short examples in binding skin geometry to joints in preparation for Melvin.

In this lesson you will look at bind skin and lattices. Lattices offer a great way to get smooth, generalized skinning around areas that can be more tedious if skinned directly such as the shoulders or the area where the legs meet.

BIND SKIN

Bind Skin is probably the most common technique of binding geometry to skeletal joints. With Bind Skin, the geometry is divided into clusters based on proximity to the nearest joint. As the joints in the skeleton move, the clusters are in turn transformed.

A cluster can be thought of as a "smart" set. The vertices of the surface are put into a set where they can be weighted, or receive percentage effects from the cluster node. Unlike a set, a cluster is also pickable in the modeling windows. Weighting and membership will be discussed in a following chapter.

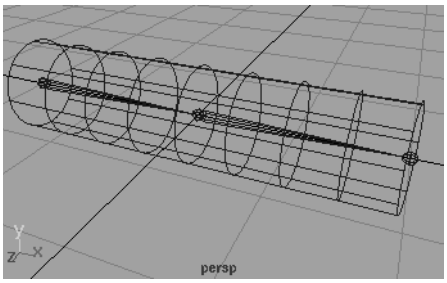
Bind Skin example

1 Create a cylinder to act as a skin

- Select **Create** → **NURBS Primitives** → **Cylinder**.
- **Scale** and **Rotate** the cylinder to a horizontal position.
- In the Channel Box, set the **Spans** and **Sections** to **8**.

2 Draw skeleton joints

- Select **Skeleton** → **Joint Tool** and create **3** joints to form an arm with an elbow.

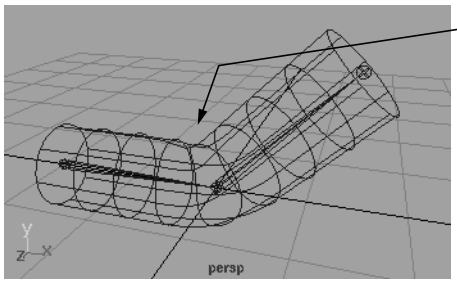


Cylinder and joints

3 Bind the cylinder to the joints

- **Select** the skeleton, then **Shift select** the geometry.
- Select **Skin** → **Bind Skin** → **Rigid Bind.**

4 Test the movement by rotating the middle joint



The Bind Pose

Once a skin is bound to a skeleton, Maya remembers the pose that the character was in when it was bound. This is useful when you are testing poses and want to return to the starting pose. To get back to the bind pose, select the *skeleton* then select **Skin → Go To Bind Pose.**

The Bind Pose is only stored for a joint if a cluster is created for it, so it is good practice to bind the entire character at one time.

Detaching Skin

If you have bo und skin that you no longer want to be bound, you can detach the skin. Do this by selecting the geometry to detach, the skeleton and **Skin → Detach Skin**.

BIND SKIN WITH A LATTICE

A lattice is another type of deformer that works in a more general manner. A lattice is a bounding box with points that will deform the geometry sitting inside the lattice. The benefit of using a lattice is that you are dealing with a lesser number of control points.

Binding a lattice example

In this exercise, you'll take the method from above a step further. For this technique you'll create a lattice deformer for the cylinder and bind the lattice to the skeleton.

1 Create a cylinder to act as a skin

- Select **Create → NURBS Primitives → Cylinder**.
- **Scale** and **Rotate** the cylinder to a horizontal position.
- In the Channel Box, set the **Spans** and **Sections** to **8**.

2 Apply a lattice to the cylinder skin

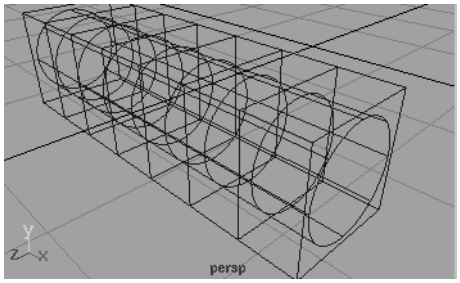
- Select the cylinder.
- Select **Deform → Create Lattice -** ❐ and set the following:

  STU divisions for the lattice to **8**, **2**, and **2**.

Disabling Nodes

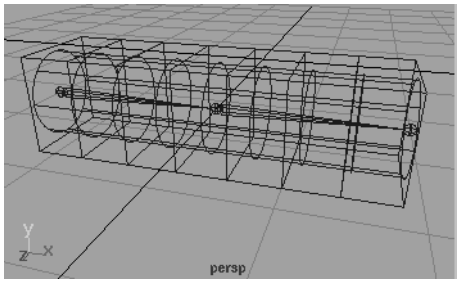
Whenever IK or joints are driven by constraints or expressions the **Go To Bind Pose** button will not immediately work. To make it work, you can select **Modify → Disable Nodes → IKSolvers, Constraints**. toggles off the control of any nodes that may prevent you from reaching the *Bind Pose*. Don't forget to **Enable All Nodes** to animate with the controls you've created.

Note: Translating any of the lattice points in Component mode deforms the cylinder.



Lattice on cylinder

3 Create 3 arm joints inside the cylinder



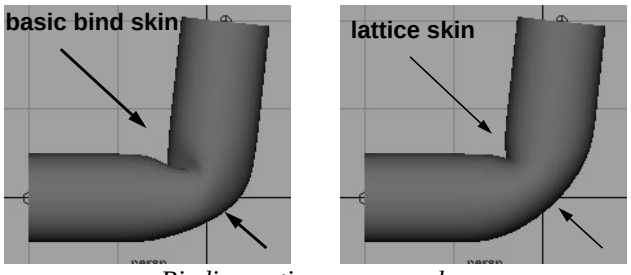
Joints

4 Bind the lattice to the joints

Make sure that the geometry is not selecting when binding the lattice to the joints.

5 Test the movement by rotating the elbow joint

In **Shaded** mode, compare the elbow bend of the basic bind skin with that of the lattice skin that you just created.



Binding options compared

You will see that the skeleton deforms the lattice which deforms the cylinder. This technique produces a more generalized skinning than the basic bind skin which should provide smoother deformations. You'll be using this technique to get a smoother skinning deformation for Melvin's shoulders and where his legs meet.

SETS

A *set* is a collection of objects or components. For example, a set might include geometric objects, NURBS, CVs, polygonal vertices, lattice points, polygonal facets, or other items. Any item you can select can be in a set.

Maya uses sets for all sorts of functions. You'll use them to divide and organize Melvin's various parts (skin, skeleton, lattice, etc.) to make the skinning process easier. Organizing the components into sets for selection is an important method for efficient workflow.

Partitions

You can also use set organization to aid in controlling group membership. By putting your sets into a *partition*, you can ensure that the member components are *exclusive* to each set. That is, you remove the possibility for overlapping membership between surface points, CVs, lattice points or whatever component you are working with. This is especially important when skinning. For example we will divide up Melvin's shirt components for Rigid Skinning with direct (bound to joints) and indirect (bound to lattices which are bound to joints) binding.

- **left sleeve** -shirt sleeve bound to skeleton (direct)
- **right sleeve** -shirt sleeve bound to skeleton (direct)
- **shirt_lattice** -shirt deformed by lattice and lattice bound to skeleton (indirect)

By using partitions, you can ensure that you do not bind parts of the shirt that are in two of these sets at the same time. This would result in double transformation. That is a CV or object component gets moved twice, once by the joint, and once by the lattice.

Note: Dual or multiple membership of geometric components will result in double transformations following Bind Skin.

This dual membership can be corrected after the bind skin operation, but is much easier if everything is organized beforehand. By creating a partition, then creating sets of skinning groups within the partition, you will maintain exclusivity of geometric and deformer components.

Partitions

An example of partitions at work are the layers you use in Maya. Layers are sets that belong to the layer partition. Geometry can only be on one layer. The partition keeps the sets organized so that the geometry is not on more than one layer.

Quick Sets

Quick Sets differ from Sets in that they are used to quickly organize a group of objects into a selection group. The **Edit → Sets → Make Quick Select Sets** function does *not* create sets under a partition and thus they are *not* exclusive.

SKINNING MELVIN

The workflow for skinning Melvin's shirt and shorts consists of combining skinning methods. Below are the key steps to complete this task:

1. Create sets for the selected components
2. Create the lattices for the shirt
3. Import the wrap for the shorts
3. Create the lattice point sets
4. Create a skeleton set
5. Select the sets
6. Bind Skin

Creating a skinSets partition

You need a partition to place your skin sets in. Again, this helps you avoid overlapping set membership and aids with selection.

Tip: Sets can have overlapping membership but **sets in a partition** will not have overlapping membership.

1 Open an existing file

- Open the scene file *Melvin_08_handControls.mb*.

2 Create a partition called skinSets

- Select the **Edit → Sets → Create Partition -** ❐.
- Enter *skinSets* in the **Name** text field and click **Apply**.

3 From the Relationships Editor, display the Partition Editing section

- Select **Window → Relationship Editors→ Partitions...**

  The *skinSets* partition should be listed in the Partition Editor. If you select the folder for "skinSets" there should be no sets or objects high lighted in the right hand side.

Adding geometry to sets

For both Melvin's upper torso (his shirt) and his pelvis region (his shorts), you will create deformers to act as indirect objects for binding skin. Before you do this, you'll create sets of the geometry points that the lattice will deform. This is an important step for working with objects that will be bound with lattice or wrap deformers and straight bind skin. By organizing Melvin's geometry into exclusive sets *before* skinning, you'll save time by eliminating the need to edit the membership after skinning. You could potentially skin up Melvin and avoid double membership, but this can be very tedious and sometimes won't be completely accurate.

To skin the lattice for the torso and the geometry outside the torso, you would need to select:

- The lattice

- The shirt sleeve CVs that are not affected by the lattice

After going through this workflow a few times, you'll appreciate the fact that partitions help with exclusivity and thereby avoid dual membership.

Dividing geometry into sets

You will create sets for the shirt, head, arms, legs, and shorts. The shirt will be divided into two sets:

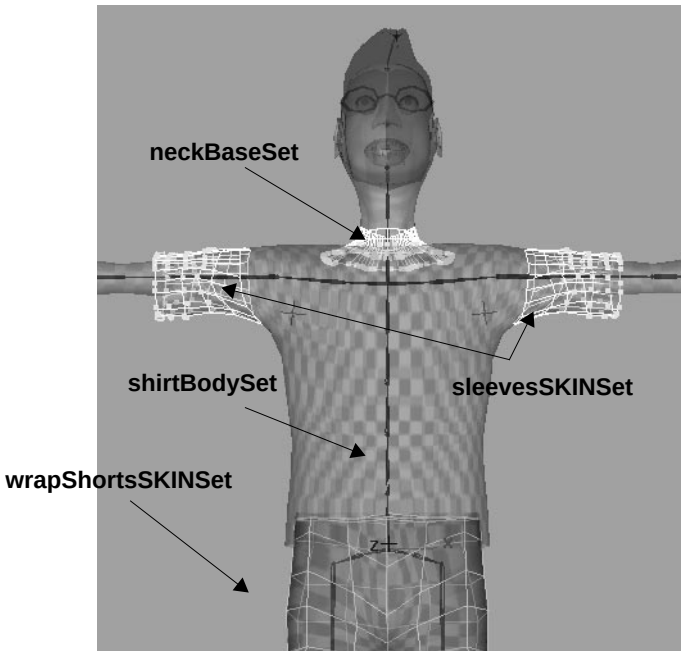
- *shirtBodySet*
- *shirtSleevesSKINSet*

and the rest of Melvin's sets:

- *shortsWrapSKINSet*
- *shirtLatticeSKINSet*
- *legsArmsSKINSet*
- *neckSet*
- *headSKINSet*
- *skeletonSKINSet*

Tip: Notice that some of the names end with "*SKIN*." These will be the sets that you will bind to Melvin's skeleton. The other sets will be deformed by a lattice and you don't want to bind those CVs. Rather it's the lattice that is deforming them that you want to bind.This naming technique makes the selection in the Set Editor much easier.

Melvin's shorts and shirt are modeled and simplified into a few different surfaces (top, middle, front, back, etc...). Because the sets will contain specific CVs in some of these surfaces, you need to divide the geometry into parts based on CVs. For this process you will work and select in Component mode. Following is a diagram displaying the breakdown of the sets you will be creating.

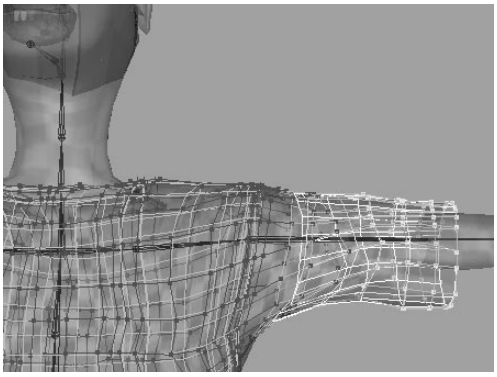


Shirt and shorts

1 Select CVs for the set

- Set the Component mode Pick Mask options to **select only CVs** and **Hulls**.
- **Select** the CVs of the shirt sleeves.

Tip: Avoid having CVs of one isoparm in two sets. When deciding where to cut off the membership for the shirt sleeves, select a few hulls down from the shoulder so you have about 3-4 rings of CVs going up from the end of each sleeve.



Selected CVs

2 Create a set for the shirt sleeves

- Select the **Edit** → **Sets** → **Create Sets** - ❐ and enter *shirtSleevesSKINSet* in the **Name** text field.

- In the **Add to a Partition** section, toggle on **Try to Add**.

  Notice the Partition pull-down menu is now highlighted.
- From the **Partition** pull-down menu, select the **skinSets** partition.

Create Set options

- Press **Apply**.

  You have now added this portion of the shirt geometry to the skinSets partition. You can easily select these CVs by selecting the set in the Set Editor and **Edit** → **Select Set Members** from the Relationships Editor **Edit** menu.

  As you add more sets to this partition (with **Try to Add**), Maya will warn you if you try to add a set with overlapping components.

Tip: If you use **Force to Add**, Maya will sort out the set membership for you. If another set in that partition contains any overlapping members, they will be pulled out of their set and forced into the new one.

---

3 Create a shirtBodySet and add it to the skinSets partition

With the sleeves CVs selected you can quickly select the body CVs by holding down the Shift key and marquee LMB dragging across the shirt. Thus toggle selecting the shirt body CVs.

- **LMB-Shift-Drag** select the shirt to toggle select the body CVs

---

Tip: If you accidentally select a **hull** that extends into the body, simply **Ctrl select** that hull, or **Undo**, and continue with **CV** selection.

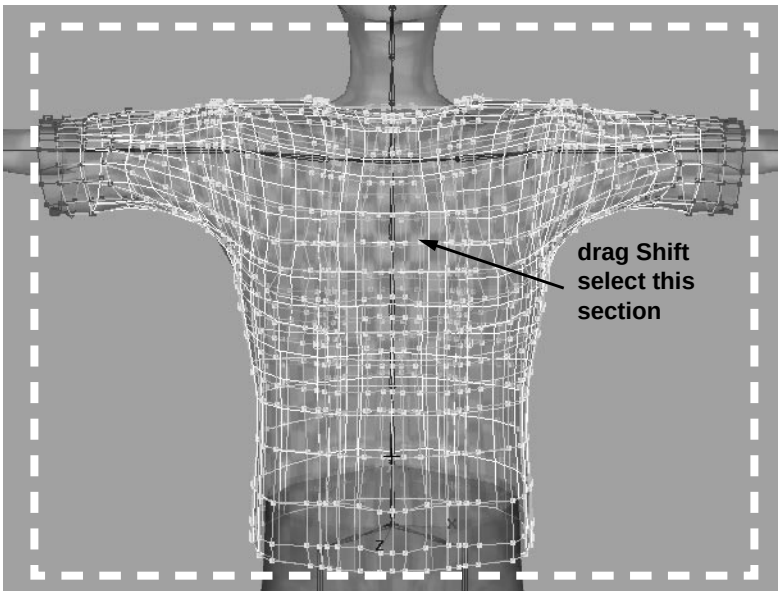
Select Modes

Shift Select toggles your selection.

Ctrl Select removes from the selection.

Shift Ctrl Select adds to the selection.

Use these different selection modes whenever possible to improve your workflow.



Selected CVs

- Select the **Edit** → **Sets** → **Create Sets** - ❒ and enter *shirtBodySet* in the **Name** field.
- **Try to Add** the set to the skinSets partition.
- Click **Apply**.

4 Save your work

Tip: To remove a set, select the set in the Relationships Editor and select **Edit** → **Delete Highlighted** from the Set Editing Edit menu. Or, with **show sets** selected in the Outliner you can select and backspace delete sets.

Creating arm and legs sets

1 Create sets for the arms and legs (including shoes and hands)

- Repeat the above steps to create the armsLegsSKINSet.

  Use the Outliner to select these objects. You aren't picking individual CVs here--pick the objects themselves.

Creating head and neck sets

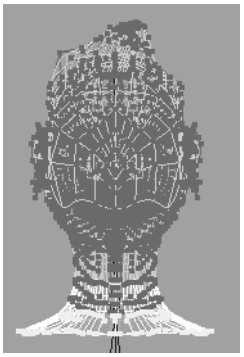
For this lesson you'll divide Melvin's head into two sets.

- **neckBaseSet** -base of Melvin's neck (for the shirt lattice)
- **headSKINSet** -Melvin's head (which will get the basic bind skin)

1 Create a set for Melvin's neck

- Hide the eyes, lids, teeth and glasses

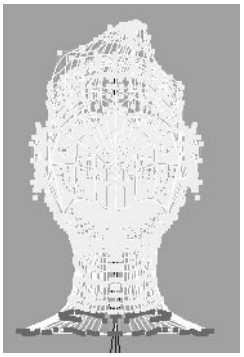
- Create a set for the base of Melvin's neck, called *neckBaseSet*, which includes the 2 hulls of the base of the neck.
- Select **hulls** and **CVs** until you have the base of the neck selected.



Selected CVs

- Add *neckBaseSet* to the *skinSets* partition.
- Create a set for the rest of the head, including the hair, face and ears. *The eyeballs, lids, teeth and glasses will be parented and do not need to be skinned.* Name the new set *headSKINSet* and put it into the *skinSets* partition.

  Shift-drag select the head with the neckBaseSet CVs selected to toggle your selection.



Selected CVs

Creating skeleton sets

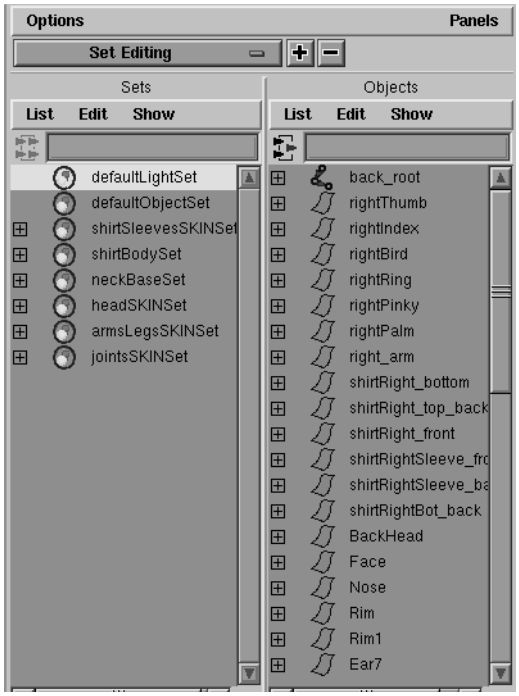
In the same way that you created sets for the geometry, you'll create a set for the skeleton. You don't have to create this set, but this will aid in the selection of all the objects that will go into the Bind Skin operation.

You can go through and select all the necessary sets in one place—the Set Editor. Selecting the objects in the work area can get quite messy and can require multiple attempts with a greater chance of error.

1 Create a set for the skeleton

- **Select** the *back_root* joint.

- Create a new set called *jointsSKINSet* and add it to the *skinSets* partition.
- Open the Set Editor and verify that the following list of sets appear in the *skinSets* partition.



Relationships Set Editor

Skinning with lattices and wraps

The indirect skinning methods provide a generalized skinning that allowing for smoother deformation when the joint is moved. Using lattices to Bind Skin is a useful technique for areas that are difficult to skin, such as the shoulders. Here the lattices will create a smoother deformation when the shoulder rotates and is easier to weight and control with Set Driven Key due to the fewer control points.

Creating the upper torso lattice

You'll set up a lattice around the mid-section of Melvin's upper torso. This lattice will be bound to the skeleton underneath. Because you've already created skin sets, you can easily select the CVs where you'll apply the lattice by selecting the sets in the Set Editor.

1 Select the shirtBodySet and neckBaseSets

- Select **Window** → **Relationship Editors** → **Sets...**
- Select the shirtBodySet and neckBaseSet sets.
- In the Set Editor **Edit** menu select **Select Set Members.**

2 Create a Lattice for the body and the neck

- Select **Deform** → **Create Lattice** - ❐.

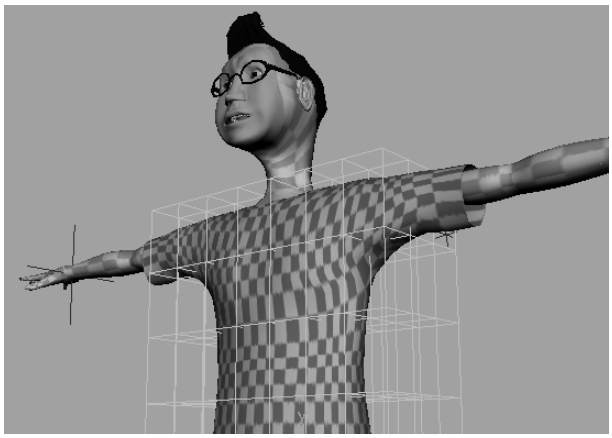
- Click **Reset** to return to the default values then enter the following:

  Enter **8, 5,** and **4** for the **S, T,** and **U resolution**;

  Select **Center Around Selection**;

  Select **Group Base and Lattice together**.
- Click **Create**.
- Label the lattice *shirt_lattice.*

Tip: The higher the resolution of this lattice, the smoother the resulting deformation—but it will also take longer to compute and assign weight.



Shirt Lattice

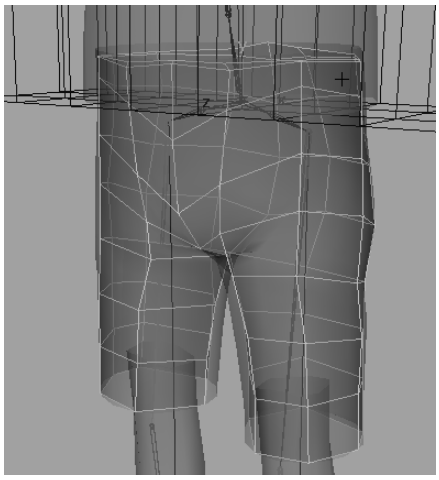
Creating the shorts wrap

Now you will set up a wrap deformer for Melvin's shorts. The goal is to have a more general control over the shorts patches. This makes weighting and other tasks simpler.

1 Create a wrap deformer for the shorts group of surfaces

- Select **File** → **Import** and select *wrapShorts.mb.*

  This is the low resolution "stand-in" object that you will to apply the wrap deformer to.
- **Cntrl select** the *shorts* geometry group and the *wrapShorts* objects from the Outliner.
- Select **Deform** → **Create Wrap**



shortsWrap deformer

2 Create a set for the lattice and wrap objects

- **Cntrl select** the *shirt_lattice* object and the *wrapShorts* object from the Outliner. Make sure the base objects are not selected.
- Select **Edit** → **Sets** → **Create Set -** ❐, and name the set *latticeWrapSKINSet* and add this to the skinSets partition

Binding lattices and skin to joints

Now you'll bind the lattice and wrap objects to your skeleton along with the other geometry that has been grouped into the sets of the skinSet partition. After you've built the sets using the partition's exclusivity properties, it is easier to select the component groupings—especially when they belong to the same object, like the shirt. This exclusivity goes a long way in ensuring that surface points and lattice components do not inadvertently end up in more than one joint cluster which can lead to double transformations when you move your character.

1 Binding "SKIN" sets to the skeleton

Now you'll select all the sets with "*SKIN*" in the name to bind skin.

- In the Relationships Editor, Set Editing section, select all the sets to bind skin. Assuming everything was named correctly, this will be any set that has "*SKIN*" in its name.

  Be sure that none of the geometry deformed by the lattices or the wrap is selected! These are already being transformed by the lattices.
- In the Edit menu for the Set Editor select **Select Set Members** to select the objects and components contained within the sets.
- Select **Skin** → **Bind Skin** → **Rigid Bind -**❐, and set the following:

  Bind to to **Complete Skeleton**;

  Color Joints to **On**;

  Bind Method to **Closest Point**
- Press **Bind.**

Note: The **Closest Joint** option binds the skin to the closest point. The **Color Joints** option applies the same colors to the joints that are used to distinguish the membership of the skin that belongs to that joint.

2 Test the results

Test the results of this bind skin, by moving the *back_root* joint and translating the wrist and ankle locators. Pay attention to the deformations around the shoulders and legs.

3 Save you work

Summary

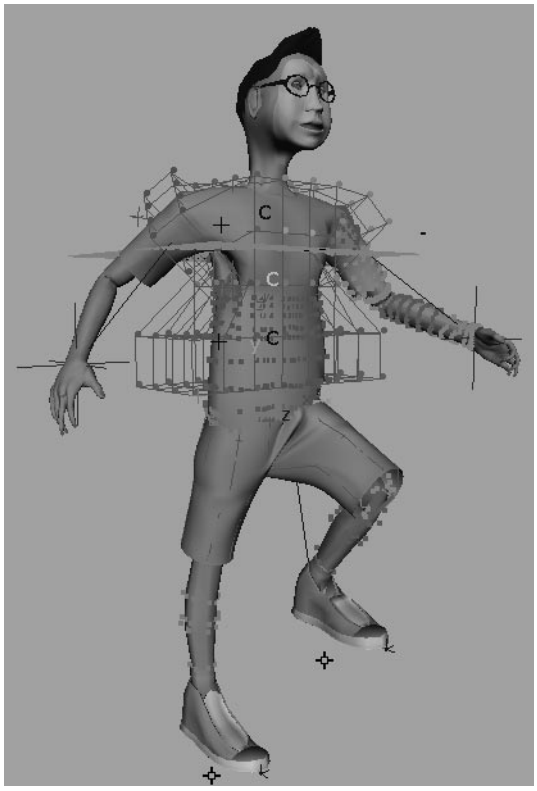
In this lesson you have learned how to:

- Use lattices with bind skin
- Use wrap deformer with bind skin
- Use sets and partitions to organize your work
- Bind skin to a skeleton
- Disable and enable IK Handles and constraints

Lesson 12

12 Set Membership

You've now bound all of Melvin's skin (lattices and geometry) to his skeleton. You will now look at making the deformations look better by weighting CVs and editing set membership.



Testing membership

In this lesson, you will learn the following:

- How to weight CVs
- How to edit Set Membership

CV WEIGHTING

As mentioned earlier, CVs can be weighted in clusters to get differing amounts of deformation.

Adjust cluster weights on a cylinder example

1 Create a cylinder

- Select **Create** → **NURBS Primitives** → **Cylinder**.

2 Make the cylinder into a cluster

- With the cylinder selected, select **Deform** → **Create Cluster**.

3 Weight the CVs in the Component Editor

- Select **Windows** → **General Editors**→ **Component Editor...**
- Switch to Component Mode
- Select the bottom row of CVs
- In the Component Editor, switch to the **Weighted Deformers** tab then press **Load Components** to display the CV weights.
- Change their weight to **0**
- Replace the selection with the next row of CVs above and press **Load Components** again.
- Set their weight to **0.33**.
- Select the next row above and set their weight to **0.66**
- Select the top row of CVs and make sure that their weight is at a value of **1**.

4 Test the new weights by rotating the cluster in X, Y or Z

5 Test the new weights by scaling the cluster in X or Z

This shows how CV weights in a cluster can vary the deformation and create bending, twisting and / or tapering effects.

Tip: Delete construction history to make the deformation permanent. This can be a good modeling technique.

EDITING SET MEMBERSHIP

Once you've bound the skin and lattices, you need to check the CV membership to ensure that each CV is only in one set and that each is in the appropriate set. As soon as you start to translate / rotate the joints, you will notice if any CVs are in the incorrect set.

Tips for editing set membership

- Watch for CVs that end up in more than one set. This results in double transforms. Using partitions can help to avoid this.
- It is better to add CVs to the desired joint rather than removing them

from the undesired joint. When you add them to a new joint, they are automatically removed from the original owner.

- Exaggerate the translation / rotation of the joint / bone when testing set membership. CVs with incorrect membership will become quite obvious.
- **Move** the root of the skeleton to check for correct membership.
- Take notes! Do a quick sketch of your character and take notes as to how you are dividing up the membership.

Editing set membership for the shirt lattice

You will be dividing Melvin's shirt lattice horizontally into slices which correspond directly to correspond with the back joints. There are some areas under the armpits that may need to be grouped with the upper arm joints instead of the back. The armpit area will likely require experimentation with grouping and weighting, and may possibly require some specialized bone structure to drive the motion of the shirt sleeves when Melvin drops his arms.

The following diagram shows Melvin's skeleton and shirt lattice. The lattice is the black grid.The dashed lines indicate how the membership should be edited. Note that the breakdown of memberships will differ from character to character. The regions are numbered to give you an idea of how the breakdown works:

A -shoulder joints

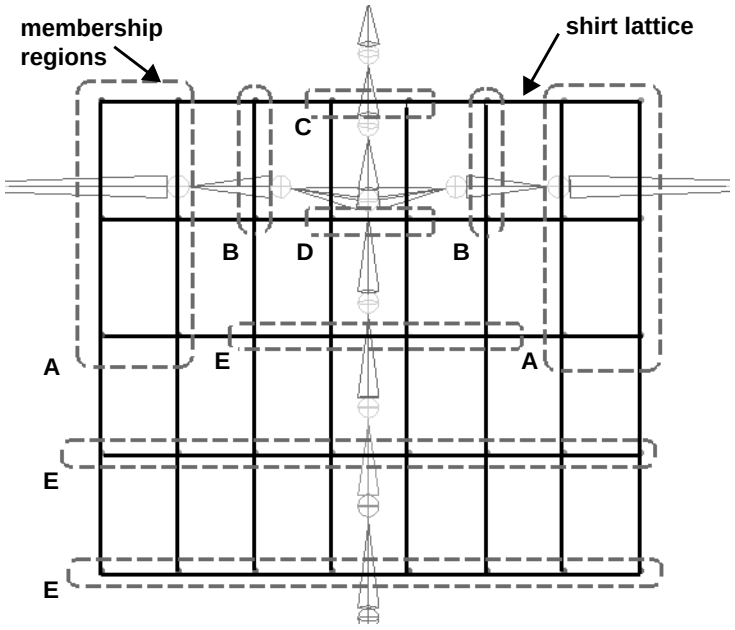
B -collar bone joints

C -neck joint

D -shoulder join joint

E -back joints

Use the diagram only as a general guideline for editing which lattice points belong to which joint.



Edit membership diagram

1 Open a file

Open the file named *Melvin_11_rigidSkinned.mb*

2 Edit the membership for the shirt lattice

- Select **Deform** → **Edit Membership Tool.**

  Notice that your cursor changes to a triangular shape to signify that you are now in a new pick mode. In the default pick mode, you can only select joints. When you select a joint, the CVs deformed by that joint will highlight. This is where you might see some obvious membership problems.

  Once you have selected a joint, you can **Shift**-select to *add* new CVs to the current set or **Ctrl**-select to *remove* CVs from the set.

- Select the joint where you want to edit the membership.
- **Shift-select** the CVs or lattice points you want to add to the current joint.
- Repeat for the other joints, continuing through the entire skeleton.

  You may find that when the hands are at the side, the shoulders are not rounded but have sharp corners. Try editing the membership of some of the lattice points between the shoulder and the collar joint. It also may require the skin to be rebound after increasing the resolution of the lattice.

Tip: The joint colors can be edited to have more contrast. Use **Options** → **Customize UI** → **Colors** to change the values of the user defined colors.

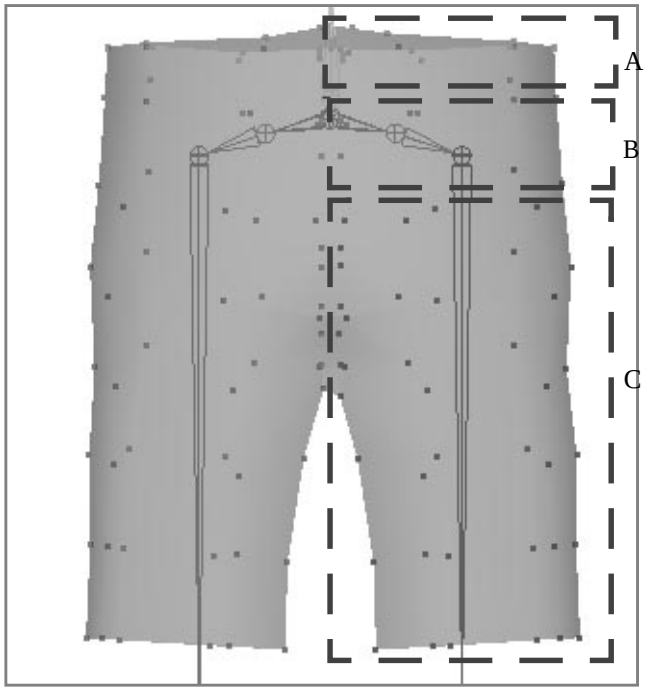
Editing set membership for the shorts wrap

This workflow is similar to the one above except you are now editing the wrap deformer weights instead of the lattice point weights. Keep in mind that the best membership differs depending on the model, the skeleton, and the resolution of the deformer. Use the following diagram as a general guideline for editing the membership of the left side of wrapShorts deformer. Use the mirror image for the right side. There are two rows of vertices in the center of the shorts. Be careful not to group both of these vertices into the same bone.one should go with the corresponding left bone and one with the right.

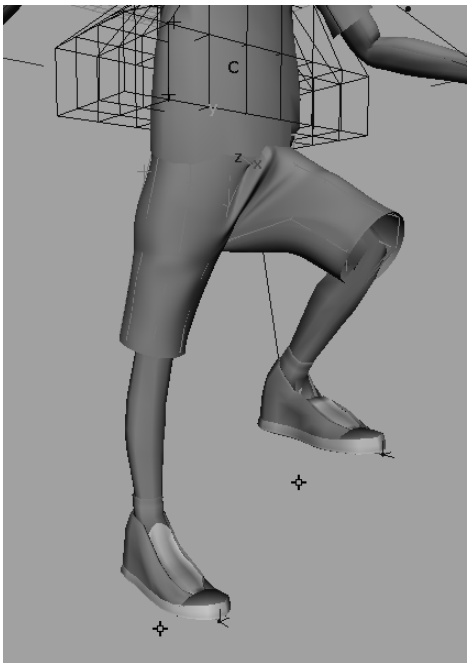
A - back_a

B - pelvis

C - hip



Membership areas for left side of wrapShorts.



Testing the bending

- Add all the CVs at the bottom of the shorts to the correct hip joint.

Tip: In edit membership mode, **RMB** click on a hull of the shorts geometry to select a whole row of CVs along an isoparm.

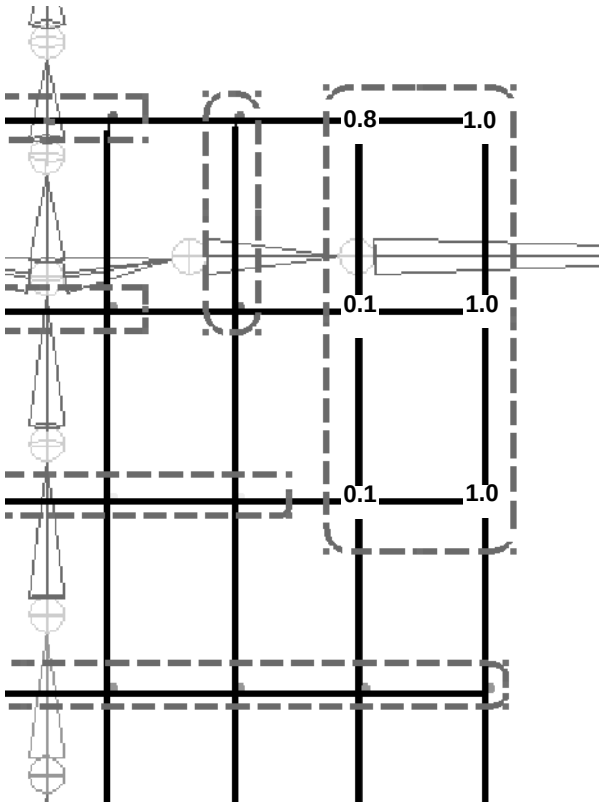
WEIGHTING LATTICE POINTS

When you lift Melvin's arms, notice that the shirt near the waist is pulled more than you want. If you change the weighting of the lattice points, you can control how much they affect the shirt.

Weighting the shirt lattice

Experiment with different lattice point weights to see how they affect the shirt when the character is moved. For Melvin's shirt you want to decrease the weights of the lattice points controlling the shoulder joints and near the armpit to between 0.1 and 0.3.

Use the following chart below as a guideline for weighting the shirt's lattice points. Note that you only changed the weight of the lattice points that are deformed by the shoulder joints. This is easy to see when in Component mode because Maya automatically colors the lattice points depending on what joint controls them.



Weighted lattice points

1 Set the weights for the shirt lattice points and CVs

- In the Set Editor, toggle on **List** → **Sets**.
- In Component mode, select the CVs or lattice points that need their weights changed. The default weight is 1.0.
- In the Set Editor, select **List** → **Update Now**. This updates the list to display the CVs or lattice points that are currently selected.
- In the Set Editor, select **List** → **Expand All frames**.
- Enter the new value for the weight in the text field at the bottom of the Set Editor.

Note: It is important to always type in the new value and press Enter on the numeric-keypad for the weight to be saved.

- Repeat steps 3-6 until complete. Remember to test the weights by moving or rotating the joint that deforms the selected CVs or lattice points.

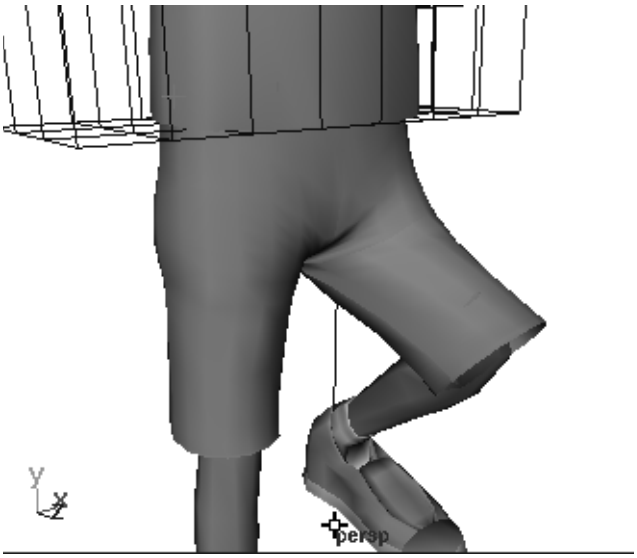
Tip: If possible, try to taper the weights so they gradually change from 0 to 1.0. A weight of 1.0 means the point will be deformed 100% by the joint and a weight of 0.0 means it will be deformed 0%.

Adjusting the weights of the shorts wrap (Optional)

You will need to taper the weights on the portion of the shorts lattice that is affected by the hip joints. This ensures that when Melvin's knee come towards his chest, the shorts don't move through his belly.

You also want to crease the shorts around the beltline. You can do this if you taper the weights of the hip lattice points from 0 at the top, to 1.0 near the middle of the leg. See the following diagram for a general idea of how to weight the shorts lattice.

The best overall method to smoothing out the skin values is to use the **Deform → Paint Weights Tool.** This tool works very similar to the weight painting you did in the smooth skinning lesson. Make sure you are smoothing the weight values on the wrap deformer since it is what is bound to the skeleton.



Smoothing applied near hip region of wrapShorts

Testing skinning with animation and poses

You haven't added any flexors and muscle control to the skinned character yet. Before doing that, you need to test the membership and weighting by putting Melvin into different poses. You should also consider setting some keyframes to verify that this skinning, membership, and weighting will prepare you to add flexors.

Below is a list of poses and animations to test the skinning:

- Move Melvin's hands to his hips - look at how the shoulders and

armpits deform. In some cases the membership and/or weighting might need to be adjusted to get the shoulders to round correctly and to crease the shirt under the armpits.

- Move Melvin's hands to his head. Don't forget to adjust the pole vector constraints.
- Make Melvin touch his right shoulder with his left hand. This is a very basic move, but it will display any skinning problems skinning.
- Move Melvin into a hurdling pose.
- Move Melvin into a crouched pose checking to see how the geometry deforms around his waist.
- Set up another walk cycle.

While testing up Melvin's movements you might find that you want to go back and re-skin him with the joints in slightly different places. You also may want to consider different resolutions for the lattices on his shirt and shorts. If possible, you'll want to get fairly decent deformations - otherwise it will be even harder to fine tune the skinning when you start adding flexors.

Exercise

Work on the weighting for the shirt, shorts, hands and fingers. You can use the file named *Melvin_12_rigidWeighted.mb*.

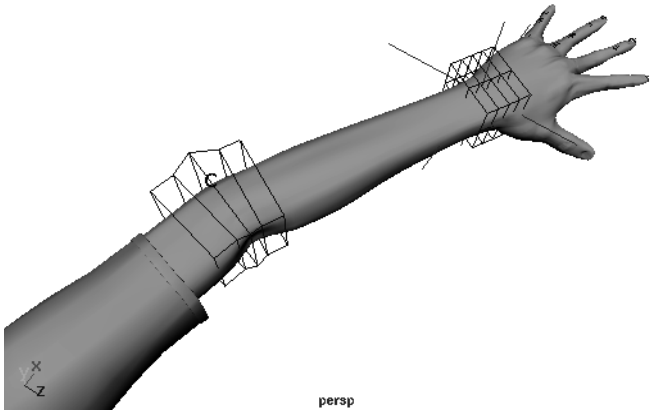
Summary

You should now have a good understanding of how to:

- Edit set membership
- Edit cluster weights
- Test membership and cluster weights

13 Lattice Flexors

Now it's time to fine tune the rigid skinning that you started in Lesson 11. Currently, Melvin's skin looks like a rubber suit and there are areas where the deformation needs to be either smoothed or controlled with more detail.



Lattice Flexors

In this lesson, you will learn the following:

- How to add a Lattice Flexor to Joints
- Adding Lattice Flexor to Bones
- Fine Tuning the Flexor with Default Attributes
- Fine Tuning the Flexor with Set Driven Key

Lesson 13

FLEXORS

When you Bind Skin with the Rigid Bind option, you will get fairly smooth deformations in most areas. But in certain regions where the joints rotate significantly (elbows, wrists, knees, ankles, etc.), you will need to add flexors to smooth out the deformations. In addition to smoothing out the deformations, flexors have some preset attributes that allow you to create realistic effects in characters—such as bulging biceps and creasing/rounding elbow joints. Lattices flexors have a generalized control on the skin. The two other flexor types (sculpt and cluster) provide different levels and types of skinning control. They will be discussed in the following lessons.

One thing to note is that flexors can only be used with skin bound with the rigid bind command. Smooth bound skin will not be affected by flexors.

Lattice Flexors

Lattice flexors are almost identical to lattice deformers. The difference is:

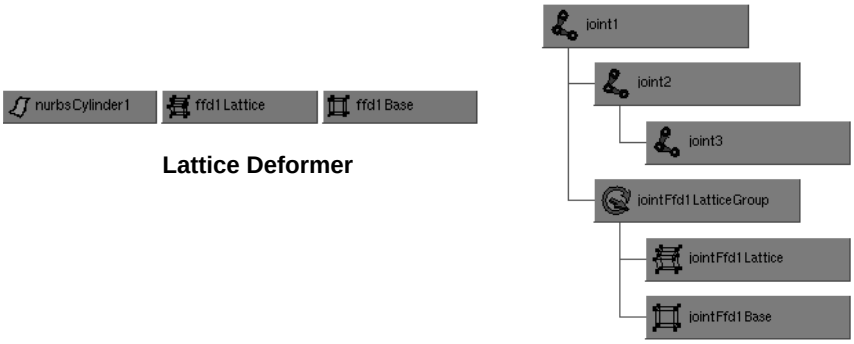
- lattice flexors are controlled by and parented to a joint
- lattice flexors have pre-defined attributes to control how they fold and bulge the skin.

Lattice flexors are ideal for smoothing skin around joints like elbows and knees. An advantage to using lattice flexors is that the underlying geometry they smooth doesn't have to be high resolution to get satisfactory results. In addition to smoothing, lattice flexors can also wrinkle skin and create muscle definition, depending whether they are added to a joint or a bone.

By default, lattice flexors are created with several preset attributes. You can also set up a Set Driven Key on the lattice points to fine tune the deformation.

Lattice Deformers versus Lattice Flexors

The best way to see the difference between lattice deformers and lattice flexors is to look at them in the HyperGraph.



Lattices in the Hypergraph

When you create a lattice flexor, a *jointFfd1LatticeGroup* is created under the *LatticeFlex_joint* as its child. This group is made up of a *jointFfd1Lattice* and a *jointFfd1Base*.

- The *jointFfd1Lattice* is what actually deforms the geometry.
- The *jointFfd1Base* is what stores the base position where the deformation starts.

The *jointFfd1LatticeGroup* looks like a regular lattice parented to the joint, but it also has some special attributes specific to deforming geometry bound to a joint.

Creating a lattice flexor on a joint versus a bone

When the flexor is created on the “bone”, the flexor is parented to the currently selected joint. If it is created on the “joint” it is parented to the parent joint of the currently selected joint.

There will also be different pre-set attributes depending on whether the flexor is created on the “joint” or the “bone”. The following attributes are created for joint lattice flexors. If any of these attributes are changed when the elbow is bent, a pseudo-Set Driven Key will be created. When the elbow is rotated back to its bind pose, the attributes will return to their default settings. The attributes will be driven by the joint on which the flexor was created. In other words, a bone flexor created for a bicep will be driven by the rotation of the elbow, not by the rotation of the shoulder.

The following table contrasts the attributes that are created.

Joint Flexor	Bone Flexor
Creasing	Bicep
Rounding	Tricep
Length In	Length In
Length Out	Length Out
Width Left	Width Left
Width Right	Width Right

Joint Flexor vs. Bone Flexor

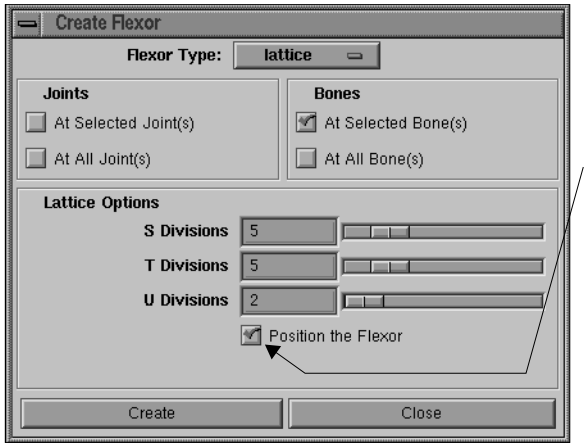
Selecting LatticeGroup

To easily select the lattice-Group, select the lattice in the work area, then click the up arrow to select the node just above the lattice.

Create Flexor options

Most of the options for flexors are the same as options for the deformer type flexor but a couple are different. One option is bone or joint flexor. The other is **Position the Flexor**. This selects the group node above the lattice and the base after creation so that they can be translated, rotated, or scaled without deforming the geometry.

Note that the flexor can be transformed at a later time by selecting the *jointFfdLatticeGroup* which contains both the lattice base and the lattice.



Flexor options

LATTICE FLEXORS FOR MELVIN'S ARMS

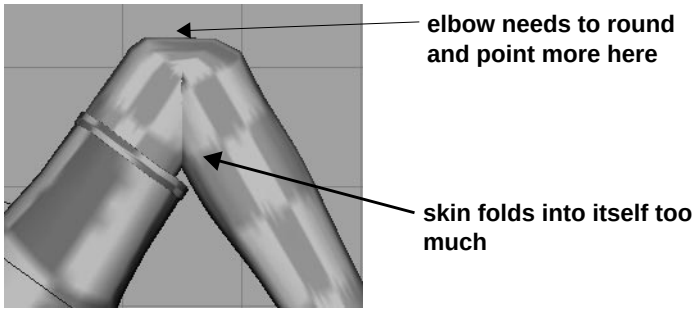
Fine-tuning Melvin's skin with flexors can be a tedious, trial and error process depending on how "tuned" you want the skin to be. You will work through some of the basics of getting skin to behave in a desirable manner. The best approach is to tweak the skin in areas where it folds and bends, such as elbows, wrists, knees, and ankles.

The Bind Skin function from the previous lesson will give you a good starting point. Flexors help take the skinning to the next level of realism. In

addition to skin folding, you'll explore ways to bulge skin to mimic real muscles and bones. You'll also look at techniques to deform the shorts and t-shirt.

Elbow

When the elbow is bent, the elbow folds into itself and you lose any sense of an underlying skeletal structure. Use a lattice flexor to help this area crease and round out a little bit at the tip of the elbow.



Elbow bending

You will use a combination of flexors to get the desired behavior for the elbow. It is important to understand Maya's ability to combine multiple flexors / deformers on a single piece of geometry because it is really helpful for tuning the skin to the correct shape. In later lessons you will look at how to combine multiple flexors to solve the elbow problems.

Tip: Always add flexors when the skeleton is in its bind pose.

---

1 Open an existing file

- Open the file *Melvin_13_readyForFlexors.*

2 Add a flexor to the left elbow

- From the bind pose, add a **Lattice** flexor to the elbow joint.
- Select the *left_elbow* joint.
- Select **Skin** → **Edit Rigid Skin**→ **Create Flexor...** and set the following:

  Flexor Type to **lattice**;

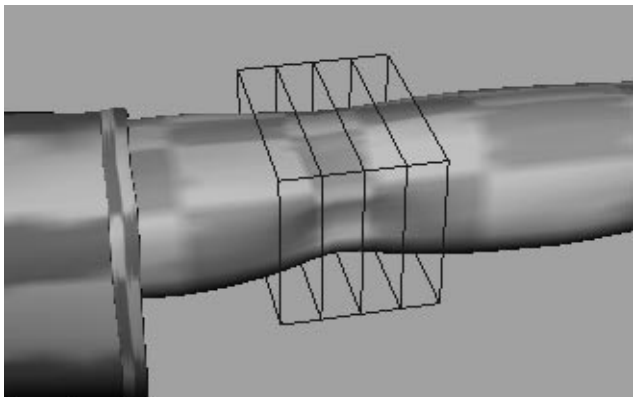
  Joints to **At Selected Joints**;

  Lattice Options to **(S=2, T=5, U=2)**;

  Toggle on **Position Flexor**.
- Click **Create**.

3 Position the flexor

- Position and scale the flexor so that it surrounds the elbow area as in the following diagram:



Lattice on elbow

- Label the flexor *left_elbowLatticeFlexor* and the group node above *left_elbowLatticeGroup*.

4 Adjust the parameters of the flexor

Bend the elbow, then adjust the flexor parameters to get the crease and roundness to look correct.

- Select *L_wristLocator* and translate until the elbow is bent at about **90-120** degrees.
- Select *left_elbowLatticeFlexor* and adjust the following attributes. Experiment a little then test the results by bending the elbow. These are some approximate values that work well for rounding the elbow:

  Creasing to **2**;

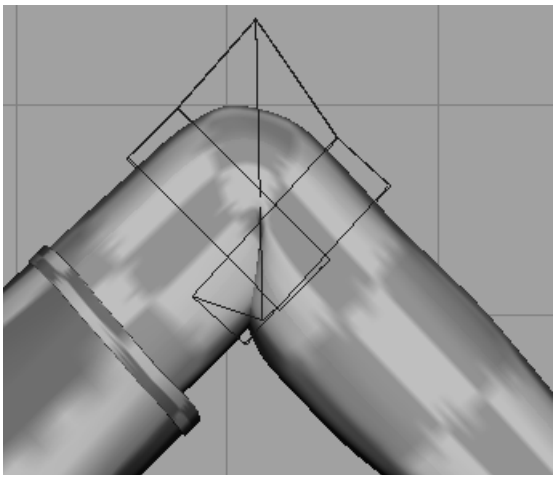
  Rounding to **3**;

  Length In to **-2**;

  Length Out to **-2**;

  Width Left to **-1**;

  Width Right to **0**



Bent flexor

Forearms

You can use a Lattice flexor to smooth out the deformations caused when the forearm is twisted. Currently, if you rotate the wrist/forearm, the deformation is a little harsh and needs to be tapered down the length of the forearm.

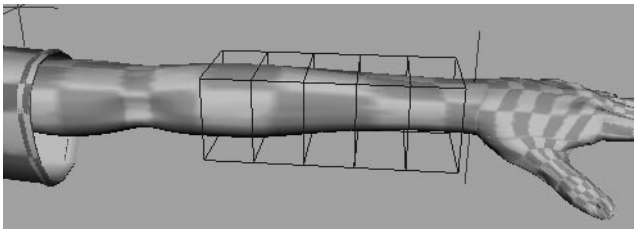
1 Add a lattice joint flexor to the left forearm joint

- Add a lattice joint flexor to the *left_forearm* joint with the following setting:

  T Divisions to **6** to **8**.

  Toggle on **Position the Flexor**

- Position the flexor, scaling it along the length of the forearm, as shown in the following figure.



Forearm lattice

- name it *left_forearmLatticeFlexor*

2 Test the flexors

- To judge how the forearm flexor smooths out the deformation, select the *wristLocator* and twist the wrist joint (which drives the forearm joint's rotate x).

3 Repeat for the right forearm

Note: A similar approach can be used for the wrists.

ANKLES AND SHOES

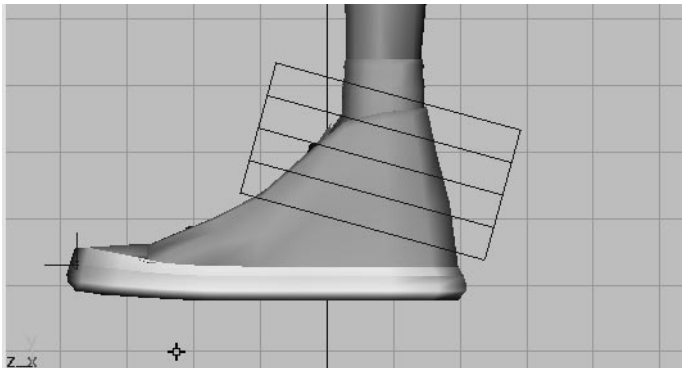
You need to add a flexor to the ankle joint to smooth out the deformation when the ankle is rotated. Depending on their membership, some of the shoe patches might fly into space when the knee and ankle are bent. Adding a flexor can help solve this problem.

1 Add a Lattice flexor for ankle bend

Adding a lattice flexor for the ankle will resolve problems similar to those encountered with the elbow and the forearm.

- Add a Lattice flexor to the *left_ankle* joint and name it *left_ankleLatticeFlexor*.
- Position the flexor as shown in the following figure. Remember to toggle on **Position the Flexor** first.

2 Repeat for the right foot



Positioned flexor

Correcting shoe membership

When you check the ankle behavior, you will you notice that some of the CVs in the back of the shoe should probably belong to the ankle joint. The more CVs of shoe geometry that are deformed by the ankle joint, the more rigid the shoe will look. And conversely, the more CVs of the shoe that the knee deforms, the more rubbery the shoe will appear when the ankle bends.

You won't add all the CVs to the ankle joint, but you'll make sure the sole of the shoe near the ankle belongs to the ankle joint. You need to shift this membership to the ankle joint to make the shoe look less like a pair of socks and more like a shoe.

- Add the CVs from the back of the shoe to the ankle joint.
- Test the membership by rotating the foot.

- Adjust the flexor attributes as necessary. Try changing some of the values for crease to make the shoe fold into itself a little more than normal.

CONTROLLING CLOTHING WITH FLEXORS

Currently Melvin is clothed in typical CG character attire—a seemingly rubber suit. You'll discuss some techniques to help his clothing act and look a little more natural. To make the sleeves look more like fabric, you could:

- deform them so they appear to collide with the shirt and react to a pseudo-gravity. One approach would be to create an "**underarm bone**". This approach requires you to rebind the skin.
- **Set Driven Key on a lattice flexor** - Currently the default flexors created so far are driven by the elbow joint. For the shirt sleeve to deform when the arms are positioned at Melvin's side, the lattice flexor needs to be driven by the shoulder, not by the elbow rotation. This approach requires that you set up a custom Set Driven Key on the flexor's lattice points.

Melvin's Sleeve

The following lesson will discuss setting up a Set Driven Key flexor on Melvin's left shirt sleeve.

1 Create a lattice flexor for the sleeve

For this exercise, keep the resolution of the lattice flexor relatively low to make the SDK setup simple when creating the lattice flexors on the shirt sleeves.

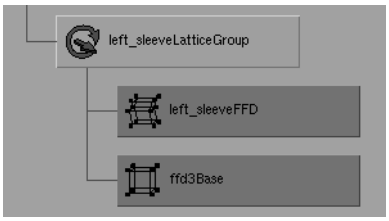
- Select *left_shoulder* joint.
- Select **Skin** → **Edit Rigid Skin**→ **Create Flexor...** and use the following settings:

  Flexor Type to **lattice**;

  Bones to **At Selected Bone(s)**;

  Lattice Options to **(S=3, T=3, U=3)**;

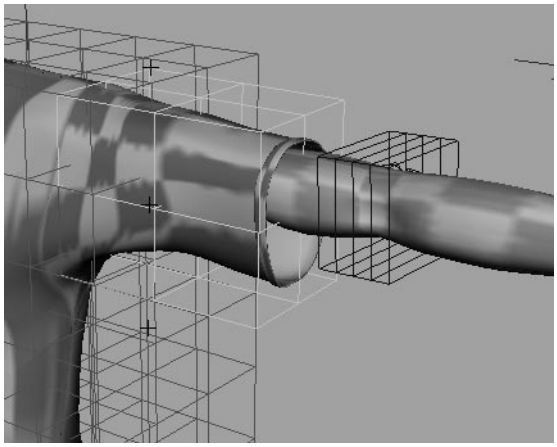
  Toggle on **Position Flexor**
- Click **Create.**
- Label the flexor FFD *left_sleeveFFD* and the flexor group *left_sleeveLatticeGroup.*



Flexor nodes

2 Position and scale the flexor

- Position and scale the flexor so that it only encompasses Melvin's left sleeve.



Sleeve flexor

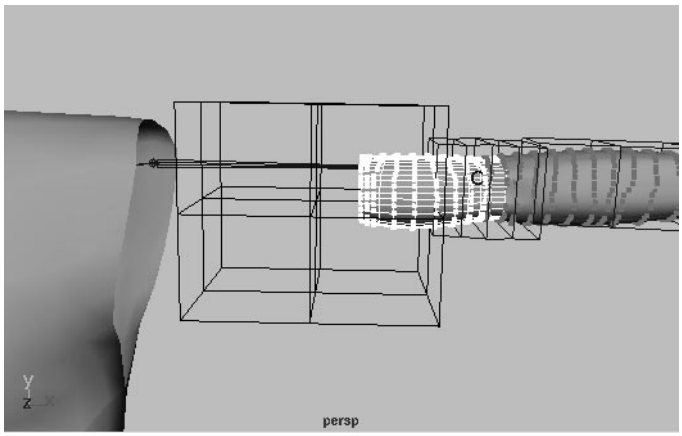
3 Edit the Membership of this flexor

Edit the Membership of this flexor so that it includes only geometry in the sleeve. Currently, it includes geometry in both the sleeve and the upper arm above the elbow.

- Hide the left sleeve geometry to make the membership editing easier.

Note: You don't want to change the membership of the shirt - only the arm geometry.

- Select **Deform** → **Edit Membership Tool**.
- Select *left_sleeveFFD*.
- **Ctrl-Select** over the highlighted CVs on the upper left arm. This will remove them from the membership of the lattice flexor.



Highlighted CV's

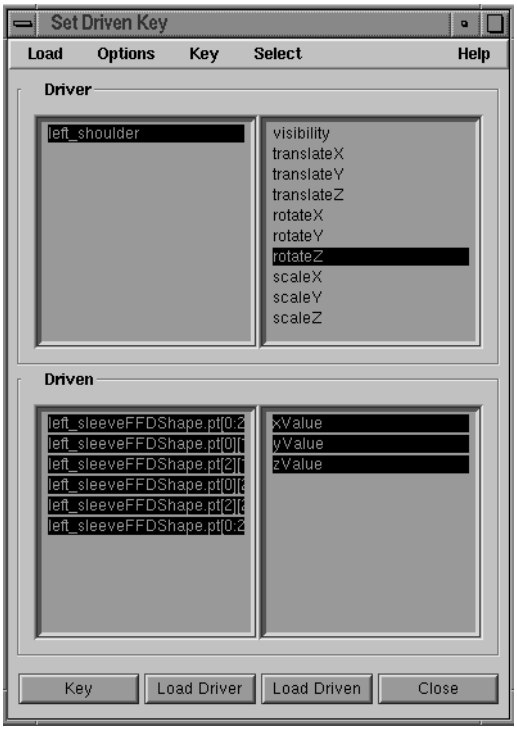
- **Display** → **Show** → **Show Last Hidden** to bring the sleeve geometry back into view.

Setting up the Sleeve Motion

To create the motion for the sleeve, you will use SDK to animate the position of the lattice points, depending on the rotation value of the shoulder joint. Limiting the number of lattice points that are animated with SDK will simplify the setup process. Select only the necessary lattice points when preparing the Driven objects in the SDK window.

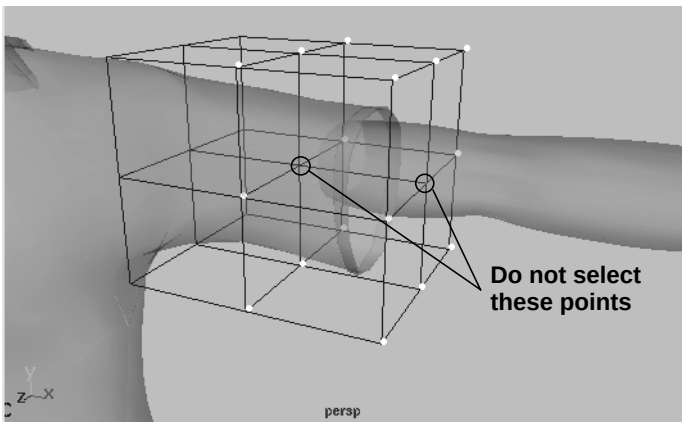
1 Create Set Driven Key for the sleeve at bind pose

In the SDK window, you will select the left shoulder joint as the Driver and the lattice points as the Driven objects.



Set Driven Key window

- Select **Animate** → **Set Driven Key** → **Set -** ❐.
- Select the *left_shoulder* joint.
- Click **Load Driver**
- Select **rotateZ** from the list of attributes in the right column.
- **Select** the lattice points nearest the hem of the sleeve.



Selected lattice points

- Click **Load Driven.**
- Select **X,Y,Z Value** from the list of attributes in the right column.
- In the SDK window, click **Key.**

This sets a key with *left_shoulder* joint at its rest (bind) pose and all the lattice points at their default position.

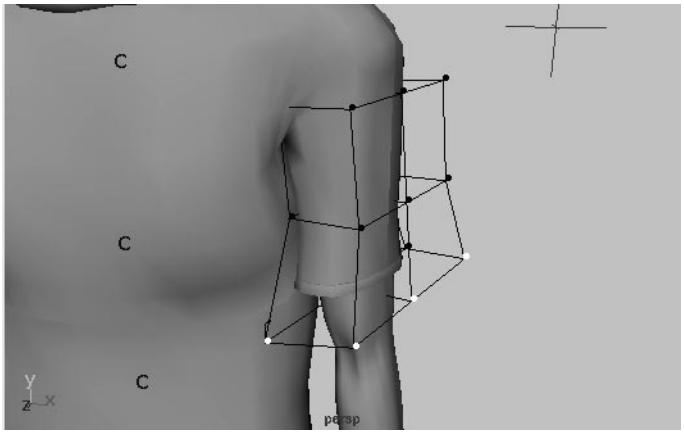
2 Create Set Driven Key for the sleeve at arm down pose

You will set a key with left shoulder joint rotated down to Melvin's side and the lattice points in a position that makes the sleeve looked folded and flared.

- Translate the *left_wristLocator* so that Melvin's arm is at his side.

Note: The sleeve may intersect into Melvin's shirt - this is what you will be correcting by moving the lattice points.

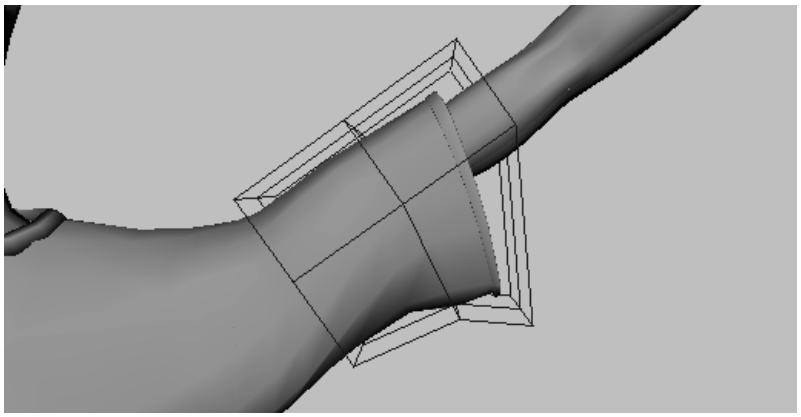
- Position the same lattice points to look flared.
- Press **Key** in the Set Driven Key window.



Repositioned lattice points

3 Create Set Driven Key for the sleeve at arm up pose

- Translate the *L_wristLocator* so that Melvin's arm is straight out above his shoulders.
- Position the lattice points to look folded and heavy.
- Press **Key** in the SDK window.



Third position

At this point, you have set up a general control that automatically gives the shirt sleeves a little more life. The next step is to start tweaking the SDK curves so that the lattice points' animation curves are not linear, but react more like clothing, bunching in places and sliding quicker when the arm raises up. Later, you will learn a technique to gain more specific control on deforming Melvin's skin and clothes.

Summary

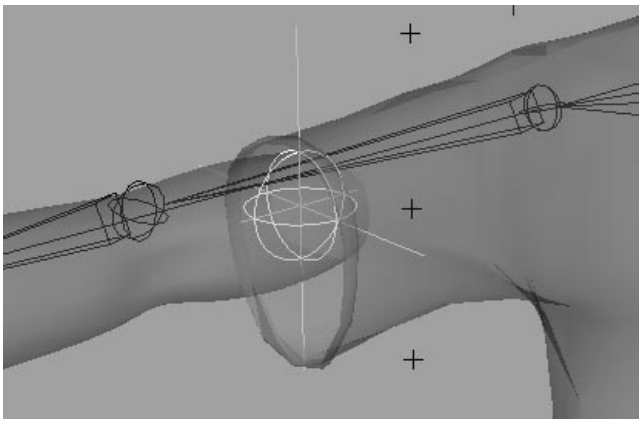
You have now completed the following tasks:

- Added lattice flexors to create more realistic character deformations
- Refined the deformations of lattice flexors
- Used SDK to drive lattice flexor deformations

Keep these things in mind as you look at sculpt flexors, cluster flexors and blend shapes so that you can contrast all of the options available.

14 Sculpt Flexors

Sculpt flexors allow you to build structures under the skin which can then be animated and scaled to mimic objects that the skin covers, like bones and muscles. Sculpt flexors can also be used to smooth a skin's deformation around a joint, however the skin geometry must have enough isoparms or vertices for the sculpt object to be effective.



Sculpt Flexor

In this lesson, you will learn the following:

- How to work with sculpt flexors
- How to add sculpt flexor to bones
- How to fine tune the flexor with SDK

Lesson 14

SCULPT FLEXORS

Sculpt flexors are best used to bulge a character's skin (e.g. biceps, triceps, tummies, etc.). You will get the most effective results if you use them to deform more general areas. In order for sculpt flexors to be successful in detailed areas, such as deformation around the elbow, they need to collide with a lot of isoparms / vertices.

Before adding sculpt flexors to Melvin's right arm, we'll look at how sculpt deformers work in Maya. Once you understand the basic concept of sculpt deformers, using them as flexors will be very similar.

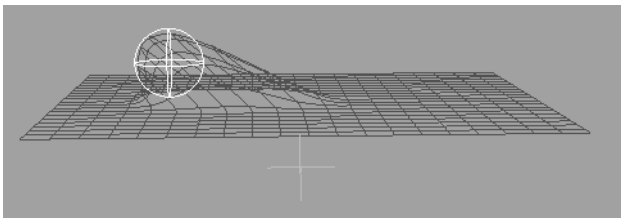
Basic Sculpt Deformers

A sculpt deformers is similar to a collision object for CVs, lattice points, and poly vertices. This deformer is located at **Deform→ Create Sculpt Deformer**. There are three types of sculpt deformers:

Stretch - this is the default and the most commonly used mode for Flexors and other deformations. There are two parts to the sculpt deformer when created in stretch mode:

Sculpt Sphere - this defines the area of the sculpt object's force field. This deformer shape is always elliptical. It will attempt to keep CVs, lattice points, and / or poly vertices outside

Sculpt Origin - this defines the direction that the Sculpt Sphere will be deformed. This deformer shape is a Locator. Moving it around in relation to the Sculpt Sphere and the deformed points will change the direction of the deformation. By default this shape is created in the middle of the Sculpt Sphere.



Sculpt

Flip - this mode is similar to Stretch, except that the Sculpt Origin is automatically hidden and parented to the Sculpt Sphere, so that it will always stay inside the Sculpt Sphere.

Project - literally projects the deformed CVs, lattice points, and / or poly vertices onto the Sculpt Sphere.

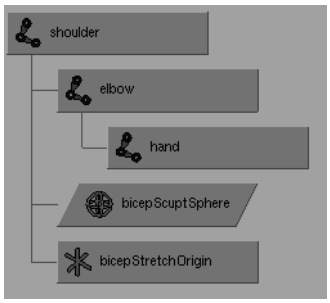
Sculpt Flexors

Sculpt flexors are almost identical to sculpt deformers. This deformer is located at **Skin → Edit Rigid Skin→ Create Flexor....**

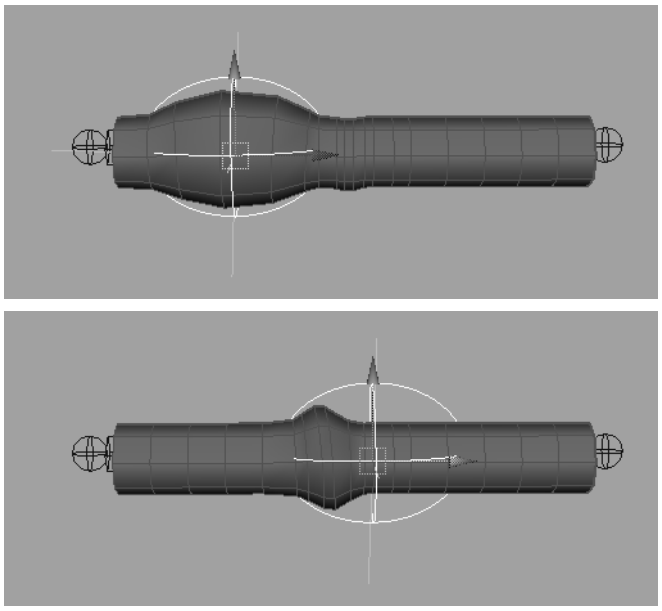
The differences are that when the sculpt object is created with the Create Flexors window:

- the Sculpt Sphere and corresponding origin automatically get

parented to the selected joint.

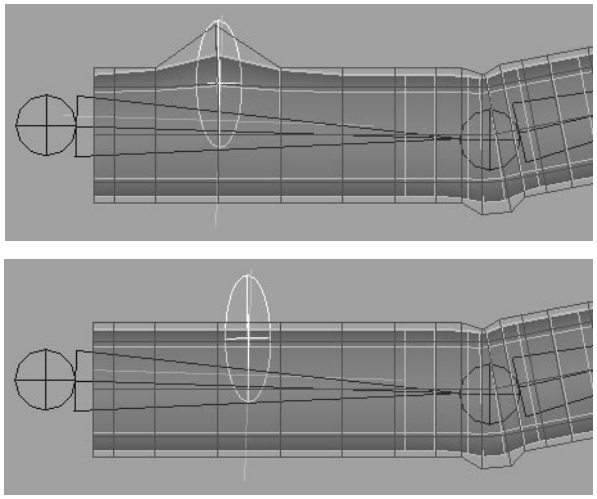


- the Sculpt Sphere automatically deforms a certain section of CVs depending on what joint is selected.



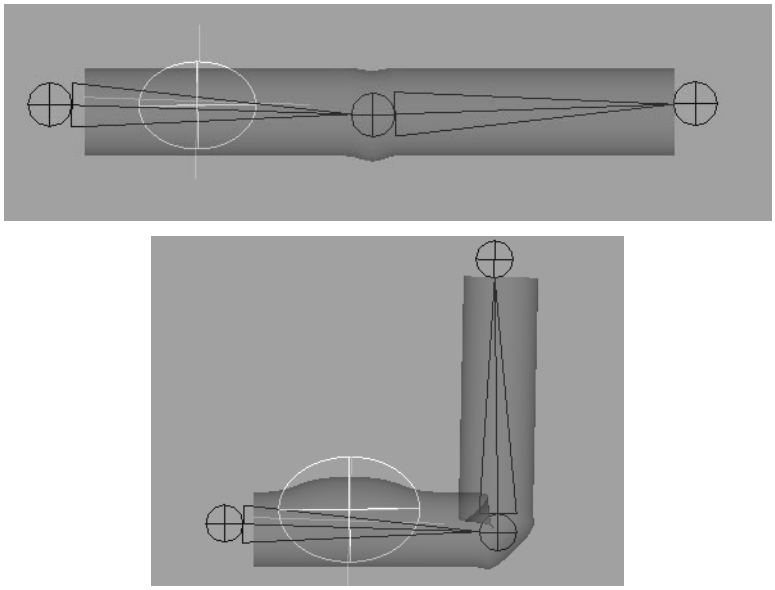
Sculpt

Sculpt flexors are best used on parts of the body where there are plenty of CVs, lattice points, and or poly vertices to intersect with. They are best used on biceps, triceps, stomachs, and other large general areas. Trying to use them on detailed areas will require that you increase the resolution of the deformed object - otherwise you will see the deformed object “pop” as the points snap onto and then off of the Sculpt Sphere.



Sculpt flexor

Using Set Driven Key you can make the Sculpt Flexor deform and bulge based on the rotation value of the elbow joint.



Sculpt effect

SCULPT FLEXOR FOR MELVIN'S ARMS

You'll use a sculpt flexor for Melvin's right bicep. Ultimately the sculpt objects needs to be parented to the shoulder joint but deform the skin between the shoulder and the elbow - so it is important how this flexor is created.

1 Open an existing file

- Open the file *Melvin_14_readyForSculpts.mb*.

2 Add a sculpt flexor to the shoulder bone

- Return to bind pose.
- Select the *right_shoulder* joint.
- Select **Skin** → **Edit Rigid Skin**→ **Create Flexor...** and set the following options:

  Flexor Type to **sculpt**;

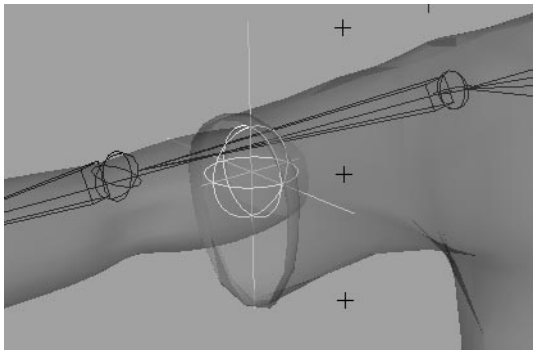
  Dropoff Type to **None**;

  Mode to **Stretch**;

  Inside Mode to **Even**.
- Press **Create**.

3 Position and scale the Sculpt Sphere flexor

Position and scale the Sculpt Sphere flexor so that its default size is inside the bicep area.



Sculpt object

- Select the *sculptSphere* and the *sculptStretchOrigin* and scale and translate until they are just under the skin where the bicep would be.
- Label the two flexor objects *right_shirtSculptSphere* and *right_shirtStretchOrigin*.

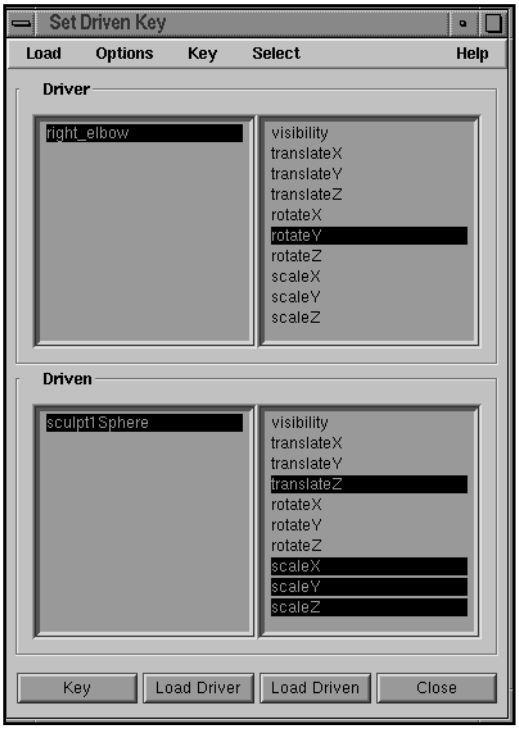
4 Save your work

Setting up the biceps bulge

Although Melvin's biceps are obscured by his t-shirt, you can add a little deformation using the sculpt object. Using Set Driven Key, animate the scale and position of the Sculpt Sphere with the rotation value of the elbow joint.

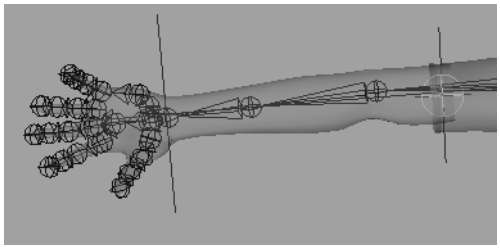
1 Create Set Driven Key for biceps

- In the Set Driven Key window, select *right_elbow rotateY* attribute as the Driver and the *right_shirtSculptSphere* the Translate and Scale attributes as the Driven object.



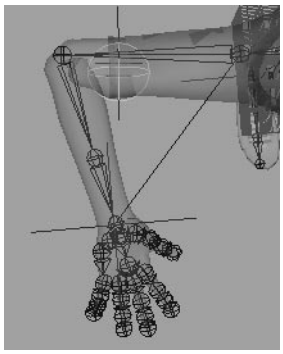
Set Driven Key window

- Set a key with *right_elbow* joint at its rest (bind) pose and sculpt sphere at its current (small) position and scale.



Arm extended

- In the Set Driven Key window, click **Key.**
- Set a key with *right_elbow* joint bent in a **90-120** degree bend in the Y-axis and the sculpt Sphere translated and scaled to bulge the shirt a little.



Flexor with arm bent

- **Translate** the *R_wristLocator* so that Melvin's arm is bent.
- Set a key with *left_shoulder* joint rotated down to Melvin's side and position the muscle in a relaxed position.

2 Save your work

Additional Exercises

If you have extra time, you can make the sculpt deformation more complex by:

- animating a few extra sculpt flexors when Melvin's arm bends. By adding more sculpt objects to deform his shirt you can give the appearance of a more complex muscle system underneath.
- The way the sculpt objects are animated will also affect the deformation - currently you only animated the bulge by translating and scaling the sculpt object. If you rotate a sculpt object that is elliptical OR translate it along the surface of the skin, it will give the appearance that the muscle is moving underneath the skin or shirt's surface.

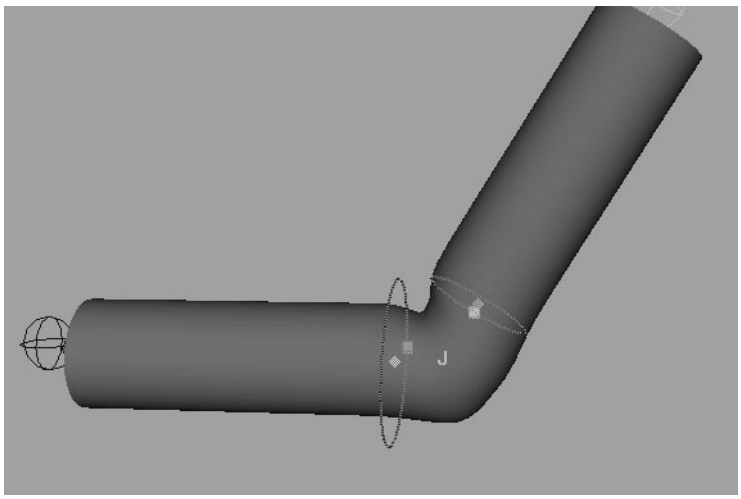
Summary

You have now completed the following task:

- Applied a sculpt flexor to create bulges
- Used SDK to control the sculpt flexor's movement

15 Cluster Flexors

Joint Cluster Flexors are the third type of flexors that can be created in Maya. While the Lattice and Sculpt flexors offer a more generalized smoothing, cluster flexors allow you to tweak the flexor's influence in more detail.



Cluster Flexor

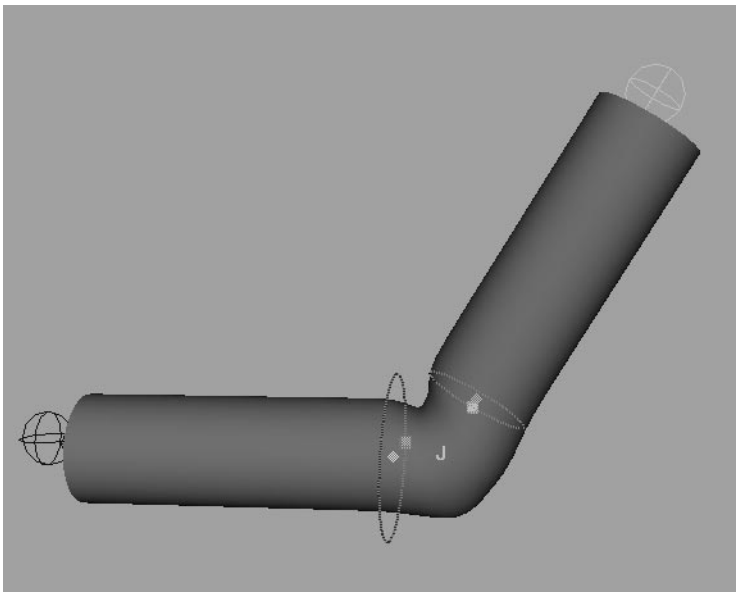
In this lesson, you will learn the following:

- How to add a cluster flexor to Melvin's right elbow
- How to fine tune the flexor with the manipulator and the Component Editor

Lesson 15

BASIC CLUSTER FLEXORS

Cluster Flexors provide more localized control than a lattice flexor, allowing you to set the individual deformation percentages (weights) of all the CVs that are affected by the flexor. Maya's joint cluster flexors have a manipulator that provides fast workflow for defining the upper and lower *bounds* of the flexor's influence.



Cluster Flexor

By selecting the elbow joint and selecting **Modify** → **TransformationTools** → **Show Manipulator Tool,** the 3D manipulator will be displayed.

- The rings designate the upper and lower *bounds* of influence on the flexor. By middle mouse dragging the center diamond, these *bound* positions can change. These values can also be changed in the attribute editor.
- The radius of the rings can be changed by middle mouse dragging on the diamond at the edge of the ring. This will change the percentage / weights.

The manipulator changes can be seen if you open up the Component Editor. You will notice the weights respond accordingly when the manip's rings are translated. The manipulator provides a quick way of setting the upper and lower *bounds.*

Note: To save the selected components in the Component Editor while selecting the joint cluster in the viewport, turn off **List** → **Auto Update** in the Component Editor.

Cluster Flexor For Melvin's Right Elbow

Following is the workflow for creating a joint cluster on Melvin's right elbow.

1 Open an existing file

- Open the file *Melvin_13_readyForClusters.mb*

2 Create the flexor on the elbow

- **Select** the elbow joint.
- Select **Skin→ Edit Rigid Skin→ Create flexor...** and set the following:

  Flexor Type to **jointCluster**;

  Joints to **At Selected Joint(s)**
- Press **Create.**

  You should see a letter "**J**" at the joint, representing the joint cluster.

3 Manipulate the values on the flexor

- Select the elbow joint and rotate it to about 90 degrees in Y

Note: You may need to disable IK handles to rotate the joint

- **Select** the joint the flexor is on. This will automatically select the joint cluster flexor.
- **Select** the **Show Manipulator Tool.**

Tip: Visualizing the flexor effects can be easier in shaded mode.

- Change the manips to see the effect and to make Melvin's elbow look good in a bent pose. You will likely want to bring the *bounds* closer together to get a more shapely elbow.

How it works

The name jointCluster can be misleading, because a new cluster is not created. This type of flexor affects the clusters on either side of the joint (in this case of the elbow and shoulder). Changing the upper/lower bound or value affects the weighting of the two clusters on either side of the joint.

You will have to switch between the inputs for the elbow and shoulder clusters in order to edit the values in the Channel box. It is important to know that the lower bound on the shoulder cluster is the upper bound of the flexor, and the upper bound on the elbow is the lower bound of the flexor.

The cluster flexor can be a good choice for making finger knuckles. There are a lot of knuckles to setup and a cluster flexor looks pretty good by

default. If the scene is not a close up of the hand you might not have to do any other work. A cluster flexor does the job with less overhead, keeping your scene cleaner as well.

Exercise

Now that you have seen all of the different flexor types, finish adding flexors to Melvin wherever necessary. Use your judgement as to which type you like best for each situation. If you're unsure, take time to experiment.

Summary

You have now completed the following tasks:

- Applied a joint cluster flexor

16 Facial Deformation

This lesson will focus on the key workflow associated with setting up a face for lipsyncing.

Facial deformation using Blendshape and Skin Clusters

You will combine a number of concepts and Maya tools to develop the controls for Melvin's facial animation.

- **Blend Shape**—for key poses, expressions, mimicking facial motion and correcting bound skin deformations.
- **Bind Skin**—using bones to control the jaw and head, and to help set up the Blend Shape poses.
- **Set Driven Key**—on attributes that drive the Blend Shapes to provide controls which are accessible through a custom window or in the Channel box.
- **Artisan**- weighting of surface components.

Lesson 16

BLEND SHAPE

Blend Shape is a powerful morphing tool. You can build several shapes and use blend shape to link them together. Blend Shape will create sliders so you can have smooth transitions between shapes after you put keyframes on the sliders.

In the most basic sense, a Blend Shape consists of two things: the *target* shape(s) and the *base* shape. It's easy to understand the basics of the Blend Shape by examining how Maya executes the Blend Shape function.

When executing a Blend Shape, Maya looks at the changes between the base and the target shapes. The value of the Blend Shape (which ranges from 0 to 1) is a percentage of how much of that difference will be added to the base shape.

SETTING UP MELVIN'S HEAD

Integrating a head (with Blend Shapes) and a pre-skinned body requires some planning. Determining how and when to bind all these parts together is a *major* decision. While Maya offers several techniques for combining parts, you may find that it is easiest to bind all the skin (head and body) together at the same time, regardless of whether the head and body are modeled separately.

This lesson shows the more challenging process, which is to skin, weight, and set up the Blend Shapes on the head as a separate entity from the body. This will require that you eventually attach the skinned head to the skinned body, covered later in the *Attaching the Head* lesson.

The basic workflow is:

1. Create blend shapes
2. Skin the head
3. Adjust weights and membership

Strategies for Facial Blend Shapes

When setting up poses for Blend Shape, these guidelines should be followed:

- Duplicate the face (base shape) in its neutral position.
- Set up Blend Shapes for the entire head rather than for individual patches.
- Pay close attention to where seams meet and maintain their tangency. You do not want weighting or membership changes to occur at the seam. Likewise you do not want any deformations to pull apart the seam between patches. Lattice deformers are a good way to avoid this because they work more evenly across these seams.
- Order the hierarchy of the geometries involved so that the base and targets appear in the outliner or hypergraph in the same order.

Setting up Blend Shapes

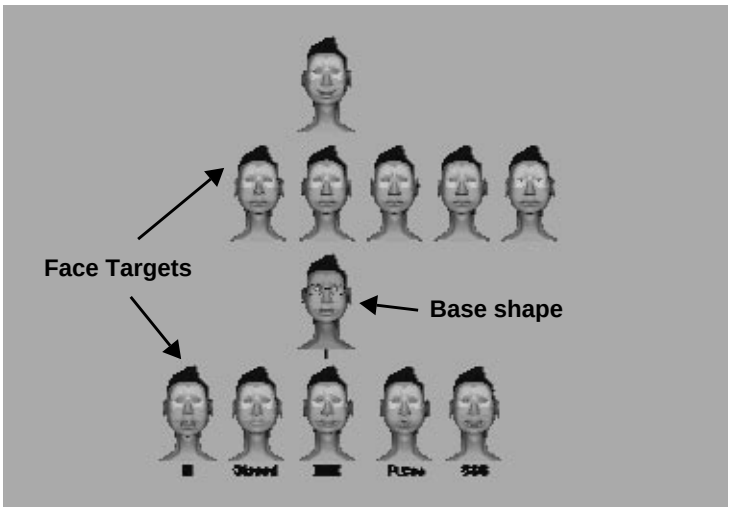
You'll create Blend Shape deformations for Melvin's face using some pre-made facial expressions that will eventually animate the face.

1 Open the scene files

- Open the file, *Melvin_16_face.mb*, from the scenes directory.

2 Import the target facial poses

- Select **File** → **Import...**, and choose the file called *Melvin_faceTargets.mb*.



Imported target shapes

3 Create a Blend Shape

- Change the pick mask to select **hierarchy**.

  This will ensure that all pieces of the heads are selected.
- **Select** all the target shape groups.
- **Shift-Select** the base group.

  This ensures that the base group is selected last.

Note: Make sure that you only select Melvin's head surfaces and do not include the eyes, glasses, teeth, and joints.

- Select **Deformations** → **Blend Shape** - ❐, and set the following:

  Blendshape Node to **facials**.

4 Test the Blend Shapes

See the blend shapes function and the controls for blending the deformations.

- Select **Window** → **Animation** → **Blend Shapes...**

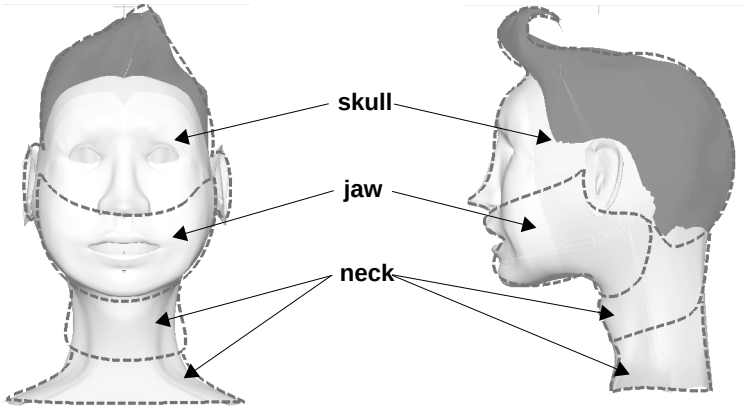
Note: It is important to note the order of the objects that are included in the head group. To blend shape between groups of objects, like this patch modeled head, these objects must be in the same order as listed in the outliner.

Joint Skin Cluster deformation of Melvin's face

Following is the method for skinning and weighting Melvin's head.You may wonder why Blend Shapes aren't used for all the facial animation or whether it is necessary to have skeletons in the head. You could use Blend Shapes exclusively for the facial poses, but combining both Blend Shapes and bound skin will give you more flexibility when creating more detailed poses and also control of Melvin's jaw through joint rotation.

1 Edit membership

- Use the following diagram as a guide to edit cluster memberships for the bound face.

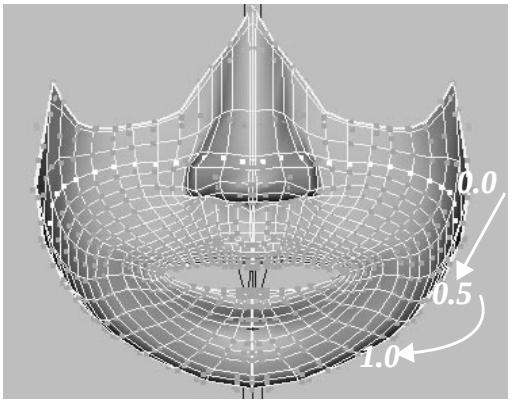


Facial clusters

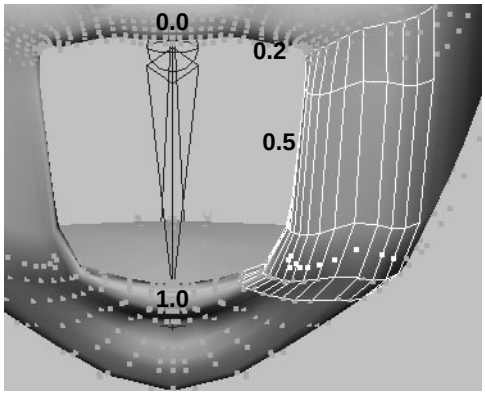
2 Weighting CVs

- Use the following diagram as a guide for weighting the CV's to achieve the proper deformation as the mouth opens and closes.

  You can manually edit CV weight for the jawCluster in the component editor or you can use Artisan to paint the weight.



Weighted CV's for mouth opening



Weights on the face

3 Use Artisan to paint weights on Melvin's face CVs

The face has already had some weighting performed on it. You will use Artisan to adjust this weight to increase the jaw joints influence across the face and smooth the deformation.

- Select the jaw joint and rotate it so that the mouth is closed.

  Notice that the cheeks are gathering and pinching.

Cheeks Creasing with Jaw closed

- Select the surface and joint to paint

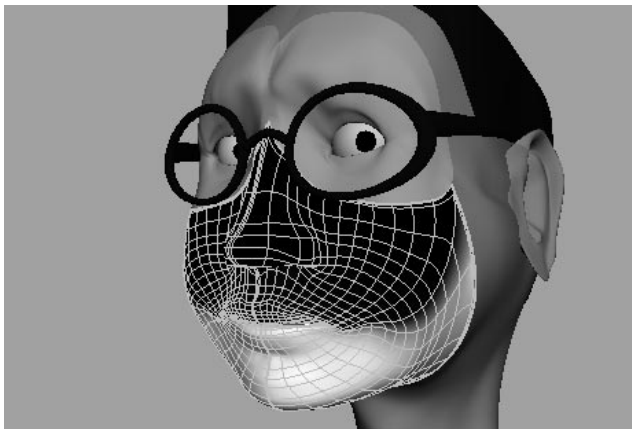
  Select the *face* surface.

Note: In the channel box you will notice what the input skin clusters are for that object. For the face surface, jawCluster1 and skullCluster1 are the input skin clusters as determined from Edit membership.

In the Outliner **cntrl** select the *jaw* joint

- **Deform → Paint Weights Tool - ❐**

  The surface will display the current weight from 0 to 1 in greyscale.



Greyscale display of weights

- Reset the Paint Weights options then set as follows:

  Weight Tab:

  Opacity to **0.5**

  Operation to **Smooth**

Stroke Tab:

Reflection to **On**

V Dir selected and value to **0.50**

- Paint the surface so that the creases smooth out.
- Test the face by rotating the jaw open and closed.

4 Add weighted influence of the jaw farther up into the cheeks

Now set the paint weights tool to scale the weight of the jaws motion farther up into the cheeks.

- Set the Paint Weights Tool settings:

  Change the **Operation** to **Scale**

  Set the **Stamp Profile** settings to:

  Radius(U) to **.325**

  Radius(V) to **.325**

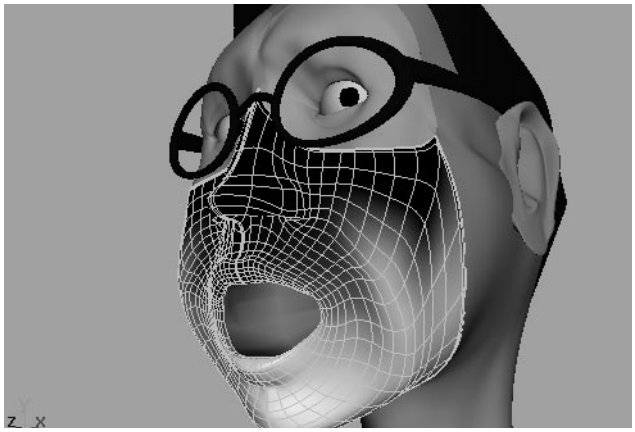
  Opacity to **1.00**

  Value to **1.00**

  Value Range:

  Paint Mult to **1.50**

- Paint higher weights up into the cheeks



Scale and Smooth weights up into the cheeks

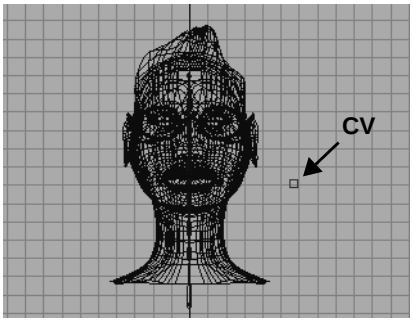
- Smooth out the values by changing the operation to Smooth and blend in the values.
- Test the operation of the jaw.

Add a slider to the blend shape control window

The blend shapes are now set up and the is face skinned and weighted. To improve animation workflow, a slider will be added to a blend shape that will control the opening of the jaw. Currently, you would have to switch between keyframing the facial blend shapes and rotating the jaw to animate Melvin.

1 Create a single CV curve

- In the front view window, place a CV beside the head In the with **Create → CV Curve Tool.**
- Rename the curve *jawOpen*.



One CV curve

2 Add the curve to the blend shape

- In **hierarchy** pick mode, **select** the *jawOpen* curve and **shift-select** Melvin's head.

- Choose **Deform** → **Blend Shape Edit** → **Add** - ❐, and set the following:

  Check Topology to **Off**

- Press **Apply**.

  A new slider is added to the blend shape. The slider will do nothing at the moment but it will be made functional using SDK to control the jaw.

3 Create a Set Driven Key between the jaw and the blend shape

- Select **Animate** → **Set Driven Key** → **Set – ❐.**
- Load the joint **jaw rotateZ** as the **driven**.
- In the Blend Shape window, click on the **select** button to make the blend shape the active object.
- Load the **facial jawOpen** blend shape as the **driver**.
- **Set** a driven key with the slider at **0** and the jaw in the default position.
- **Set** a driven key with the slider at **1** and the jaw in the open position.

4 Test the setup

- Move the jawOpen slider to see the jaw open.

  Now all the facial animation can be done from one place.

5 Save your work

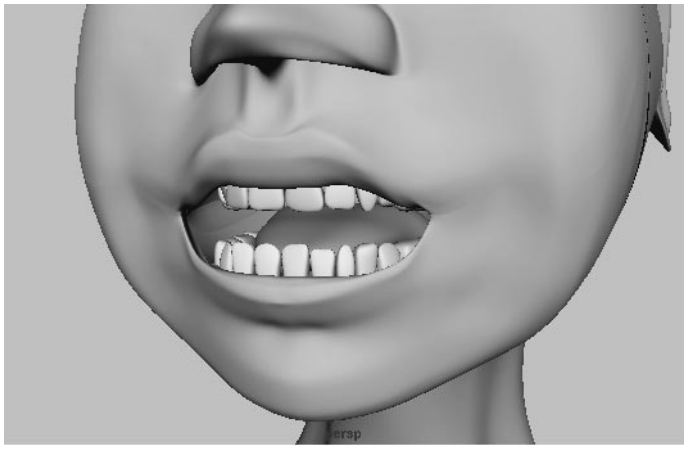
Summary

You have now completed the following tasks:

- Applied blend shape to a face
- Weighted a face using Artisan
- Added a new slider to the blend shape window and created a control for it.

17 Lip Sync

You will be applying the general controls you created for Melvin's face to effect lip syncing.



Lip syncing

In this lesson, you will learn the following:

- How to import audio files into Maya
- How to lip sync

Lesson 17

FACIAL ANIMATION CONTROLS

In the previous lesson, you set up facial controls using a combination of Blend Shapes and bind skin. The Blend Shapes provide detailed poses such as phonemes (AA, EE, OO, etc.), as well as some specialized expressions and facial muscle control. In addition to these positions, you can use the jaw joint to open, close, and shift the mouth around to deform the bound skin. Using these two techniques for deforming, Melvin can speak in sync with the audio file.

To control the motion, you have several options available:

- You can use the blend shape window and key poses or combinations of blend shapes derived from key poses.
- You can create attributes that are Set Driven Keyed to the Blend Shape mixtures. These attributes can then be controlled from the attribute editor or from your own custom UI panel.
- Combine both of these methods. While creating attributes and your own UI, you can also incorporate SDK on the relevant joints that are animated. This will provide the most accuracy and ease of use.

Facial animation techniques for lip-sync

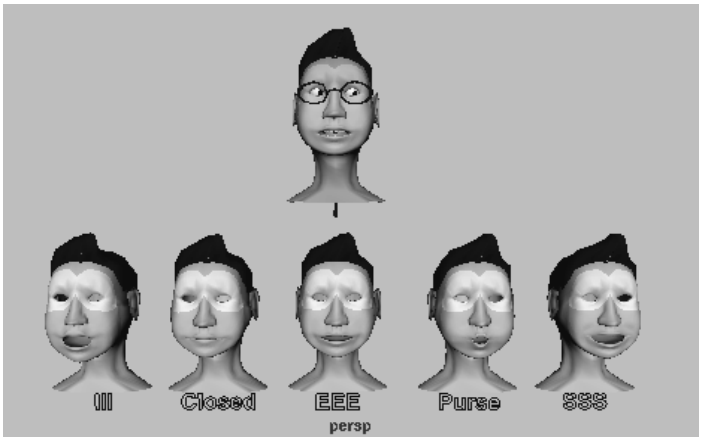
Facial animation for lip-sync can be broken down into some basic steps.

Phonetic Breakdown

The sound track is analyzed on a frame-by-frame basis and a chart of a particular sound or part of a word is made. The type of sound or phonetic interpretation is noted and aligned with its location in time.

Target shapes

Once the dialogue has been charted, the necessary phonetic shapes need to be created. These shapes will work in conjunction with jaw movements and other facial animation techniques. Below is a simple chart of the basic vowel and consonant sounds.



Basic vowel and consonant sounds

Basic keyframing

The basic timing of the character's movement will come from the sound track displayed in the timeline. You can use a simple mouth open and close to quickly establish a feel for the sequence.

Shape the mouth

Introduce Blend Shapes and other facial techniques to complete pronunciation of the syllables as charted. Accents are very important and generally will proceed the sound by several frames.

Whole face

The mouth is only one part of the speaking face. The audience will focus on the eyes if they are not distracted by bad lips and vice versa. Think of the whole head when animating.

This lesson can only provide a cursory overview of this very specialized form of animation. Some other tips include using motion study and, more importantly, a mirror to see how the face behaves when communicating. Test your animation back with sound and don't be surprised if you find your lip sync is late. The general practice is to ensure the visual action proceeds the sound, sometimes by as much as 10 frames.

DIALOGUE SHAPES

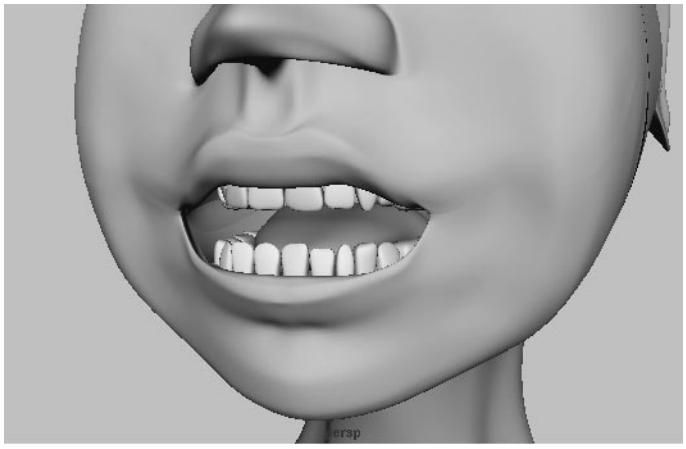
Below is a simple chart of the basic vowel and consonant sounds. Use this as a guide to the phonetic breakdown. Vowels are usually the main emphasis, with the consonants *b, f, m,* and *l* the next most important. Accents are places where the character is making a bigger and slower movement. Pay attention to these accents and address them properly with the whole face.

Vowels

The four basic vowel shapes are:

"A "and "I "

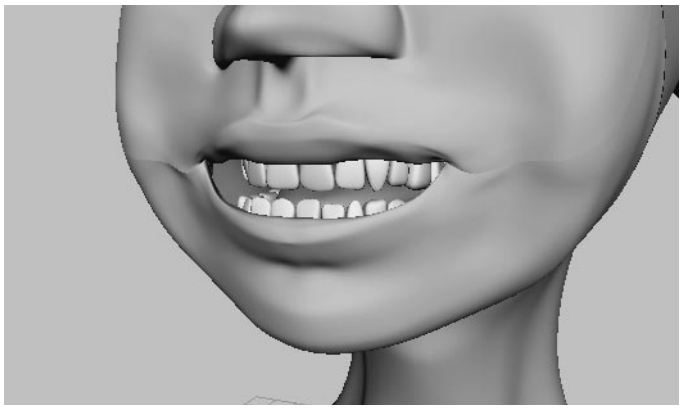
As found in words like *hey,way,day,high.*



"A" and "I" vowel sounds

"E"

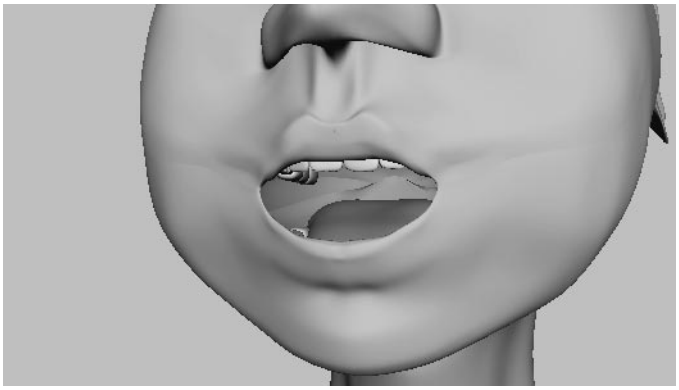
As found in words like *he, see, bee, tree.*



"E" vowel sounds

"O"

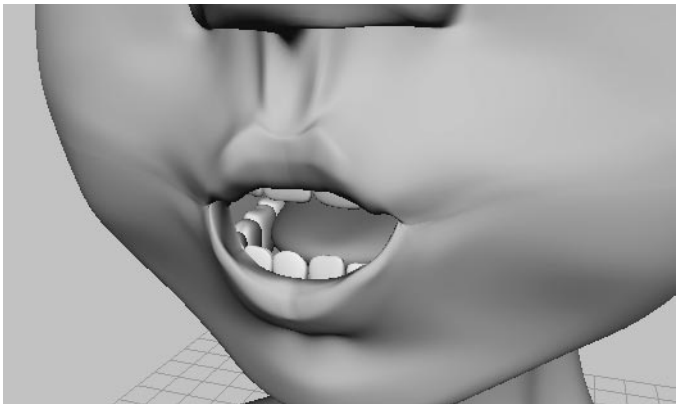
As found in words like *oh, row, crow, joe.*



"O" vowel sounds

"U"

As found in words like *you, do, stew, lou.*

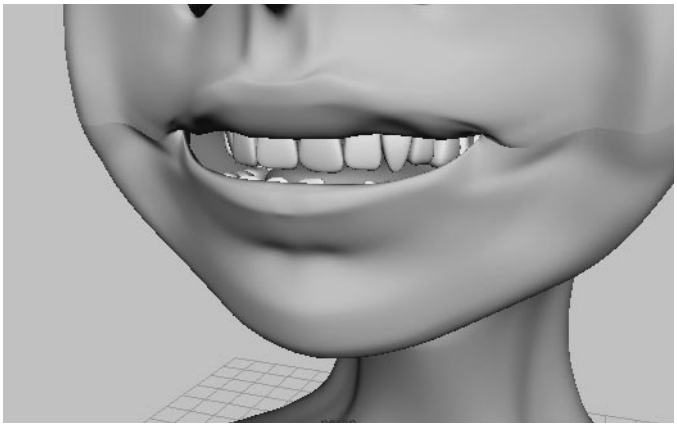


"U" vowel sounds

Consonants

The Primary Consonant Shapes are:

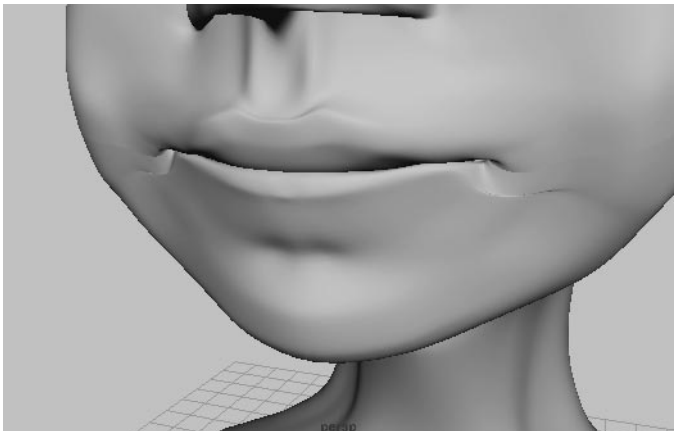
C, D, G, K, N,R, S, Y and Z



C, D, G, K, N, R, S, Y and Z consonant shapes

M, B, P

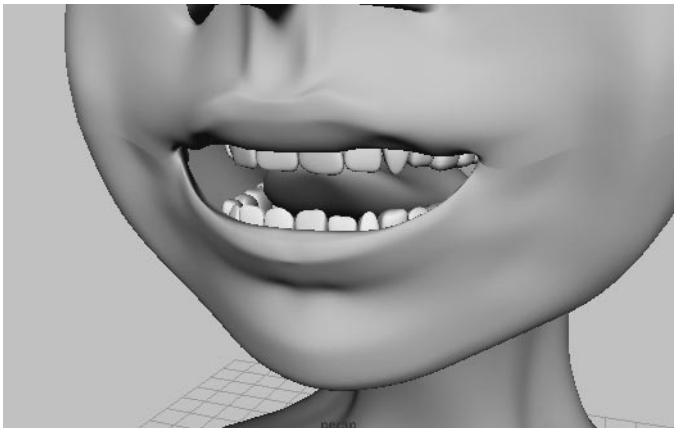
As found in words like *Mmmm, Bee, Princess*



M, B, P consonant shapes

L, “TH”, D

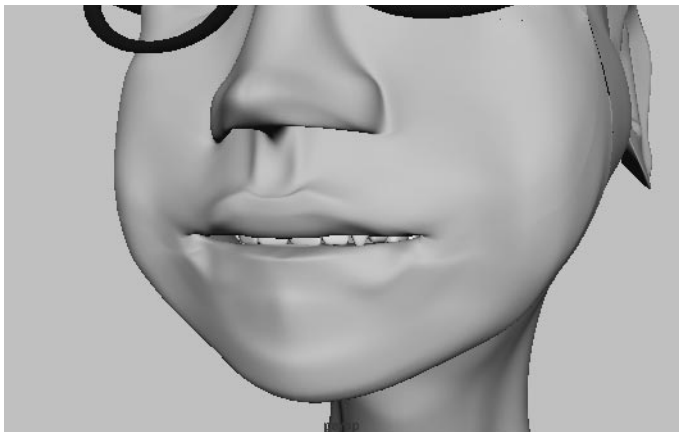
As found in words like *the, duh, love.*



L, “TH”, D consonant shapes

F and V

As found in words like *fiddle,very*.



F and V consonant shapes

From these basic shapes you can create most key shapes for the initial blocking of the sequence. The tricky part is getting these shapes to transition properly, from one shape to the next, with the sound track. You will need to study how each word is formed and also how the mouth and face prepare for a word. The “P” for example will require that the mouth close and pucker before it opens to produce. Many sounds do not come from the lips. Instead they come from the throat through an open and static mouth. This is why Lip Sync Animation does not look right on its own on a frame by frame basis. It must be viewed in real time with the sound. When in doubt, leave it out.

ANIMATING THE LIP SYNC

You will work through the steps to create some lip sync animation on Melvin.

1 Import the scene file

- Open the file *Melvin_17_faceBlends.mb*

  This scene has the default blend shapes set up for you.

2 Import an audio file

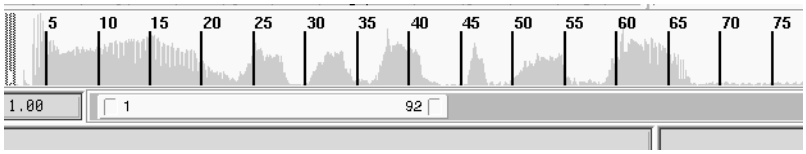
In the `~/maya/projects/melvin/sounds/` directory you will find an assortment of dialogue passages. Search this directory for an appropriate dialogue track.

- select **File** →**Import** ❐ and set the following;

  Type is to **sound**

- Click **Import**
- Load the sound file by **RMB** in the **timeline**. Select **Sound** → **SoundFileName.aiff.**

  The waveform will appear in the time line. To hear this in playback you should set the **Playback Speed** to *Normal* in your animation preferences.



Audio in the Timeslider

3 Chart the placement of the syllables with respect to the timeline

- Chart the necessary shapes to fulfill the syllables, including jaw movement and blendshape addition.
- Begin from frame one and scrub the time line to hear the sound.
- Write down the sound relative to the frame as you listen. You can use shorthand symbols to diagram and chart the relationship between sound and the time line.
- Note any shapes which are needed and have not been provided. You will need to create these shapes or work with a combination of the existing shapes.

Tip: If you are experiencing performance problems, refer to the Optimization section in lesson 10.

4 Animate the poses with respect to the imported audio file

- Derive a basic blocking of the shot from animating the jaw joint opening and closing.
- Shape the mouth and set keys on the blend shapes as they are applied. You can work in broad strokes then go back and refine or begin with first syllable and move forward as you are satisfied.

Sound Files

Maya uses sound files of *aiff* or *aifc* types. The SGI utility *mediaconvert* will convert many audio formats, including *wav*, to *aiff*.

Tip: You can key the individual blend shape sliders or use the **key all** button to set keys on all the sliders. The "key all" button generally does a better job of keying in these types of operations

5 Playback the animation as it progresses

As you work with the character it is advisable to make playblasts early on. This feedback is necessary to anticipate how the flow of the motion is occurring. You may be surprised how distracting your motion is. Avoid the temptation to apply keys on every frame in an effort to pronounce every little nuance and syllable.

Movieplayer

The SGI movieplayer may playback in a mode that does not display every frame. This allows the audio to playback synchronized, but it will drop frames from the display and not give an accurate view of the motion. If this happens, turn on the option to play every frame.

Tip: To see and hear the playback in real-time you may have to run **dmplay** from the command line. O2 systems will allow synchronized audio playback when **dmplay** is used with the hardware compression engine: dmplay -p graphics, engine=ice melvinTalks.mv should work.

6 Save your work

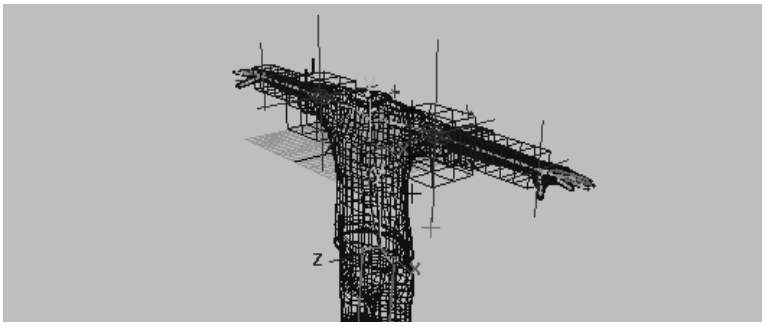
Summary

In this chapter you've learned how to:

- Import sound files
- Lip sync with blendshape

18 Attaching the Head

One important aspect of creating characters for animation is having the ability to make changes to a character *during* the production process. Changes will and do occur. Unforeseen shot requirements can creep in and necessitate changes. Typically, the character will be in a constant state of evolution. New and improved versions of the character can be substituted in place of the "working" version the animators have been using to rough in their shots.



Setting up for head replacement

In this lesson we will focus on the following areas:

- Adding Geometry to a bound object
- Point and orient Constraining a new Replacement object
- Parenting a new head onto Melvin

Lesson 18

GEOMETRY SUBSTITUTION

An important workflow to understand in Maya is geometry (or object) substitution. Currently you have two parts of Melvin that need to be integrated in one scene:

– Melvin's Head with Jaw Joint and Blend Shape setup.

– Melvin's Body that is skinned, weighted, and possibly animated.

While Maya provides several ways of integrating scene / geometry revisions, this lesson will focus on one way to put the pieces together while minimizing the amount of, or ideally avoiding, lost work.

The goal of this lesson is to expose you to a couple ideas and theories on attacking this seemingly intimidating topic because geometry substitution can become very complicated.

Geometry Substitution Scenario

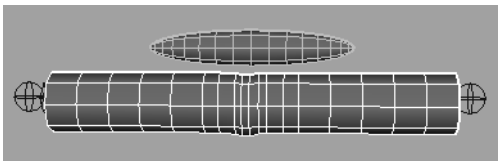
Assume that you had initially set up and animated the Melvin character that was completed after the Flexors lesson. Once production begins, another animator works on fine tuning the Head for facial animation. You give this person a duplicate copy of the skeleton and Neck / Head geometry. The other animator's task is to set up the scene file created in the Facial Animation Setup and Lip Sync chapters.

The goal is to integrate the new Head scene into the one you are currently animating. Fortunately, you have labeled and organized everything in your scene to make this process easier. Before going into one technique for combining these scenes together, you need to understand the basics of adding new geometry to a pre-skinned character.

Example - Adding new geometry to a pre-skinned cylinder

Membership and weighting are extremely time-consuming processes and cannot be discarded without careful consideration. When you replace a component, like a hand or face, you must be careful not to lose the deformation setup (weighting, membership). Redundant backups of every scene must be maintained. Do not rely on the Undo—no matter how reliable it presents itself to be. There may be times when new geometry is replacing geometry that has correct weighting and membership in place which, in many cases, can result in lost work if not done with care. In this lesson, you'll see how you can minimize the impact of substitutions on the rest of the character by detaching and reparenting skeletons while maintaining skinning groups for the rest of the character.

First you will look at simply adding un-skinned geometry to deform with skinned geometry. In the following example, a cylinder is skinned to a joint chain. The goal is to skin a sphere so that it is also deformed by the joints.



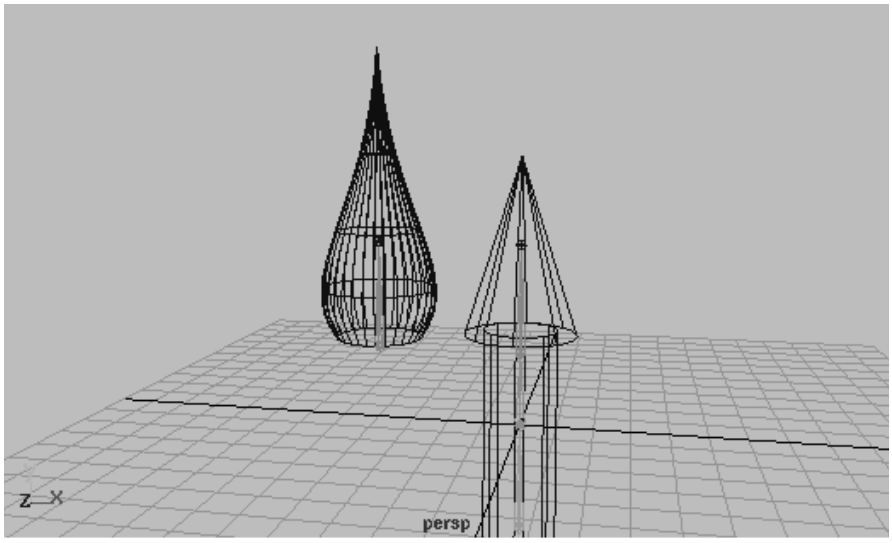
Adding a piece of geometry to a skin

Two Techniques to skin the sphere to the joints:

- Use the **Deform→ Edit Membership Tool** to add CVs from the sphere to the correct joint. In some cases this may be the best choice, especially if the sphere is already skinned to other joints. Editing membership of skinned surfaces only works on rigid bound geometry, while you would have to bind the new geometry to selected joints if you were working on smooth skinned geometry. Realize that if you use this technique in areas where there will be a lot of deformation, such as in the area of the elbow, the deformation could be harsh due to all the CV weights being the same.
- **ReBind Skin** - by selecting the sphere and the joints, you can bind the sphere so it will be included in the earlier skinning.

Example - Point and Orient Constrain a new arrowhead

A slightly more complex example would be similar to attaching Melvin's head. What if our neck and back joints were previously animated, *and* you needed to integrate a new head with a new jaw joint and blend shapes? The goals would be to 1) keep the animation and 2) replace the original head with the new. A simple example could be as follows. Below there are two arrows: both are bound to joints and one is animated. The ask involves removing the first arrowhead and replacing it with the new arrow.



Simple substitution

The scene file *demo_arrowHead.mb* is provided in the demo directory as a starting point for this exercise.

1 Open the file

- **File** → **Open** `~/maya/projects/Melvin/scenes/demo/demo_arrowHead.mb`

Tip: Use the Layer Editor to display and hide the different versions of the arrow heads.

---

2 Use Point and Orient constraints to attach the new head

- Select the existing *neck* joint then **shift-select** the corresponding joint on the new skeleton (*newHeadRoot*).
- Select **Constrain** → **Point**.
- Select **Constrain** → **Orient.**

3 Hide or delete the old head

This technique assumes that the joints in the two scenes are in the same place. If they were not in the same place you could add a group hierarchy to the respective joints and constrain these groups as offsets.

Corrective Blend Shape option

If both files had heads of similar topology and animation, you could use a corrective blend shape to force the original head to inherit the deformations of the newer head. Don't forget to consider deformation order. Also, don't forget to use the world origin option in the **Deform** → **Create Blend Shape** – ❐.

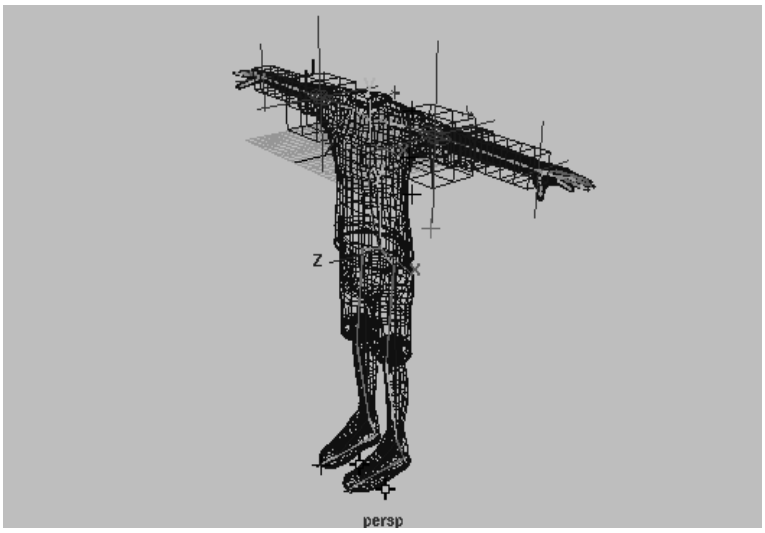
REPLACING MELVIN'S HEAD

In this workflow you will parent the new lip sync animated head to an existing Melvin body prepared for the operation. This could be a scene that has animation on it, but you will work as if it has been returned to bind pose while the new head is being prepared.

1 Open the scene file

- Open the file *Mlevin_19_headless.mb.*

  This scene file is the completed version of Melvin without his head. His head geometry and corresponding joints that will be replaced have been deleted.



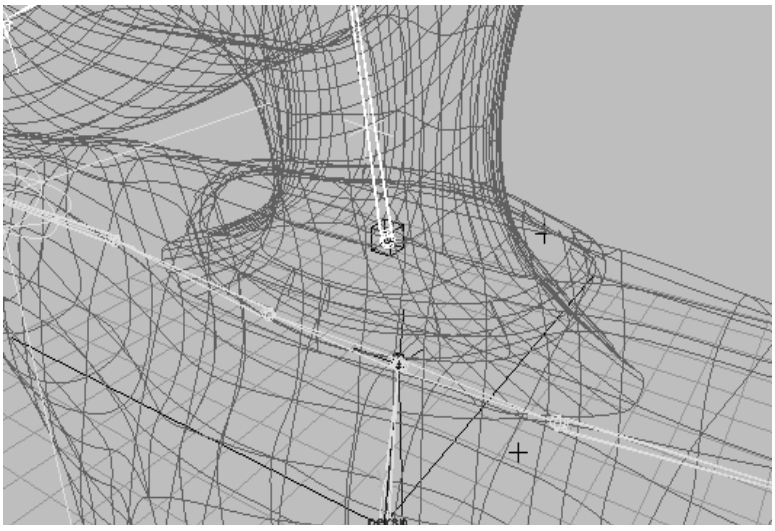
A head will be attached to this body

2 Import the Lip Sync Scene

Import the Lip Sync Scene you have created or the scene file demo_melvinTalks.mb in the ~/maya/projects/Melvin/scenes/demo directory

- Select **File** → **Import...** *demo_melvinTalks.mb.*

3 Return both Melvin hierarchies to their bind pose if necessary



These skeletons must be combined

4 Parent the root of the head joint to the skeleton

- Open the Hypergraph.
- Locate the root joint of the head skeleton hierarchy and the *back_shoulderJoin*.

Tip: Nodes can be reordered in the hypergraph by using **Ctrl MMB** to drag and drop nodes into place.

- Drag and drop the head with **Ctrl MMB** onto *back_shoulderJoin*.

  The results from this will be different than using the **parent** command. Parent will insert a group node into the hierarchy to preserve the position of the clustered CVs, which could complicate membership editing.

Reminder

Red is the default color representing CV's or points that do not belong to any group.

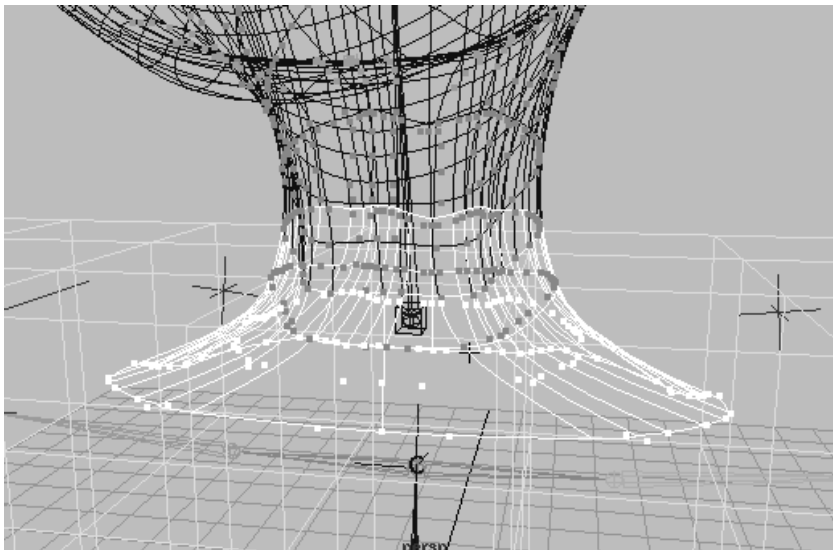
5 Transfer the CVs from the neck joint to the shirt lattice

The original Melvin uses a lattice for skinning the torso and the base of the neck, so you will need to edit the membership of the neck's CVs to ensure they are deformed by the shirt lattice and not the base neck joint.

Using the Edit Membership tool, transfer the membership of the CVs currently belonging to the bottom neck joint to the shirt lattice set. You may want to refer to the lesson where Melvin was originally bound to the skeleton so that you set the membership up the same way.

- Display the *shirtLattice* if it is hidden.
- Hide the shirt and collar geometries to aid in selection of the neck CVs.
- Select the **Deform → Edit Membership Tool.**
- Select the *back_shoulderJoin* joint to see what CVs it deforms.

  You should see that it deforms a group of lattice points, and that some of the lattice points do not belong to any joint.
- Add these lattice points to the *back_shoulderJoin* joint.
- Select the *back_neckJoin* joint and note that the new neck geometry belongs to this joint.



Edit membership of the neck

- **RMB** click on the neck geometry and select **hulls** from the marking menu.
- **LMB-ctrl** select the bottom two hulls to remove them from this membership.
- **Select** the shirt lattice.
- **Shift-select** the two bottom hulls to transfer the membership to the lattice.
- Press **q** to exit from the Edit Membership Tool.

6 Add lattice points to the neck

There are a couple of lattice points that will be red because they are not currently a member of any cluster. Use the **Edit Membership Tool** to add them to a neck cluster.

7 Test the membership

- Test the membership by translating Melvin from his root.

  If you have made an error you should see either CVs that double translate or orphans that get left behind. Add them to the appropriate cluster.

8 Save your work

Summary

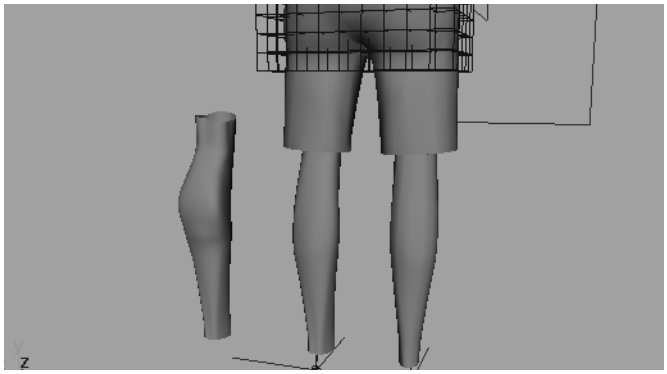
In this lesson you've learned how to replace parts of an animated character. This is not always part of the animation process but is useful to know in case changes are be required.

Appendix A

A Blend Shape

Blend Shape is deformer type that lets you use several target shapes to drive a base shape. The tool creates controls for each of the target shapes that can be edited and animated to control the look of the base shape.

In this lesson, you will use Blend shape to add some bulging to Melvin's calf. You will duplicate his existing leg then sculpt the new surface to show the bulging. Blend shape will then be used to create the blended deformation between the bulge target and the original leg. The value of the blend can then be animated using Set Driven Key so that the rotation of the knee joint animates the bulging.



Blend Shape

In this lesson, you will learn the following:

- How to set up a Blend Shape muscle bulge
- How to work with the deformer order
- How to use Set Driven Key with a blend shape

BLEND SHAPE

Blend Shape is a powerful morphing tool. You can build several shapes and use blend shape to link them together. Blend Shape will create sliders so you can have smooth transitions between shapes after you put keyframes on the sliders.

In the most basic sense, a Blend Shape consists of two things: the *target* shape(s) and the *base* shape. It's easy to understand the basics of the Blend Shape by examining how Maya executes the Blend Shape function.

When executing a Blend Shape, Maya looks at the changes between the base and the target shapes. The value of the Blend Shape (which ranges from 0 to 1) is a percentage of how much of that difference will be added to the base shape.

Make a calf muscle bulge

This is the procedure to make the calf muscle bulge.

1 Open an existing file

- Select **File** → **Open.**
- Select the file *Melvin_calfNoBulge.mb*

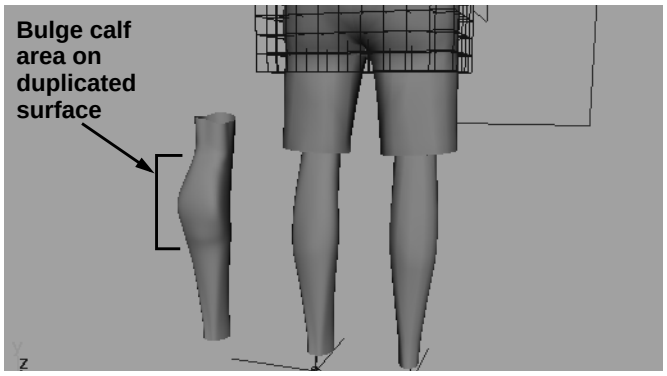
2 Duplicate the lower left leg

- Select **Edit** → **Duplicate.**
- Rename the new leg *calfBulge.*

3 Model the calf bulge

- **Select** the leg.
- Press **F8** to go into component selection mode.

  Make sure that the selection mask is set up to only pick CVs.
- Pull CVs on the leg surface to make the calf appear bulged.



Bulged calf

4 Create the blend shape

- Press **F8** to go back into object selection mode.
- Select the *calfBulge* geometry followed by the original leg geometry.

Tip: When defining a blend shape the base shape must be selected last.

- Select **Deform** → **Create Blend Shape** - ❐. In the option window, set the following:

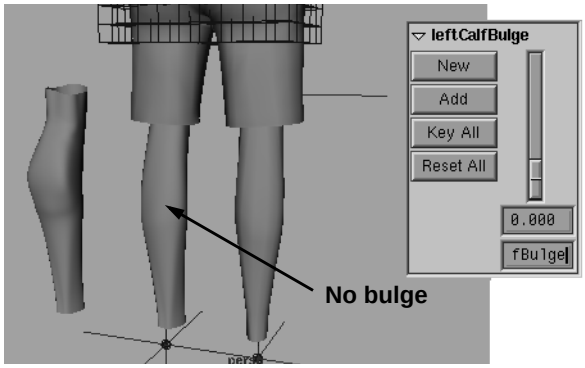
  Name to *leftCalfBulge.*

- Press the **Create** button.

5 Test the Blend Shape

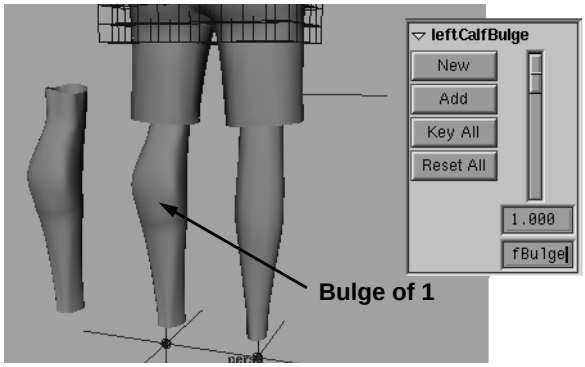
- Select **Windows** → **Animation Editors** → **Blend Shape...**

  This will open up the Blend shape editor which has a slider for controlling the bulge of the calf.



Blend shape editor with bulge of 0

- Move the vertical slider up to 1 and the calf will bulge.



Blend shape editor with bulge of 1

Tip: You can also access the blend shape by clicking on the *leftCalfBulge* input node in the channel box and changing the value of the *calfBulge* field. This attribute is accessible through the Attribute Editor as well.

This method for a muscle bulge is quite effective. It not only lets you control the amount of bulge but lets you create a very specific shape. For example, an arm will bulge differently if it is lifting a pencil vs. a computer monitor.

Blend Shape Options

Listed below are some of the options available on the Blend shape tool.

Envelope

This is a scale factor for the slider. A value of 2 will double the effect of the slider, 0.5 will make it half and -1 will invert the effect.

Origin

This can be either *Local* or *World*. Local means the change takes place relative to the base shape. World will cause the blend to move from the position of the base to the position of the target.

For blend shapes of this nature you will generally want to use *Local*. You can compare the effects in the Attribute Editor.

In-Between

In-Between will chain targets together. If it is set to on, a slider could change from a frown, to a face at rest, to a smile. If the smile were the base, the selection order would be frown, default and then smile.

(Cont'd on next page)

Blend Shape Options

(Cont'd)

Check Topology
Check Topology *off* will allow you to blend shapes with different numbers of vertices. When turned *on*, it will return an error stating that the shapes have a different vertices count.

Delete Targets
Delete Targets will delete the targets after the blend shape has been created. Once created, the targets are not needed, so this can result in improved performance.

It is a good idea to save a copy of your targets somewhere in case you need to use them later to make adjustments.

Deformer Order

It's important to be aware of deformer order. Notice what happens when the knee is bent and blend shape is set to 1.0. The knee will 1) deform the leg to a bent position and then 2) the blend shape will deform the leg back towards its original position with a bulged calf. If you switch the deformation order around, so that the blend shape is executed first, the leg will still get bent, but it gets bent *after* the blend shape deforms the leg.

Try this with your newly setup leg:

1 Rotate the knee backwards

- Set the blend shape back to **0**.
- **Rotate** the knee backwards.

  You may need to disable IK Handles

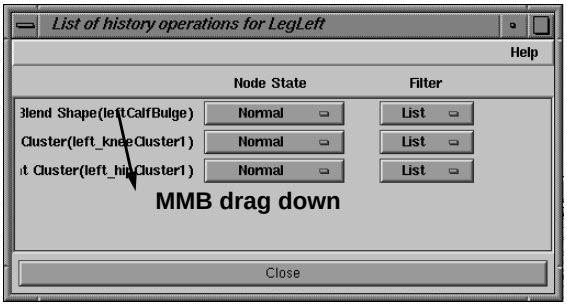
2 Bulge the calf

- Move the vertical slider up to 1 and the calf will bulge

  You should see the leg geometry move off the bone and into the position that the blend shape was created in. This happens because the clusters from the bind skin operation affect the leg and then the blend shape. This should happen in the opposite order.

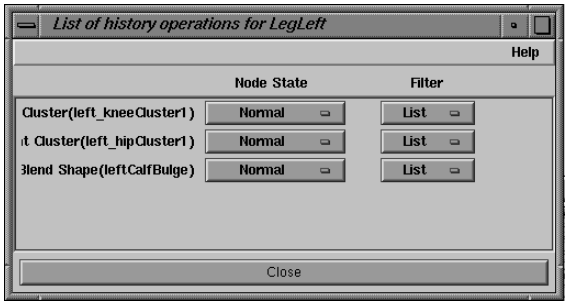
3 Reorder the deformers

- With the **RMB**, click on the leg geometry and, from the popup menu, select **Inputs → Complete List.**
- This opens up a window that shows you a list of the deformers currently affecting this surface.



List of deformers

- Use the **MMB** button to drag the *Blend Shape* to the bottom of the list. This reorders the deformers.



Reordered deformers

4 Bulge the calf

Now things should be working as expected.

Blend Shape Integration

The Blend Shape control window is a good tool for controlling the deformation, but you don't want to have to go there every time you want the corrective blend to occur. The window is designed for more complex blends like facial animation, and isn't designed to be used effectively in cases like the calf bulge.

You will integrate the bulge with the rotation of the ankle using Set Driven Key. To drive the *blend shape* with SDK, you'll do the following:

1 Load the Driver

- Open the Set Driven Key window with **Animate→ Set Driven Key → Set - ❐.**
- **Select** the *left_ankle rotateZ* attribute.
- Press **Load Driver**.

2 Load the Driven

- Press **Select** in the blend shape window to make it the active object.
- Select the *leftCalfBulge* attribute as the **Driven** object.

3 Set a driven key in the unbent position

- With the *left_ankle rotateZ* and *leftCalfBulge* attributes selected press **Key**.

4 Set a driven key in the bent position

- Rotate the *left_ankle* in Z to its maximum bend position.

Note: If you have not disabled IK solvers you will not be able to bend the knee.

- Set the *leftCalfBulge* to a value of 1.
- With the *left_ankle rotateZ* and *leftCalfBulge* attributes selected, press **Key**.

Blend Shape Control

Listed below are some of the options available in the Blend shape Control window.

Select

This will select the blend shape so that it can be controlled through the channel box or manipulated in the Dependency Graph.

Tip: To delete a blend shape, Select it with this button and then delete it.

Key

This will allow you to key the current value of the slider at the current frame.

New

This will create a new blend shape much like using the **Deform → Edit Blend Shape - ❐.** Pick the targets, followed by the base, and press the **New** button.

New targets can be added to an existing blend shape with **Deform → Edit Blend Shapes → Add - ❐.**

5 Test the ankle rotation

- As the ankle rotates you should see the calf bulge.

6 Save your work

Summary

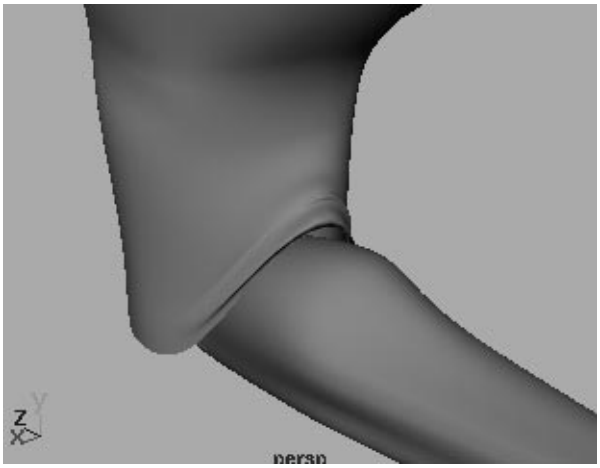
You have now completed the following tasks:

- Set up blend shape
- Re-ordered the evaluation of deformers
- Used SDK to control the effect of a blend shape

Appendix B

B Wire Deformers

Wire deformers can be used in many ways, such as making waves on an ocean or creating wrinkles on a face. In this lesson, the *wire deformer* and *blend shape* tools will be used to further enhance the behavior of Melvin's clothing. This will also correct the problem of Melvin's shorts intersecting his legs.



Creating wrinkles with the Wire deformer

In this lesson, you will learn the following:

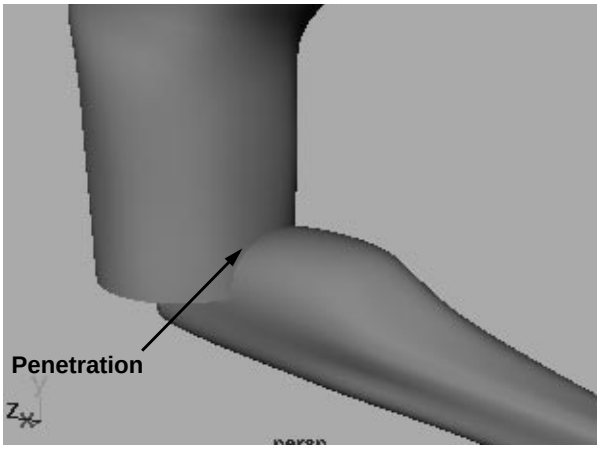
- How to use wire deformers
- How to use blend shape to deform curves

WIRE DEFORMERS

Wire deformers are curves that influence a surface. A relationship is defined between a curve and a piece of geometry. As the curve is transformed, or its shape changed, it will be reflected on the surface.

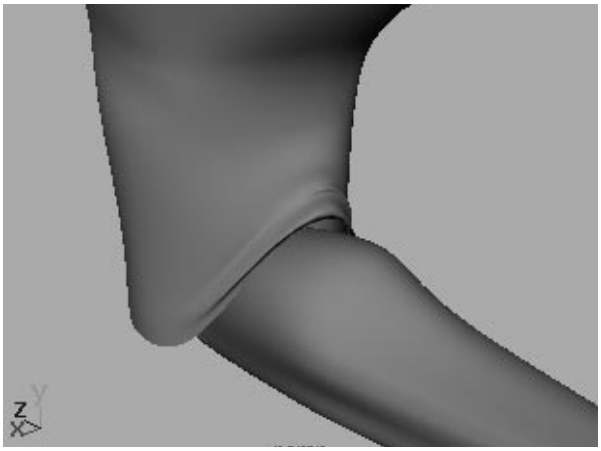
Deforming Melvin's shorts cuff

When Melvin bends his leg at the knee, he will eventually bend it far enough to cause his calf to intersect with the cuff of his shorts. One way to fix this is to deform the cuff of the shorts with a wire deformer so the cuff bunches up where it meets the back of his leg.



Penetration problem

The wire deformer can then be changed with a blend shape and controlled with SDK so that the shorts are affected gradually as the knee bends.



Corrected wrinkle

Create a Wire Deformer on the Shorts

The first step in this process is to create a wire deformer for the shorts.

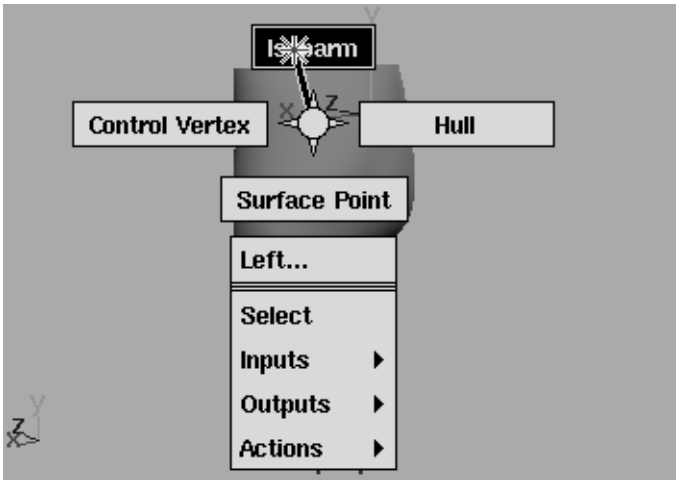
1 Open an existing file

- Select **File** → **Open.**

- Select the file *Melvin_calfBulge.mb*

2 Duplicate a curve from the shorts

- In the viewport or the Outliner, select the shorts_left_back_bot surface.
- Hide all other surfaces.
- From the **RMB marking menu**, select **Isoparm.**
- **LMB-drag select** an Isoparm that is running (vertically) up the back of the shorts.



Right Mouse Button Marking Menu

- Select **Edit Curves** → **Duplicate Surface Curves.**

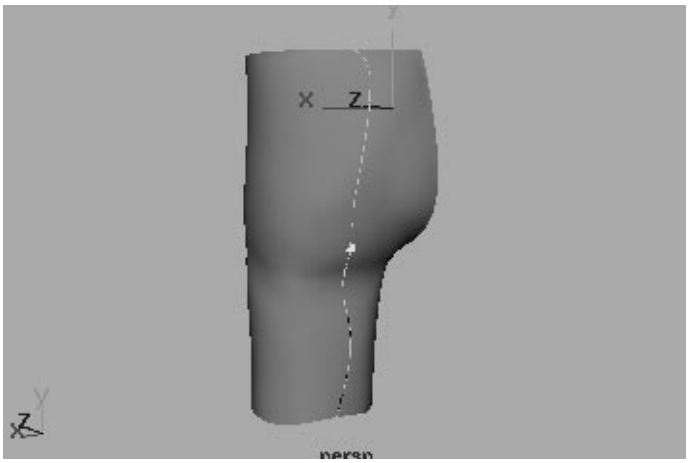
3 Detach the curve

Only the bottom of the shorts need to be affected by the wire deformer, so the curve just needs to be as long as that region.

- **Select** the duplicated curve.
- From the **RMB marking menu**, select **Curve Point**.

Tip: It may be easier to select this curve point by either temporarily hiding the shorts surface, or using the pickmask to make curves selectable and surfaces not.

- Drag a curve point about 1/3 of the way up from the bottom of the shorts.



A curve point on the duplicated curve

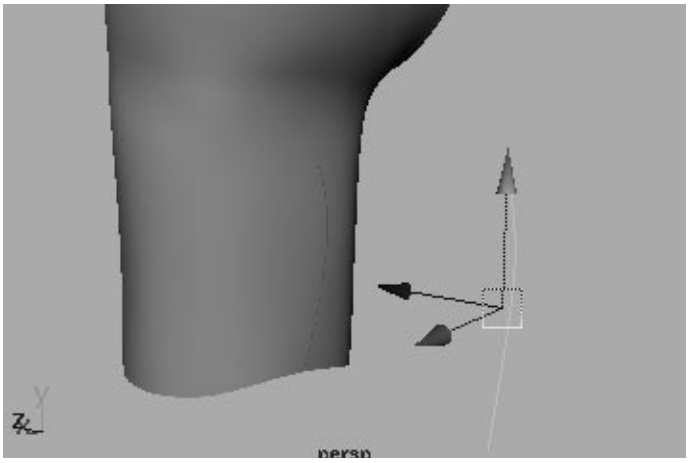
- Detach the curve at the selected curve point with **Edit Curves → Detach Curves.**

  This will detach the curve and create two curves from the curve point that was selected.
- **Delete** the upper half of the curve and
- Rename the lower curve *wireCurveBase.*

4 Duplicate the curve

Since you will use blend shape later to change the shape of the curve, it should be duplicated so that you have an untouched curve to work with.

- **Duplicate** the curve *wireCurveBase* and move it (backwards) behind the shorts.
- Rename the curve *wireCurveTarget.*



Moving the wireCurveTarget behind the shorts

5 Delete history on the curves

The curves were created from the surface. You must delete the curves' history so that they are not inadvertently affected by any changes made to the surface.

- Select *wireCurveBase* and *wireCurveTarget*.
- **Edit → Delete by Type → History.**

6 Create a Wire Deformer

- **Select** the left shorts surface.
- Select **Deform → Wire Tool.**
- Press **Enter** .

  This will define the left shorts as the surface where the wire will be added.
- Select *wireCurveBase* to define it as the wire that will affect the shorts and
- Press **Enter**.

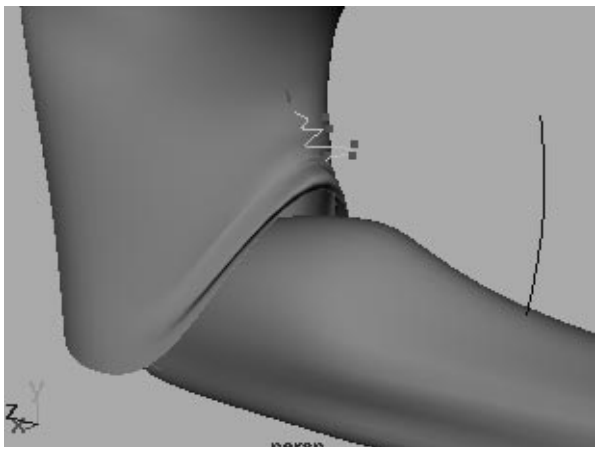
  The curve is now a wire for the shorts surface.

7 Change the wire shape

- **Rotate** the *left_knee* in Z to Melvin's maximum bend position.

Note: You may need to disable IK in order to rotate the knee.

- Unhide Melvin's left leg to help shape the wrinkle.
- **Select** *wireCurveBase* and press **F8** to change to component mode. You may want to turn hulls on in the pick mask as a visual guide.
- **Move** the CVs up in Y and back and forth in Z to create a bunched up look as shown below. X-Ray mode may help while you do this.



deforming the wire

As you move the CVs you should see the surface change, so that you can get a better idea of how to shape the curve. You can also select multiple CVs and scale them. You can use the up and down arrow keys' *pickwalk* function to walk through the individual CVs.

8 Refine wire attributes

You may find that the wire is not affecting a broad enough area of the surface. You can adjust wire specifics in the Attribute Editor.

- **Select** the *shorts_left_bot_back* surface.
- Open the Attribute Editor and select the *wire1* tab.
- Increase the value of the **Dropoff Distance** to have the wire affect more of the surface.

9 Reorder Deformers

Since Melvin was bound to the skeleton before setting up this wire deformer, the deformer order needs to be changed.

- With the **RMB**, click on the shorts geometry to get the popup menu and select **Inputs** → **Complete List.**
- Use the **MMB** to drag the wire down in the list so that it is evaluated before the FFD.

10 Save your work

Controlling a wire deformer with Blend Shape

Now that the wire deformer is applied to the shorts, a blend shape can be added so that the shorts can easily be changed from non-deformed to the deformed version. SDK will also be added so that bending the knee will control the blend shape.

You will continue working with the same file:

1 Create a Blend Shape with the two curves

- Select the *wireCurveTarget* first and the *wireCurveBase* curve second.
- Select **Deform** → **Create Blend Shape -** ❐**,** and set the following:

  Blendshape Node to **shortsCuff**

2 Drive the blend shape with Set Driven Key

- Select **Animate**→ **Set Driven Key** - ❐.
- Select the *left_knee* joint and load it as the **Driver**.
- Select the *ShortsCuff* blend shape node from the Blend Shape Editor and load it as the **Driven**.

3 Set a driven key in the unbent position

- Select **left_knee RotateZ** and **shortsCuff wireCurveTarget** attributes.
- Verify the **wireCurveTarget** value is set to **1**.

- Press **Key**.

4 Disable the IK handles to rotate the left knee

Since the left knee rotation will be driving the wrinkle, the IK handle needs to be disabled, otherwise it will not rotate.

- Select the *L_ankleIK*.
- Select **Skeleton** → **Disable Selected IK Handles.**

5 Set a driven key in the bent position

- **Rotate** the *left_knee* around Z to Melvin's maximum bend position.
- Set the **wireCurveTarget** to a value of **0**.
- Press **Key**.

6 Test your knee rotation

As the knee rotates you should see the shorts bunch up.

7 Save your work

Summary

You have now completed the following tasks:

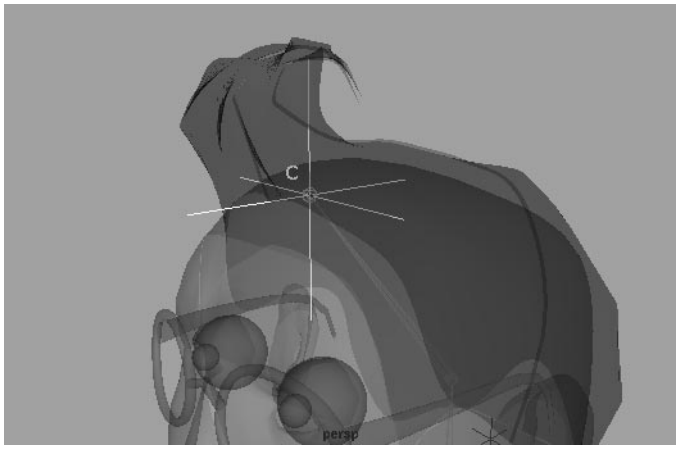
- Created a wire deformer by extracting a curve from an isoparm
- Created a Blend Shape
- Controlled the wire deformer with actions from other objects using SDK

Appendix C

C Secondary Motion

Maya provides several ways to give your character's skin a more realistic look. You have already used flexors to get some bulging and tucking at the joints. To further enhance Melvin's animation, you will create *secondary motion* or independant elements that carry their own mass and relative movement. This can give more life to your character's clothing, hair, muscle and parts that jiggle.

This lesson will use a cluster to make Melvin's hair bounce as he walks, derived from Melvin's own motion.



Creating secondary motion for hair

In this lesson, you will learn the following:

- Review cluster deformers
- Get translational data from a point on the character
- Use Bake Simulation to get animation curve data
- Copy and paste keys
- Edit keys in the Graph Editor

Cluster Deformer for Melvin's Curls

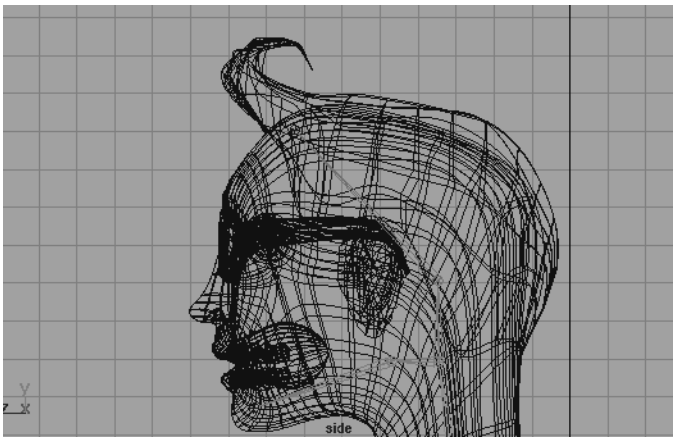
Once the main animation is laid out for the character, you can cluster the parts of the character's skin that you want to jiggle. Based on its current motion, you can add extra motion by keyframing the clusters.

In the following exercise, you will import a scene where Melvin is walking in a treadmill-type cycle. The goal of this exercise is to make the curls on top of Melvin's head bounce around as he walks. First you will create and weight a cluster deformer around the curl of hair, then animate its *translateY* to give the effect of the bounce.

1 Open an existing file

- Open the file *16.Melvin_walk.mb.*

  This opens a scene with Melvin walking.
- With the side view active, playback the scene to preview how Melvin is bouncing. The active frame range should be set from 1 to 30.

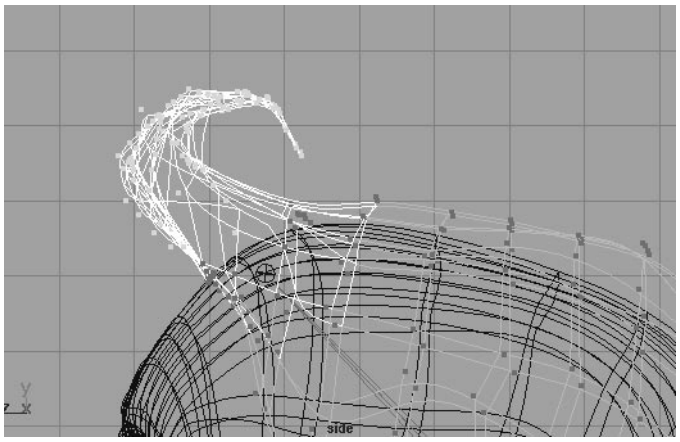


Preparing to animate Melvin's curl

2 Create a cluster of CVs around the curl of hair

Note: There are two sets of hair in this scene. One set is for the low res. version of Melvin that you use to animate interactively and it is parented to the skeleton. The other is part of the high res. Melvin Skin that is skinned to the skeleton. The following steps are to be used with the high-res layer.

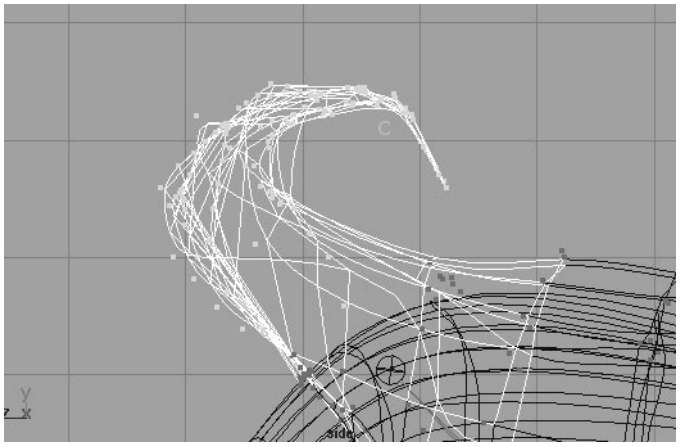
- In Component mode, **Select** the CVs that make up the curl of hair on the left and right hair surfaces.



Selected CV's for the cluster

- Select **Deform** → **Create Cluster** - ❒ and set the following:

  Mode to **Relative**.
- Press **Create.**
- Rename the cluster *curlCluster*.

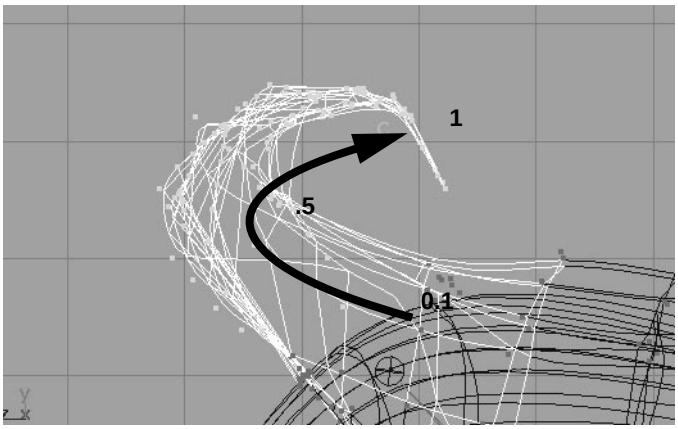


Clustered CVs

New Cluster

3 Taper the CV weights

Using the Component Editor, you will taper the weights of the CVs so that they are.5 in the middle, 1 at the tip and 0 on the scalp. You will also move the cluster over a couple of units to make it easier to preview your weighting.



weighting cluster

- To make it easy to see the effects of the weighting, **Move** *curlCluster* in the z-axis about 2 units
- In Component mode, **select** the scalp section of CVs that are being deformed by *curlCluster*.
- In the Component Editor, set their weights to **0.1**.

  Now they are deformed by 10% of the total cluster deformation.
- Continue selecting horizontal rows of CVs to taper the weights ranging from the .01 (at scalp) to 1.0 (at the tip) so that there is a smooth transition of deformation.
- **Move** *curlCluster* back to the origin (0, 0, 0).

4 Parent curlClusterGroup to the skull joint

When you play back the animation, you will notice that the *curlCluster* does not move with the head. You will need to parent *curlCluster* to the skeleton. **Select** *curlCluster* and **Shift-Select** the *skull* joint node.

- Press the **p** key to parent them.

  The *curlCluster* is grouped so that it will not be directly affected by movement of the skeleton.

Tip: To quickly find the skull joint or any other node; you can select the objects in the viewport then use **Show** → **Selected** in the Outliner, or **View** → **Frame Selected** or **Frame Branch** in the Hypergraph. Both of these are available by your right mouse button as well.

ANIMATING THE "JIGGLE"

To make the curl look like it bounces, the curlCluster needs to animate whenever Melvin's head changes direction. You will generate the *curlCluster's* motion from Melvin's head movement.

Melvin's head movement is the result of keyframed animation on several components including the *back_root* joint and the splineIK clusters, as well as the neck joints. Instead of generating an animation curve from these

separate components, you will extract the world space movement of the top joint in the head(*skull* joint).

Extracting the World Space Location of a Joint

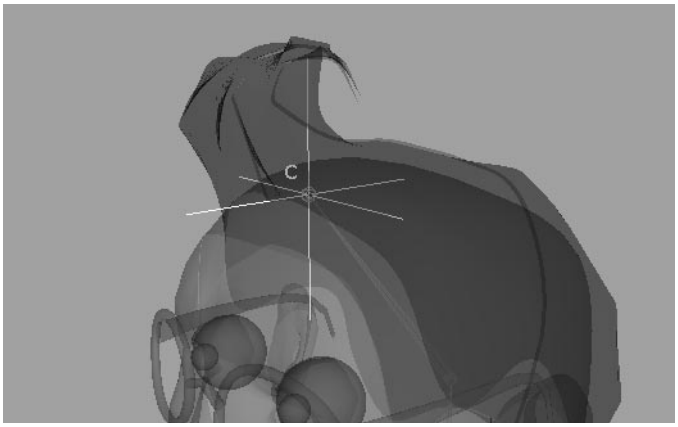
Since the *skull* joint is a child of the skeletal system, it does not have information relative to world space or the origin. This can be overcome by constraining a Locator to that joint and baking out keyframes on that Locator.

1 Create a Locator

- Select **Create** → **Locator.**
- Rename this Locator *trackLocator.*

2 Move the Locator to the skull joint

Move the *trackLocator* to the *skull* joint by using the **v** key to snap to points.



Locator snapped to skull joint

3 Freeze Transformations on the Locator

You need to freeze the translations on the *trackLocator* so that its animation curves are not offset from the *skull* joint.

- **Select** *trackLocator* and then select **Modify** → **Freeze Tranformations.**

  You may want to scale the *trackLocator* up so that it is easier to pick.

4 Point Constrain the Locator to the skull joint

- In the viewport select the *skull* joint and **shift select** *trackLocator.*
- Select **Constrain** → **Point**.

5 Bake the Keyframes for the Locator

You can get the world space data for the *trackLocator* by using the **Bake Simulation** tool. This tool takes objects that are not animated and creates keyframes at the desired resolution.

- Select *trackLocator.*

Bake Options

By Time controls how often a keyframe is created. The default value is set to 1.00, which creates a key at every frame.

Channels controls where keyframes will be created. If you select channels in the channel box (using the From Channel Box option), curves will only be created for the channels you've selected.

- Select **Edit** → **Keys** → **Bake Simulation** - ❐, and set the following:

  Press **Reset** to ensure you begin with the default settings.

  Time Range to **Time Slider** (check that your time slider is showing frames **1** → **30**)

  Alternatively, you could select **Start/End** and enter **Start** and **End** frames of **1** and **30**.

- Press **Bake.**

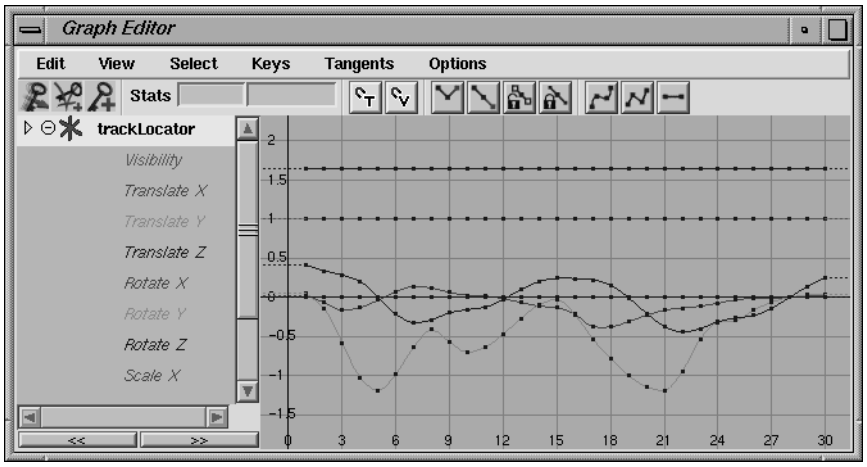
  The Bake Simulation creates animCurves with keyframes at every frame for each keyable channel.

Working with the extracted data

You now have animation curves which describe the motion of the skull joint. You will be removing unnecessary animation curves from this joint before applying it to the *curlCluster*.

1 View the curve in the Graph Editor

- Select the *trackLocator*.
- Open the Graph Editor and **RMB View** → **Frame all** to see the curves that have been created.



The trackLocator channels

2 Delete static channels

The *trackLocator* will have animation curves for every channel, but the only channels with animation are *Translate*. The static channels have keyframes but each keyframe is the same, so these can be deleted to clean up the graph editor.

- Select **Edit** → **Delete by Type** → **Static Channels.**

3 Copy and Paste the Keys from the Locator to the curlCluster

- **Select** the *trackLocator*.
- In the Graph Editor, Select **Edit** → **Copy.**
- Select the *curlCluster*.

- In the Graph Editor, Select **Edit** → **Paste.**

4 Test the motion

The cluster should now be animated and this can be verified by playing it back in the timeline.

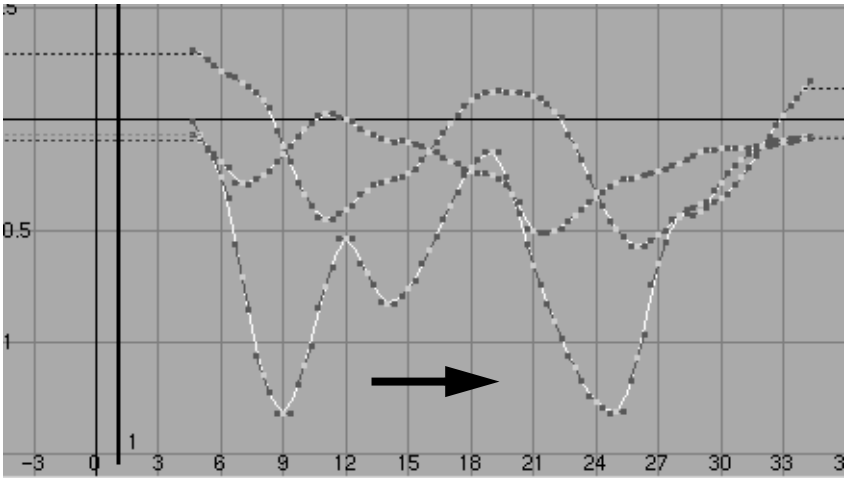
Offsetting the motion

The motion of the *curlCluster* should be secondary to that of Melvin's head. The hair should react to Melvin's head as if it has its own mass. This can be achieved by offsetting the *curlCluster* translation curves.

1 Translate the keys to create relative secondary motion

- Select all of the *curlCluster* Translate keys.
- Turn on **Time Snap**.
- Use **Shift -MMB** and drag to the right to move the keys 3 frames.

  This will make the *curlCluster* lag behind the head as if it is following that motion.



Moved keyframes

2 Cycle the curves

Since this animation of Melvin is on a cycle, the curlCluster should also be on cycle.

- With the curves selected, **Curves** → **Post Infinity** → **Cycle**

3 Save your work

Try adding this technique to other parts of Melvin such as his shirt or shorts.

Summary

This example was simple because it was a cyclical motion: the air follows the motion of the head. A more complex example would be if you were creating a fat character that jumps and stops. The stomach would continue

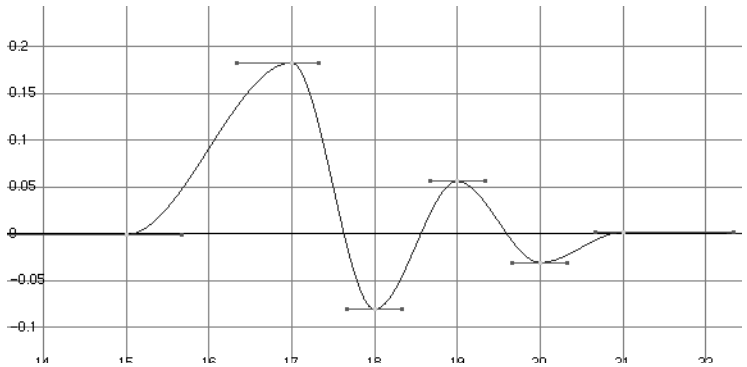
Key editing tips

Below are some tips for editing keys:

- **f** frames selected
- **a** frame all
- **Ctrl-Alt-LMB** box zoom
- **q** toggles out of current function to select keyframe or curve
- **w** toggles into a mode for translating keyframes or handles
- **r** toggles into a mode for scaling keyframes
- **shift** constrains vertically or horizontally. This applies to transforming keys or manipulating the view.
- To accurately set a key's value or time, use the **stats** fields in the upper left side of the graph editor window

to move after the skeleton stopped, rather than just following the motion a few frames behind.

You could apply this technique to a cluster on the stomach and then add keyframes in a sine wave pattern that flattens over time. The figure below illustrates this.



A jiggle that reduces over time

Appendix D

D Alternate Feet

Following are some alternative solutions for animating and controlling a character's foot.

Inverse foot

Creating a foot that responds naturally is quite a challenge. Here is a technique that allows for direct manipulation of foot placement and rotation, as well as realistic flexing and pronation, with a minimal amount of manipulation.

1 Create five joints

- Draw the following 5 joints with **Auto Orient** set to **None**.
- Name the joints *Hip, Knee, Ankle, Ball, Toe.*

2 Set up RP IK handles

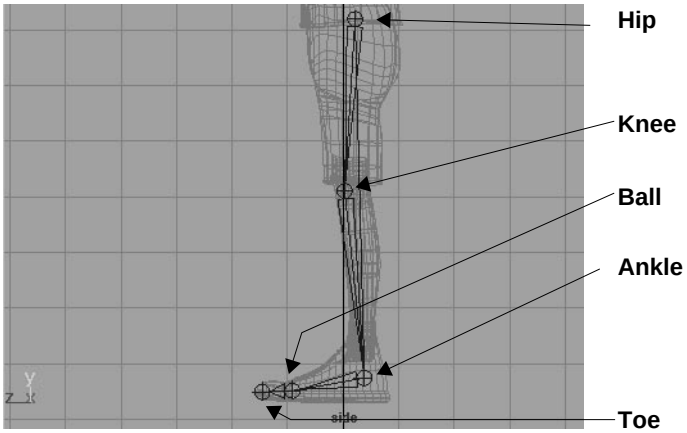
- Set up RP IK handles between:

  Hip and Ankle joints;

  Ankle and Ball joints;

  Ball and Toe Joint;

- Label the IK handles: *AnkleIK, BallIK*, and *ToeIK*, respectively.

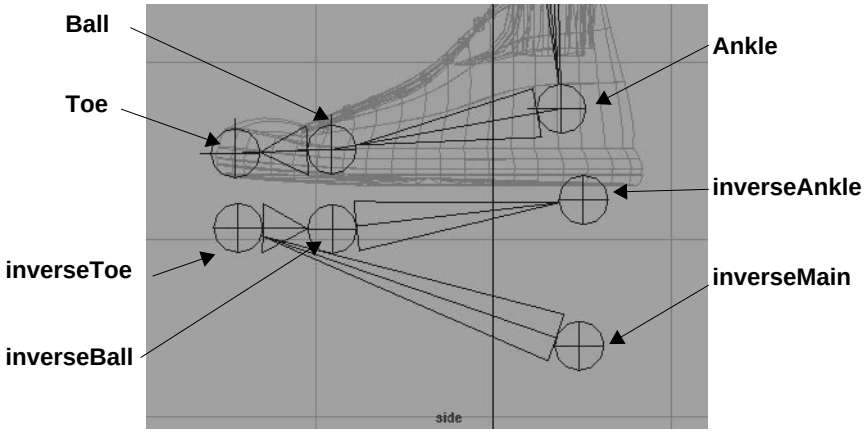


Leg joints

3 Create four control joints

- Create the following 4 joints to serve as controls for the IK handles (shown below):

  inverseMain, inverseToe, InverseBall, and *inverseAnkle.*



Reverse foot control

- Position the inverseToe, inverseBall, and inverseAnkle joints so they are in the same position as the Toe, Ball, and Ankle Joints.

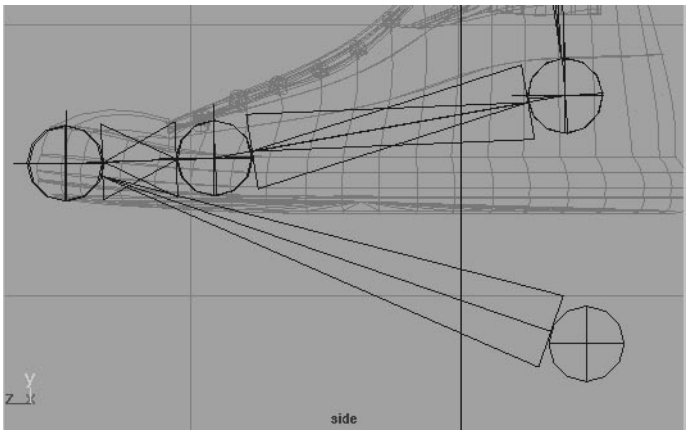
4 Set up point constraints

Point constrain by shift-selecting the joint then the IK handle, followed by **Constrain** → **Point**.

inverseToe to *toeIK*

inverseBall to *ballIK*

inverseAnkle to *ankleIK*



Point constrained foot control

With this inverse foot method you now have the following controls:

- *inverseAnkle* (rotate the foot)
- *inverseBall* (rotate)

- *inverseMain* (translate and rotate)

IK AND FK COMBINATION FOOT

This method creates Inverse Kinematics from the hip to the ankles, but requires Forward Kinematics for the ankle rotation and the bending of the toe.

Skeleton set-up

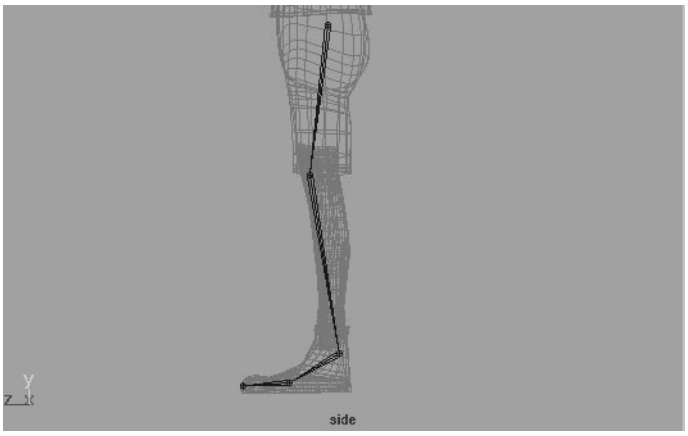
The skeleton for this method is similar to the one used in this course except it has no heel joint.

1 Create the legs and feet skeleton

- Starting with the hip, place 5 joints (hip, knee, ankle, ball, toe). Draw the knee in a bent position.

2 Move the legs into position

- **Move** his leg to his left side.
- Mirror these joints for the right leg.



Leg setup

Setting up Single Chain IK

Set up a Single Chain solver between the hip and the ankle of each leg.

1 Set up an SC solver between the hip and ankle

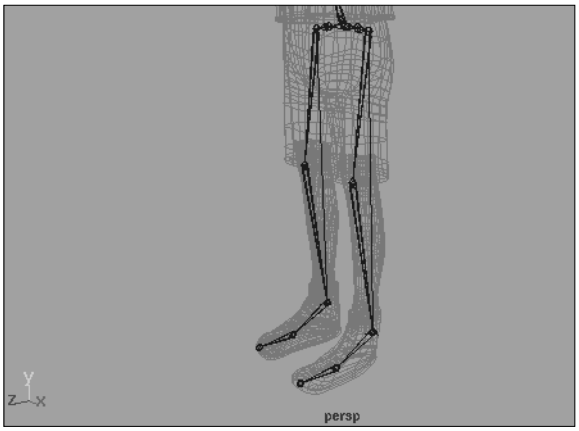
- Select **Skeleton → IK Handle Tool -** ❐

  make sure **ikSCsolver** is selected.

- Select the *left_hip* to establish the starting point of the IK chain and then the *left_ankle* to establish the ending point of the IK chain.

  Note that an IK chain with joint effector is automatically created. The end effector is the transform position of the IK handle.

2 Repeat for the right leg



IK from hip to ankle

3 Label the IK handles

- label the IK handles *left_legIK* and *right_legIK*.

Parenting the IK and Orient constraints

Note that if you select the IK handle and rotate its *y* axis, the leg rotates with it. By parenting this handle to another object you can have two levels of control:

- Y-rotating the parent object will rotate the leg
- Y-rotating the IK handle will allow the knee to point in a different direction than the foot. You can use this second level of control when Melvin bends down.

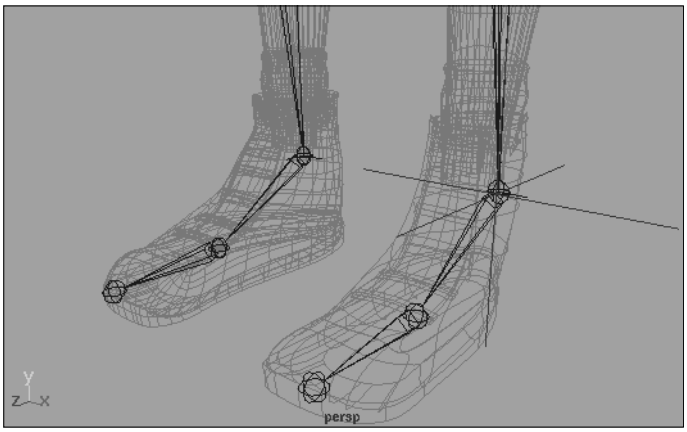
Once this parenting is complete, you want the foot to point in the same direction as the locator. Using an Orient constraint on the ankle joint, the foot's rotation will be the same as the locator.

Parenting the IK

First you'll create locators to serve as parent objects for the IK handles. Then you'll repeat the process for the other leg.

1 Create a locator and place it near the left ankle.

- Select **Create** → **Locator**.
- Translate the locator to the ankle.
- Scale the locator bigger to make it easier to select later.
- Label the locator *left_ankleLocator*.



Locators on ankles

2 Parent the IK handle to the locator

- In the Outliner, select *left_legIK*.
- Ctrl select the *left_ankleLocator*.
- Select **Edit** → **Parent** from the main menu.

3 Repeat for the right leg

- name the locator *right_ankleLocator*.

4 Orient constrain the ankle joint to the locator

To make the foot rotate with the locator, you'll use orient constraints.

- In the Outliner, select the *left_ankleLocator*.
- Ctrl-select the *left_ankle* joint.
- Select **Constrain** → **Orient** from the main menu.

5 Repeat for the right leg

You should now be able to select either ankle locator to move the ankle and rotate the foot and leg.

Bending toes and twisting the knee

For more realistic character motion, the toes should bend when a character walks. To manually animate toe-bending, you would select the ball joints and rotate them for each step the character takes. Instead, you'll streamline the animation workflow by adding an attribute to the ankle locators. Then you'll use this attribute to drive the bending of each toe.

Adding attributes

You'll create two attributes, **bendToe** and **kneeTwist**, that will be used to drive the *x*-rotation of the ball joint and the *y*-rotation of the IK handle.

1 Add an attribute to the locator at the ankles

- Select *left_ankleLocator*.

- Select **Modify** → **Add Attribute** and set the following:

  Data Type to **Float**;

  Attribute Type to **Scalar**.

  Name the attribute **kneeTwist**.
- Click **Add**.
- Add another attribute to the locator called **bendToe**.

2 Connect the kneeTwist attribute to drive the rotation

- Select **Windows** → **General Editors** → **Connection Editor...**
- In the Outliner, select *left_ankleLocator*.
- In the Connection Editor, click **Reload Left**.
- In the Outliner, select *left_legIK*.
- In the Connection Editor, click **Reload Left**.
- In the left column of the Connection Editor, click **Knee Twist** and in the right column, click the **Rotate** folder then click **RotateY**.

  You have now created a connection between the **KneeTwist** attribute and the **RotateY** attribute. When you adjust **KneeTwist**, you are actually rotating the leg IK —and thus offsetting the leg rotation from that of the foot.

3 Connect the bendToe attribute to drive the ball joint

- Select **Windows** → **General Editors**→ **Connection Editor...**
- In the Outliner, select *left_ankleLocator*.
- In the Connection Editor, click **Reload Left**.
- In the Outliner, select *left_ball* joint.
- In the Connection Editor, click **Reload Left**.
- In the left column of the Connection Editor, click **BendToe** and in the right column, click the **Rotate** folder then click **RotateX**.

  You have now created a connection between the **BendToe** attribute and the **RotateX** attribute. When you adjust BendToe, you are actually rotating the *left_ball* joint—and thus bending the toe.

4 Repeat for the right ankle

Now you can animate everything below the waist by selecting either of the ankle Locators, foot placement, ankle rotation, and toe bending.

EXPRESSION FOOT

This example is similar to the foot in Chapter 2, but it is done with expressions and a slightly different hierarchy.

The file *legs_exp.mb* is reference for this.

Build the leg with SC IK

Build the leg with IK handles to control the different parts of the foot and the leg.

1 Build a leg skeleton

- Draw a skeleton leg with the following joints: *hip, knee, ankle, heel, ball* and *toe*.

2 Add IK Single chain handles

- Add a chain from the hip to ankle named ***R_ankleIK*.**
- Add a chain from the ankle to ball named ***R_ballIK*.**
- Add a chain from the ball to toe named ***R_toeIK*.**

3 Create a control hierarchy for the right foot

- **Group** the *R_ankleIK* and rename the new node *R_ballRoll.*
- Set the pivot to the ball
- Group the *R_ballRoll* beside the *R_ballIK* and rename the new node ***R_toeRoll*.**
- Set the pivot to the toe
- Group the *R_toeRoll* beside the *R_toeIK* and rename the new node ***R_heelRoll*.**
- Set the pivot to the heel
- Group the *R_heelRoll* and call it ***R_foot*.**

4 Add Attributes

- Add two attributes to the rfoot node called **roll** and **angle**.

5 Add an expression

- Add the following expression to the *R_foot* node:

```
if (R_foot.roll < 0)
                R_heelRoll.rotateX = R_foot.roll;

else if (R_foot.roll >= 0 && R_foot.roll <
R_foot.angle)
   {
   R_heelRoll.rotateX = 0;
   R_ballRoll.rotateX = R_foot.roll;
   R_toeRoll.rotateX = 0;
   }

else if (R_foot.roll >= R_foot.angle)
   {
   R_ballRoll.rotateX = R_foot.angle;
```

```
R_toeRoll.rotateX = R_foot.roll - R_foot.angle;
}
```

The roll attribute on the foot will make the foot roll from heel to toe. The angle attribute controls the point at which the ball stops rotating. Try setting it to an initial value of 30.

ALTERNATE LEG IK SET UPS

Here you'll use the rotate plane (RP) IK solver for Melvin's legs and arms. The difference between this solver and the Single Chain (SC) solver is that the RP solver provides a way to control the direction the knees and elbows will point. This is achieved by using a special aiming constraint called a **Pole Vector**. You will use **locators** as the pole vector constraints and animate them as needed to control the aiming of the knees and elbows.

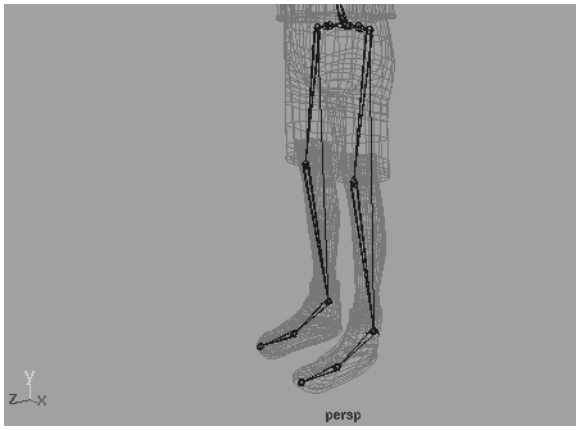
Rotate Plane IK

You will set up Rotate Plane solver between the hip and the ankle of each leg. The RP solver is the default IK solver.

1 Set up an RP solver between the hip and ankle for each leg.

- Select **Skeleton** → **IK Handle Tool** ❐ and make sure **IkRPsolver** is selected.
- Select the *left_hip* to establish the starting point of the IK chain and then the *left_ankle* to establish the ending point of the IK chain.
- Notice that an IK chain with joint effector is automatically created. The end effector is the transform position of the IK handle.

2 Repeat for the right leg



Leg setup

- Label the IK handles *left_legIK* and *right_legIK*.

Point and Orient constraints—legs and feet

To help control motion, Maya offers several constraint options: point, orient, aim, and pole vector. Constraints limit the motion of the constrained entity based on the type of constraint. Before you add the constraints, you'll create locators to control multiple keyframeable attributes with one entity (the locator).

At each ankle you'll create locators to constrain both the position of the IK handles and the rotation of the foot. You'll point constrain the IK handles to the locators and orient constrain the ankle joints to the locators. An added benefit of the point constraining is that it will set up a pseudo "sticky-type" anchor for the feet. The benefit of this approach will become apparent when you start to animate and pose Melvin.

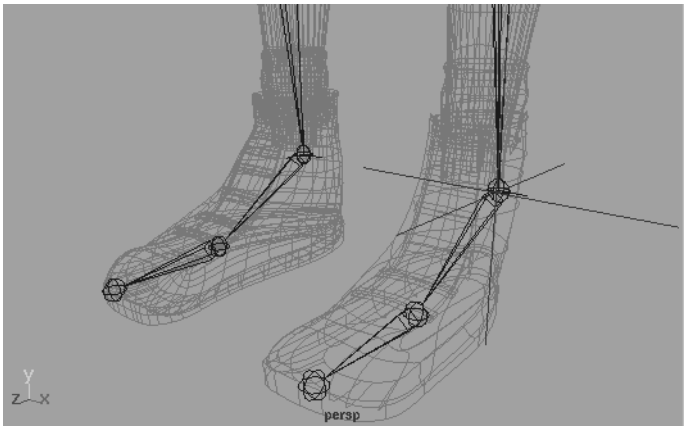
Adding constraints

You'll set up point constraints at each ankle to constrain the position of the IK handles to the locators. You'll work on one leg, first creating the locators, then adding the constraints. Then you'll repeat the process for the other leg.

1 Create a locator and place it near the left ankle

- Select **Create** → **Locator**.

  A cross-hairs appears at the world origin.
- **Translate** the locator to the ankle. You do not have to be precise in your positioning of the locator.
- **Scale** the locator bigger to make it easier to select later.
- Label the locator *left_ankleLocator*.



Control for the feet

2 Point constrain the locator to the IK handle

- In the Outliner, select the *left_ankleLocator*.
- **Ctrl-select** *left_legIK*.
- Select **Constrain** → **Point** from the main menu.

3 Orient constrain the locator to the ankle joint

- In the Outliner, select the *left_ankleLocator*.
- **Ctrl-select** the *left_ankle* joint.
- Select **Constrain** → **Orient** from the main menu.

4 Repeat steps1-3 for the right leg

- name the locator *right_ankleLocator*.

You should now be able to select either ankle locator to move the ankle and rotate the foot.

Pole vector constraints—knees

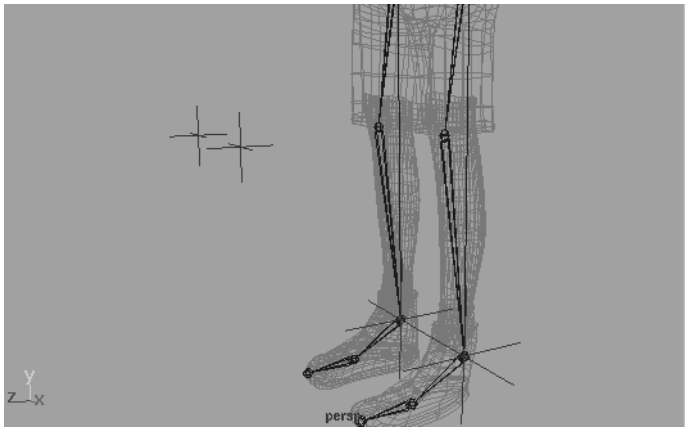
The Rotate Plane solver includes a constraint to control which plane the RP solve will work in. This constraint is the Pole Vector constraint. It's a great way to visually determine which direction the knees will point.

Adding pole vector constraints

You'll add Pole Vector constraints to Melvin's knees. This constraint is also useful for elbows.)

1 Create a locator for each leg and place it out in front of each knee

- Label these locators *left_KneeLocator* and *right_KneeLocator*.



Constraints for the knees

2 Pole Vector constrain these locators to the IK handle of each leg

- In the Outliner, select *left_KneeLocator*.
- **Ctrl-select** *left_ankleIk*.
- Select **Constrain** → **PoleVector** from the main menubar.

  Notice a line is drawn between the *left_KneeLlocator* and *left_hip* where the *left_legIK* begins.
- Repeat for the right leg.

Parenting the pole vector locators

You'll want these pole vector locators to travel with your character when you animate. There are several places to parent these locators depending on the preference or the needs of the shot. Since you want these pole vector locators to travel *with* the character when you animate, you'll parent them to the skeleton. There are several places to parent these locators but you'll parent them to the root because this covers most of the general poses for the animation.

1 Parent the PoleVector Locators to the root of the skeleton

- In the Outliner, MMB drag and drop *left_kneeLocator* and *right_kneeLocator* over *back_root* joint.

Tip: There are several other places you could parent these pole vector constraints for the knee, depending on the behavior you want or the needs of the shot:
- hip joints
- ankle or ball joints
- keyframe by hand (no parenting)

Appendix E

More Facial Setup

This appendix will look at setting up different aspects of the face, including:

- Using clusters for eyelids
- Aim constraints to control the eyes
- Spline IK applied to the tongue
- Applying sculpts and lattices to a face
- Modeling a closed mouth with wire deformers

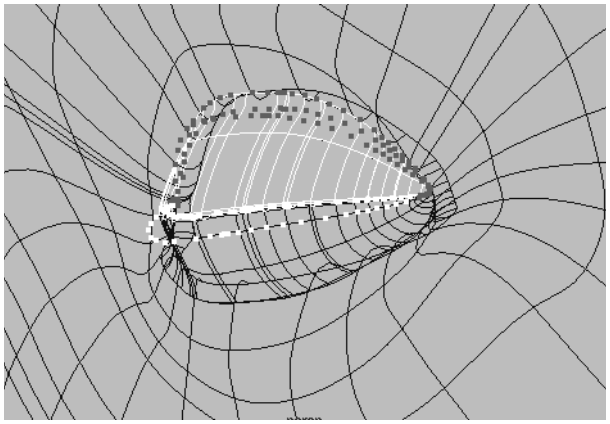
EYELIDS

Eyelids are folds of skin that are pushed together to cover the eyeball. They have thickness and generally have a seamless transition from the cheek. One method of creating and animating the eyelid is to model your character with this feature in mind.

This example uses separate eyelids as lofts that were generated from isoparms on Melvin's eye socket. This example starts after the eyelid is built.

1 Cluster the eyelid

- Select the *uppereyelid* object.
- In component mode, select the bottom rows of CVs.



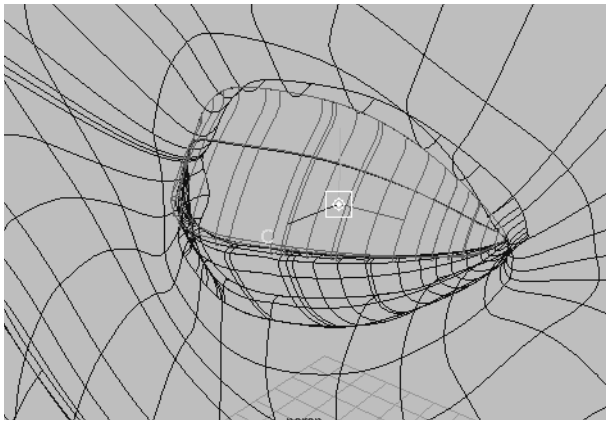
Making a cluster

- Create a cluster for these CVs by selecting **Deform** → **Create Cluster –** ❐, and set the following:

  Mode to **Relative**

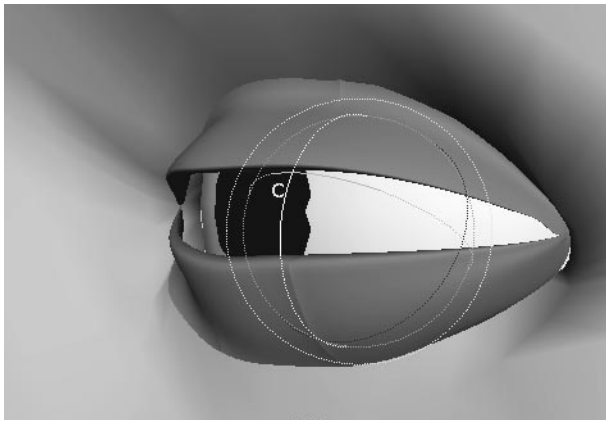
2 Move the pivot of the cluster

- **Select** the cluster then press the **Insert** key to select the pivot manip tool.
- **Translate** the pivot manip back to the center of the eye.
- Press the **Insert** key again.



Moved cluster pivot

3 Rotate the cluster to test



Rotated cluster

4 Add the rest of eyelid to the cluster

Using Edit Membership and the Set Editor you can add most of the other eyelid CVs to this cluster. You would also weight them for realistic motion and falloff.

5 Repeat for the bottom eyelid

The same procedures can be applied to the bottom eyelid.

Add control to the eyelids

Controlling these eyelids is a simple matter of creating an attribute that can control the rotate x of these clusters using set driven key. Since the neck is the most animated joint of the head, this is a logical place to add attributes for facial control. Adding a selection handle to the neck joint and then adding our facial control attributes to this joint will create a centralized method for selecting. The jaw is another place you could apply these attributes as this will be an animated joint as well.

1 Add an attribute to the neck joint

- Add an attribute called **leftBlink** to the *neck_joint*.
- Open the **Set Driven Key** window and **select** this attribute as the **Driver**.

2 Create a Set Driven Key for the eyelids

- Select the *clusters* as the **Driven**.
- Set keys with the **leftBlink** at **0** and the eyelids open.
- Set keys with the **leftBlink** at **1** and the eyelids closed.

Tracking history of the face

The eyelids are an integral part of the face and eye socket. As you apply deformations to the face you will want the eyelids to track with these deformations which could be created from a blendshape target. To get the eyelids to track, you could include these objects into the blendshape target so that they get deformed as well. Or, you could maintain the history of the loft that created the eyelid. The curves that generated the loft came from curves on the surface of face. If this history is maintained there is a good chance that the eyelids will track. Again, by incorporating the eyelid into the modeling of your character you will get a much better integration at animation time.

Another method is to parent the eyelids to the eyeballs. This keeps the eyelid and eyeballs as separate structures that are not part of the bind skin. They are parented to the joints and do not participate in blendshape deformations. This way, the eyes maintain their shape as the skin moves around them. They look more solid, but you have to make sure that the skin deformations do not interpenetrate the eyes too obviously.

EYEBALL ORIENT CONSTRAINTS

Aiming the eyeballs can be obtained by **Orient Constraining** their orientation to a locator. This locator can then be placed in the scene and animated to guide the character's attention from a single point.

- **Group** the eye and pupil. This ensures that the eye moves as a unit and it will make the local axes line up with the world so that the eyes do not move strangely when the constraint is added.
- **Orient Constrain** these groups to a locator.
- **Group** the locator under the Character.

MELVIN'S TONGUE WITH IK SPLINE

A critical part of believable Lip Sync is the tongue. Many phonemes such as the "th" in the "the**"** word involve a very visible up turned tongue against the upper front teeth.

The tongue could be included in the blend shape or you could use. IK spline like you did on Melvin's back. You can create a tongue from a sphere, create joints for it, bind the skin and add IK spline.

1 Open an existing file

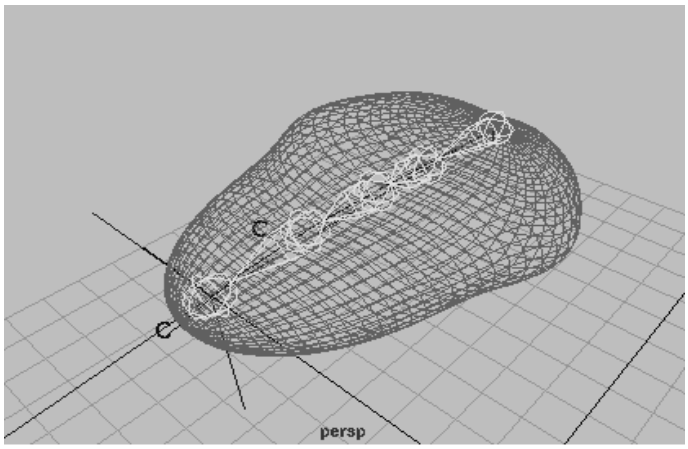
- Import the file *melvinTongue.mb*.

  This is the geometry and skeleton

2 Run SplineIk through the Joints

3 Add clusters to the splineIK

- Add two clusters to the splineIK curve CVs at the tip of the tongue and in the middle of the tongue.
- **Parent** the clusters under the SplineIK curve.



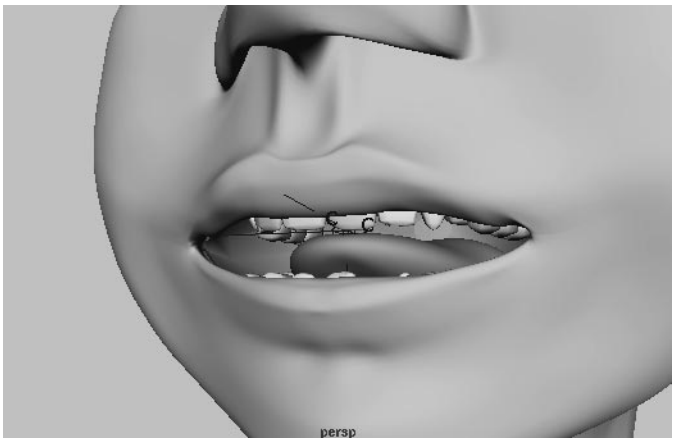
Skeleton for the tongue

4 Bind the tongue geometry to the tongue skeleton

Note that you cannot parent the tongue skeleton onto the jaw joint without running into a problem with the tongue geometry. The tongue geometry can not be skinned to the skeleton and also parented under the Melvin skeleton. This will result in double transformation of the tongue geometry. To remedy this:

- **Move** the tongue geometry to a group with the rest of Melvin's geometry that is skinned.
- **Bind** the tongue to the skeleton.

  Controlling the tongue is now a matter of selecting either of the clusters and translating or rotating them to position the tongue.



Tongue control

SCULPT DEFORMERS IN THE FACE

Sculpt Deformers provide a means of deformation that is uniform and shape driven. They work well for moving a structure under the skin that is solid like a bone. They also provide a shortcut for building a deformation that otherwise would involve a significant amount of CV tweaking. Here are a few places that Sculpts might speed up building a target or integrating their movement with head joints and Set Driven Key:

Flaring Nostrils

Mode to **Flip**;

Inside Mode to **Even** or **Ring**

Adams Apple movement linked to the jaw

Mode to **Stretch**;

Inside Mode to **Even**

Tongue in cheek

Mode to **Stretch**;

Inside Mode to **Even**

Keeping the eye socket clear of the eyeball

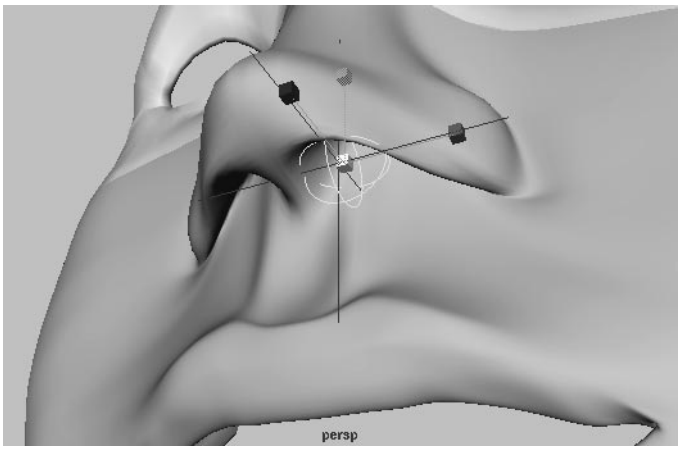
Mode to **flip**;

Inside Mode to **Ring**

Make sure that the sculpt is occurring *after* the lattice deformer in the history of the eye socket surface.

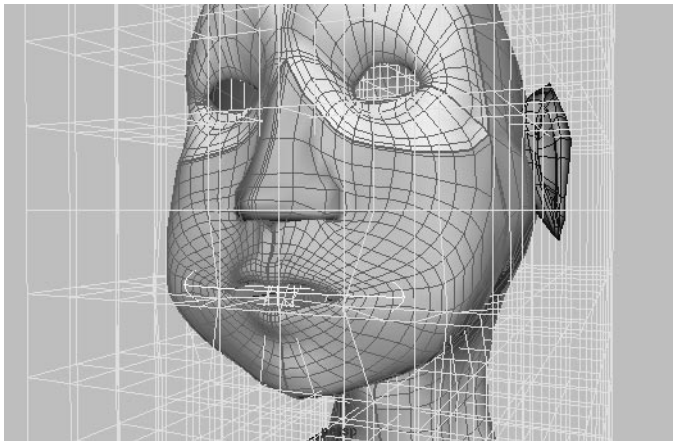
Inflating the cheeks

Apply the sculpt to the lattice that is deforming the face or mouth.



Sculpt deformer inflating nostril

You can also deform other deformers, like lattices, with the sculpt deformer which can provide a powerful method of getting a uniform Lattice deformation that would be difficult to do by tweaking individual lattice points.



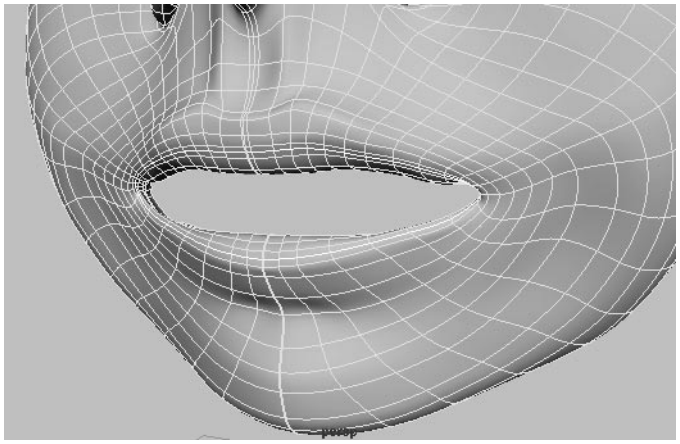
Sculpt on Lattice for even control of Lattice points

CLOSING A MOUTH WITH WIRE DEFORMERS

As seen above, deformers can be used to facilitate the creation of facial features and mechanics. This example will use a wire deformer to model a new target for a blend shape.

1 Create a wire curve

- **RMB** marking menu select **Isoparm** on the surface
- **MMB** select an Isoparm that is running around the Lips.



Select an isoparm on the lips

- Select **Edit Curves** → **Duplicate Surface Curves**.
- Select **Edit** → **Delete by Type** → **History** to remove the history on the duplicated curve.
- Rename the curve *wireCurve.*

2 Create a wire deformer

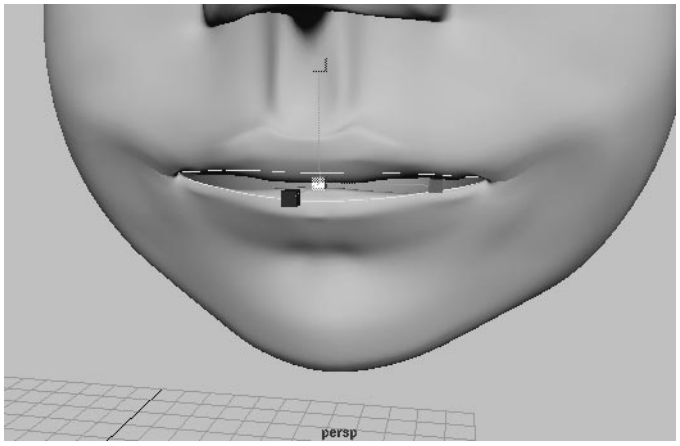
You will make the extracted curve a wire deformer.

- Select **Deform** → **Wire Tool.**
- Select the face object and press enter
- Select the *wireCurve* curve and press **Enter**

  This creates a wire deformer on that curve which will only affect the selected surface

3 Manipulate the wire deformer

- Scale the *wireCurve* in Y to pull the lips together.



Closing the mouth