Innovations Report

Automated Rig and Tutorial for Biped Character Setup

Rachael Hender

BACVA3 2007

Contents

Abstract

This report outlines research into the area of automated rigging as well as design and implementation of a tutorial based automated rigging script, created in MEL, aimed as a learning resource for beginners in computer animation. Current automated rigging systems and character setups tutorials are analysed, as well as a breakdown of the fundamental key features that make up a stable general purpose rig.

Introduction

I have chosen to undertake this project as I have a particular interest in character setup and rigging. My previous experiences as a PAL Leader involved explaining key concepts behind character rigging and developing a tutorial seemed an excellent way to collect my ideas and consolidate them. The idea for an auto rig came from the factor that rigging can usually be a long painstaking process and by integrating a tutorial into this, users could learn whilst also productively creating a rig.

But what is rigging and why use an auto rig? *The Complete Reference Maya 6* talks about the basic concept behind setting up a character for animation.

> To animate a character effectively, you need a structure that will allow you to control the deformations of the geometry based on a real-life organism……This type of deformer is of a special class called skeleton. A skeleton in Maya serves the same function that a skeleton serves in real life – it forms the internal structure of the character and it serves as the framework for performing character movement and deformation.[1]

The setup of a character can be a lengthy process and relies upon a level of technical knowledge.

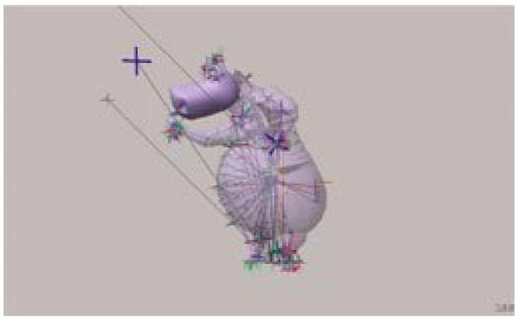


Figure 1.1 – The rig for Gloria the Hippo from Madagascar. To a beginner the controls used here may look complicated and confusing.[2]

A rig generator or 'auto rig' is designed to cut down on the time taken for this process by having options for a pre-existing rig scripted into a useful tool. However, it is important to remember that characters come in many different shapes, sizes and species. One type of rig created by a rig generator is not going to work for every character needed. A biped rig

[1] Meade, Tom and Arima, Shinsaku *The Complete Reference Maya 6*, (Osborne Publishing, United States, 2004) p.241-242
[2] http://www.studiodaily.com/filmandvideo/technique/casestudies/4467.html

generator is useless to animal characters and even subtle differences in human characters, such as the number of fingers, would require the end result to be modified. As an auto rig script for every eventuality would be incredibly difficult to implement, it is more often that an auto rig generator produces a good base rig that can then be developed further. I intend to use the functionality of an auto rig to help beginners learn rigging from the start.

Aims

The overall aim of this project was to create an automated rigging script for a biped character incorporating a comprehensive tutorial for beginners. For this I would use Maya's embedded scripting language MEL to create what I hoped would be a simple and informative way to learn rigging whilst generating fast results.

The main specification for my rigging script was as follows:

- Adapt to generate a skeleton structure for a biped character of any proportion
- Create a basic control rig that is functional and robust
- Create a rig that is suitable for beginners with simple and clear controls
- Integrate a thorough tutorial, explaining the methods and processes used
- Have a straightforward and friendly graphical user interface (GUI) for ease of use
- Leave users having obtained knowledge in the area of rigging and ideally a new found confidence to attempt further rigging in future.

Research

Current Automated Rigs

Before attempting to create my own automated rigging script, I researched existing examples to assess how they worked and what I liked or disliked about them.

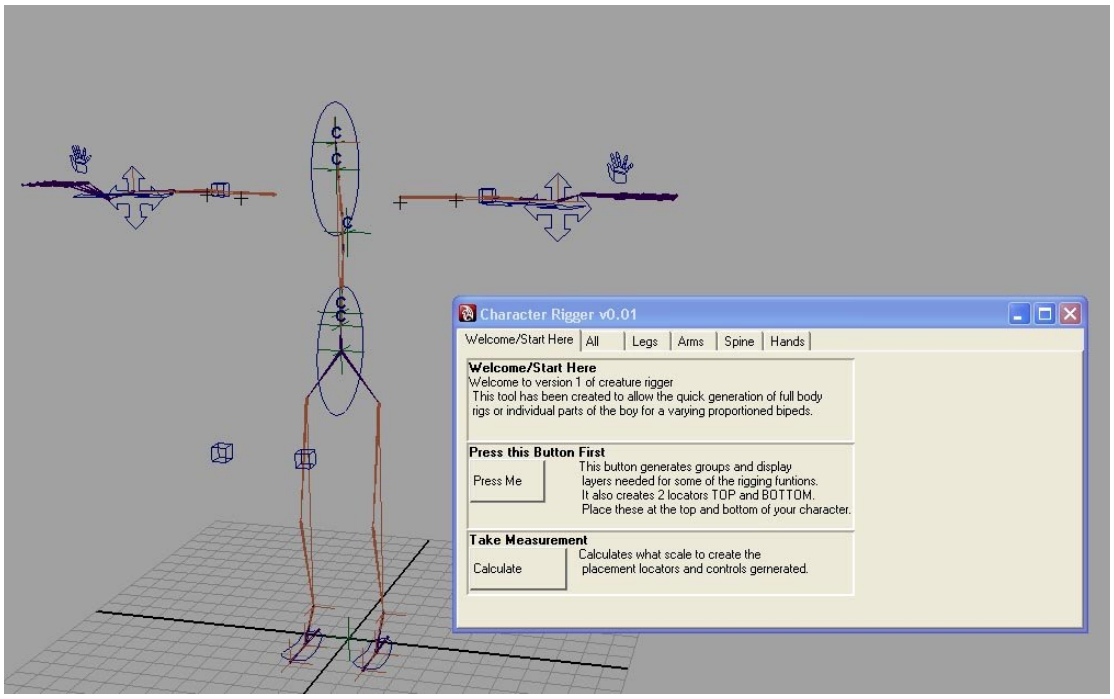
James Nicholl – Character Rigger v0.01

The first automated rigging script I came across was an 'Innovations' project from last year.[3] It works by finding the height of your character by calculating the distance between two locators. Then a series of locators are created representing places in the body where you would expect to find a joint. You can change the position of these locators to fit inside your character mesh, choosing how you would like your arms and legs setup and how many spinal joints to create. This ends the interactivity of the user and a skeleton and completed rig will appear in seconds.

The likeable aspects of this auto rig are the clear interface and the choices available to the user. There is also the option of just creating individual rigged parts such as arms or legs which helps with the adaptability that is so hard to accomplish with an automated rig.

[3] http://ncca.bournemouth.ac.uk/gallery/view/32/Automated_Biped_Character_Skeleton_and_Control_Rig_Generation

Although the choices available for this setup were impressive I found that the final rig was overly complicated with confusing controls and is not one that I would want to create for a beginner.



Welcome/Start Here
Welcome to version 1 of creature rigger
This tool has been created to allow the quick generation of full body rigs or individual parts of the boy for a varying proportioned bipeds.

Press this Button First
This button generates groups and display layers needed for some of the rigging funtions. It also creates 2 locators TOP and BOTTOM. Place these at the top and bottom of your character.

Take Measurement
Calculates what scale to create the placement locators and controls gernerated.

Figure 1.2 – Character Rigger v0.01 and rig. The GUI is straightforward and informative. Notice the tabs for creating individually rigged parts. The rig however is confusing and overly complicated.

Lee Croudy's AutoRig v1.1a[4]

This auto rig script, made by a self confessed beginner at Maya, creates a skeleton which you then have to position to fit inside your character. This is good because you can instantly see the skeleton and how it is connected, compared to using locators which are just points in space making the positioning hard to visualise. However, positioning the joints is slightly complicated by the fact that they are already in a hierarchy and you cannot move one without affecting others. I also discovered during testing that with more extreme rigging poses the control curves would not be created in the correct locations.

The interface contained no explanation of how it should be used and in general the script seemed "buggy". Nevertheless the rig itself was simple, but to some extent did not add enough functionality in my opinion, so would not be very helpful to beginners.

[4] http://www.simplycg.net/viewtopic.php?p=3682

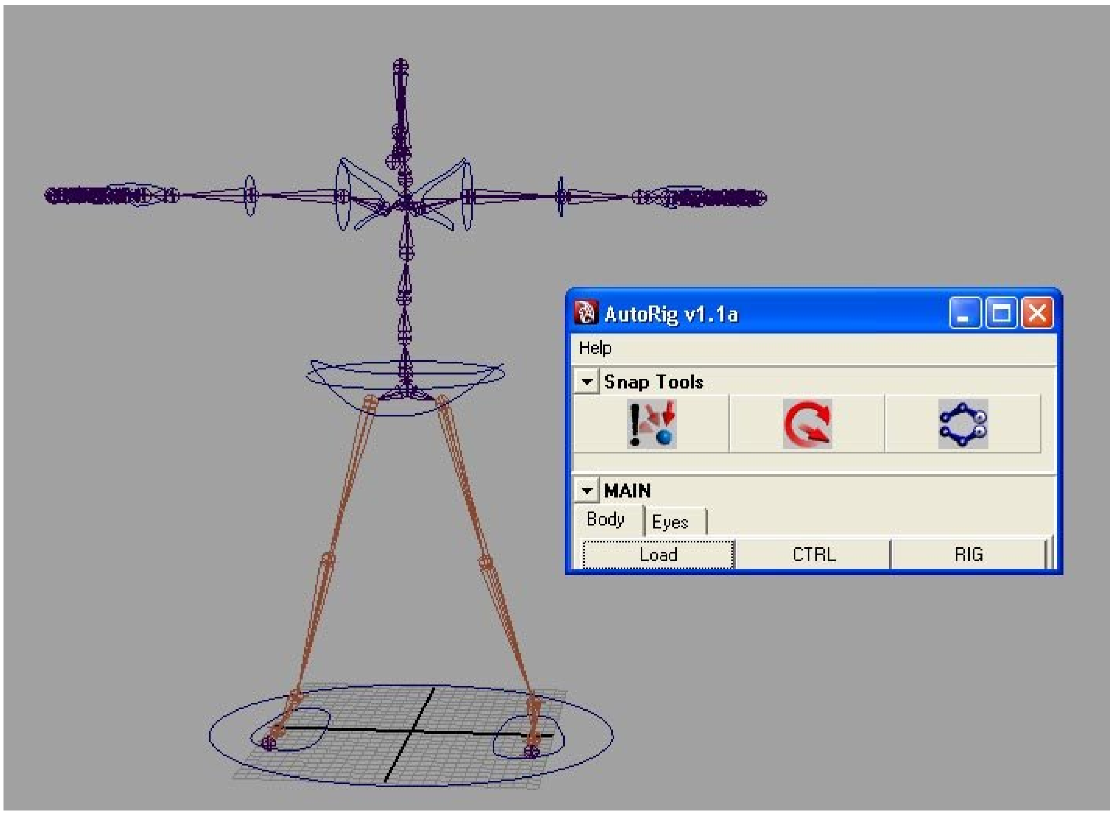


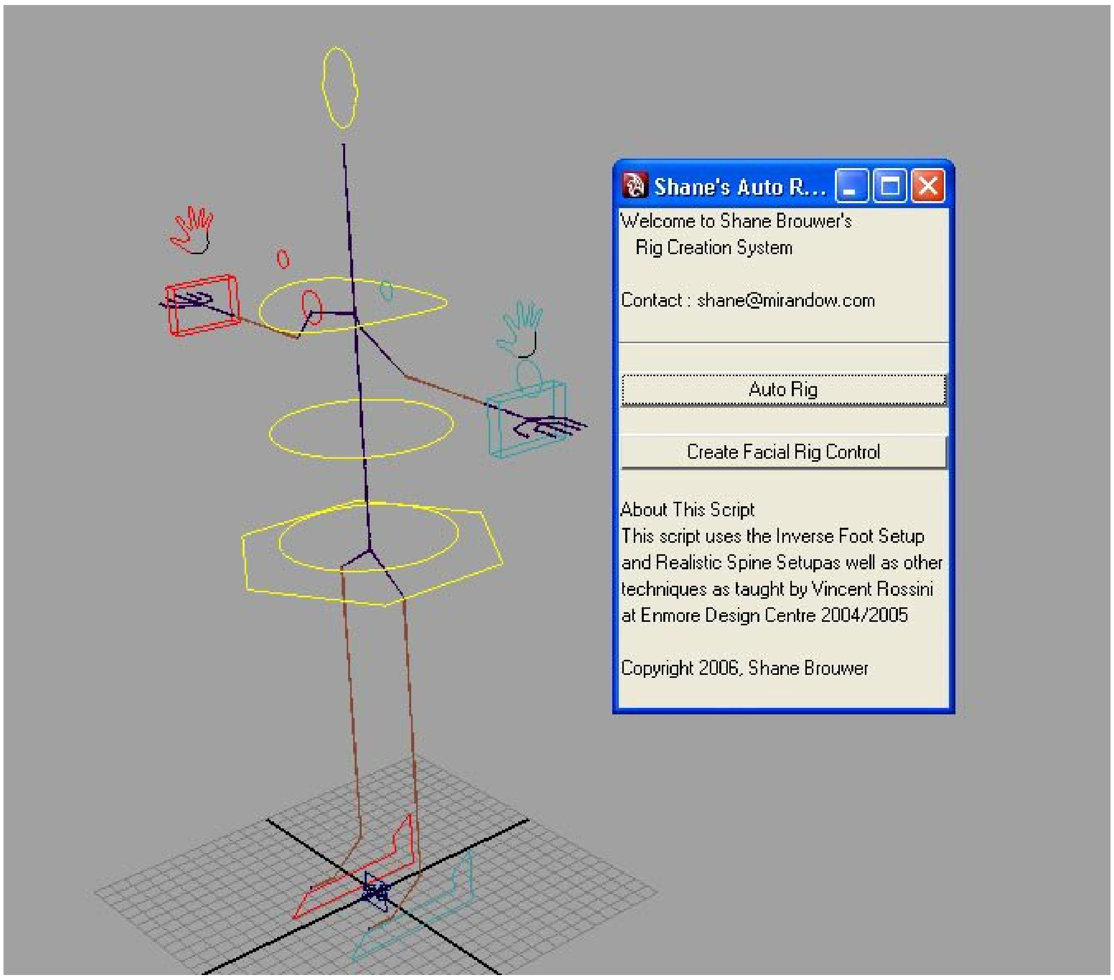
Figure 1.3 – AutoRig v1.1a and rig. This is a simple rig to use but the interface has no instructions or explanations. The icons do not even have a description.

Shane's Auto Rigger[5]

This is a simple fast and effective way of producing a rig for any biped character. The script again uses locators which are created and left to be positioned by the user. The completed skeleton and rig is created in just three steps. This was the easiest auto rigger to use out of the ones I tested, however, there were no options for what you were creating and very little explanation into what the script would do.

The rig itself was incredibly simple to use and much more suitable for a beginner. As well as creating the rig some extra functionality was built in to aid the skinning process such as the option to select all bind joints or go to the bind pose. For the sheer simplicity this was the auto rigger with the most potential.

[5] http://www.highend3d.com/maya/downloads/mel_scripts/animation/4049.html


Shane's Auto R...

Welcome to Shane Brouwer's
Rig Creation System

Contact : shane@mirandow.com

Auto Rig

Create Facial Rig Control

About This Script
This script uses the Inverse Foot Setup and Realistic Spine Setupas well as other techniques as taught by Vincent Rossini at Enmore Design Centre 2004/2005

Copyright 2006, Shane Brouwer

Figure 1.4 – Shane's Auto Rigger and rig. The rig is friendly with helpful symbols and colourful curves. The rigger is easy to use but lacks information.

Houdini Auto Rigger

So far I have looked at three individually made auto rigs. However it is not just programers with a passion for scripting who recognise rigging is an area that needs improving if a wide range of people are to benefit from a software package.

In 2006 Side Effects Software announced that Houdini 8.1 would include a new "artist friendly" auto rig feature. The official press release stated:

> Houdini 8.1 makes its character tools more accessible than ever. A new Biped Auto Rig tool lets you quickly position your character's joints and generate production-ready rigs at the touch of a button.[6]

This is great news for animators who are unconfident at rigging, allowing them to speedily progress through the character setup process. Conversely, I feel you cannot really be conversant with a rig without fully understanding the more technical side of how it was created because if it breaks, how will you fix it!

[6] http://www.sidefx.com/index.php?option=com_content&task=view&id=470&Itemid=55

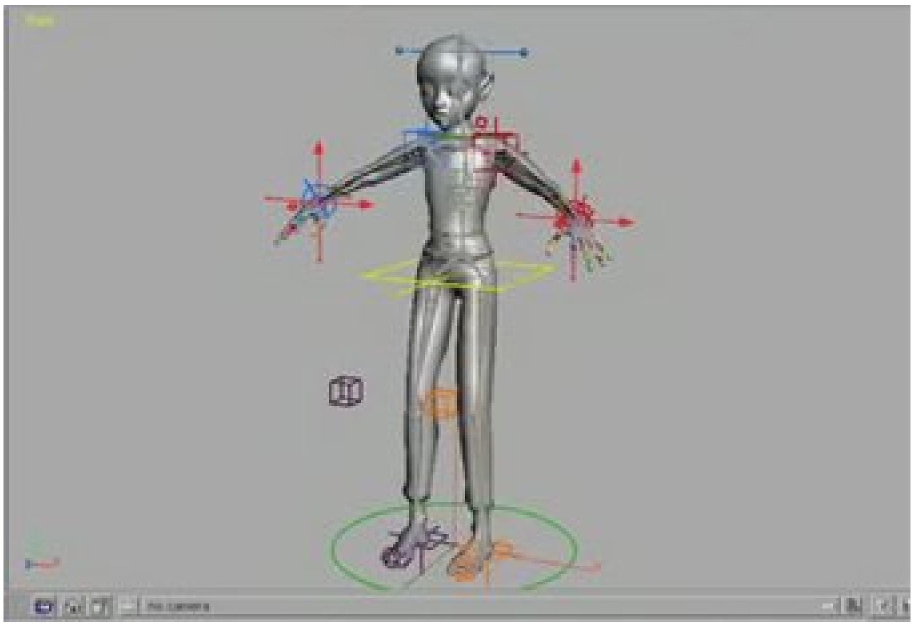


Figure 1.5 – Houdini 8.1’s auto rig in action. The skeleton has been fitted inside the body and controls have been created. No pop up interface is needed as the auto rig is integrated into the software.[7]

Current Rigging Tutorials

There is no shortage of tutorials on how to rig a character; the difficulty is finding the right tutorial for what you want your character to be able to do. A tutorial for an overly complicated rig that will take hours to setup is pointless if you are only planning to animate a walk cycle. Again a squash and stretch rig would be of no use to someone animating realistic human movements. Knowing what to choose for the right occasion relies upon some underlying knowledge of how rigging works. My research so far, alongside those tutorials I refer to when working on character setup, should give me insight into how to develop my own tutorial.

3D Buzz Rigging Tutorial

This video tutorial of how to setup a full biped character rig was created by popular 3D community www.3dbuzz.com. I used this tutorial to create my first ever character rig. Taking you through skeleton setup, controls for body parts, as well as more complicated features such as IK splines and clusters, this tutorial is a really useful way for a beginner to follow how a rig is created. A lot of the basics are covered alongside clear explanations and background reasoning. The final rig provides ample movement and functionality for a beginner.

The Complete Reference Maya 6

This book covers a basic overview of all aspects of the software. There are two chapters that cover character setup thoroughly with explanations of all the tools needed. Pictures throughout are helpful as well as showing the icons for tools to aid quick location in Maya.

[7] http://www.sidefx.com/index.php?option=com_content&task=view&id=471&Itemid=132

There are also several separate tutorial sections demonstrating key aspects of character setup. Complicated terminology such as kinematics, keyed relationships and constraints are explained, while menus such as the expression and connection editor are covered in detail to give a thorough understanding.

There is a lot of information in this book that would take a considerable period of study before the processes could be fully understood. This makes me realise my tutorial should be concise whilst still getting across the concepts behind rigging.

First Year MART Notes

My first introduction into how to setup a character for animation was from Henry Bush's 'Modelling Animation and Rendering Techniques' lectures in the first year. The notes from these lectures are available online[8] and are probably still used to teach students today. They cover the theory behind joints, skeletons, character controls, constraints and much more. It is far from a step by step as to how to rig a character, but teaches the principles that you would need to understand in order to then go and create you own rig. I shall be keeping these notes in mind when creating my tutorial.

The Art of Rigging Volume 1[9]

CGToolkit's definitive rigging guide is designed to teach the theory and practices used by technical directors in industry today. Most of the information in this book would be too advanced or unnecessary for a beginner looking to quickly rig up their character; however, the first chapter on basic character setup has some very handy tips that would be useful for everyone to learn. I am a particular fan of the foot setup in this book, which I have used in the past, amongst other gems of information.

There is no definitive method for producing the perfect rig as it is a vast subject area. Throughout my research I have looked at many tools and tutorials available today to help people to setup up their characters. I feel that a comprehensive, fast and effective tutorial to help beginners learn the basics has been overlooked and perhaps this is why the whole concept of rigging is often considered daunting. Basic principles should be taught so people can learn to produce simple and effective rig setups for themselves and have the confidence to do this time and again. This is what I hope my script will be able to accomplish.

Audience

It is important to spend time considering the audience for whom my tutorial will be written. The language has to be simple and clear without taking technical terminology for granted. It has to be assumed that the audience has only a very basic knowledge of the software, ideally enough to have already managed to model a character for setup. It is also important to not over-complicate explanations as there is the chance of disheartening the reader and making them lose interest. I will work on the premise of giving a basic overview, where less is more, and users can be directed to further information and explanations through website links.

[8] http://www.spookypeanut.co.uk/notes/
[9] Ritchie, K, Callery, J and Biri, K *The Art of Rigging Volume 1,* (CGToolkit, United States, 2005)

Designing a Base Rig

Creating a rig setup that is going to suit the majority of users is a difficult task and cannot satisfy every situation. I hope that by following the setup of this general purpose rig, users will gain the knowledge and confidence to become more ambitious in making a production specific rig. An overcomplicated base rig will fail to help beginners learn basic principles and a rig lacking in functionality will fail to communicate the principles of rigging. I want to keep the setup as simple as possible and yet at the same time maximise functionality. There are several key features every good rig needs which can be broken down into four parts, the setup of the skeleton, the leg, the arm and the body. This does, however, leave the facial setup that will not be covered in this tutorial as it is in itself too broad a subject.

The Skeleton

A good understanding of skeletal anatomy is important when setting up a character's skeleton for although this is a much simplified version, it is essential to know which are the important joints and how they work. There are two different types of joint to consider when setting up a 3D character, ball joints which can rotate in every direction i.e. hip and hinge joints which rotate in only one direction i.e. knee. In general most biped characters will have very similar skeletal setups with the number of joints in the spine being the most variable factor. For my skeleton setup I will produce a basic skeleton as shown in Figure 1.6.

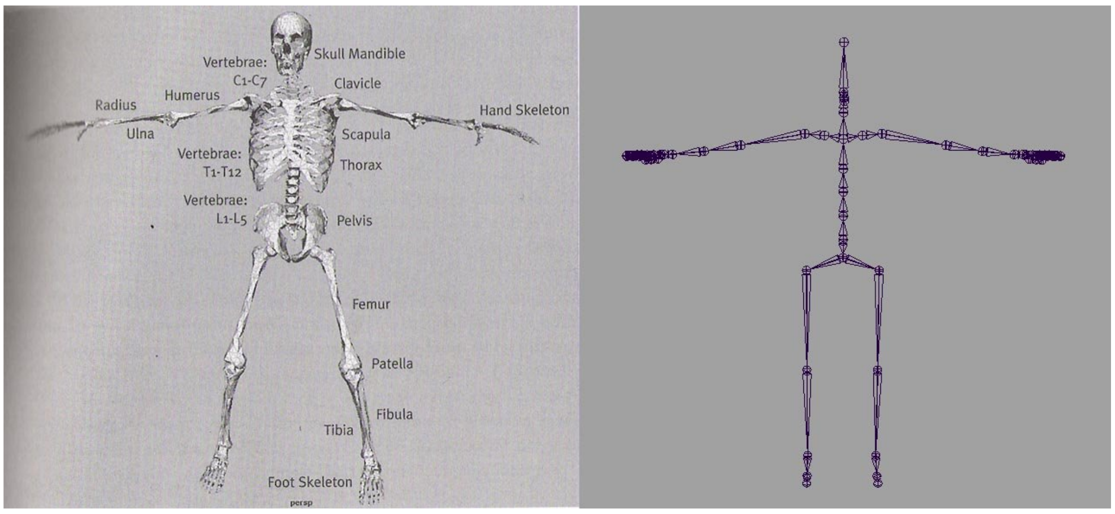


Figure 1.6 – A human skeleton (modelled in CG) compared with a simplified version for animation that includes four spinal joints and an extra joint for rotations of the jaw.[10]

The Leg Setup

Most commonly legs are setup with inverse kinematics (IK) where a series of joints are connected together using an IK handle. The joint at the end of the IK hierarchy controls the animation, so when used with legs for example, it will make the foot stick to the floor. It is also common to add attributes to the foot control that will handle foot roll, an essential for walk cycles. This is how I plan to setup the legs in my script.

[10] Maraffi, Chris, *Maya Character Creation – Modeling and Animation Controls*, (New Riders Publishing, United States, 2004) p.99

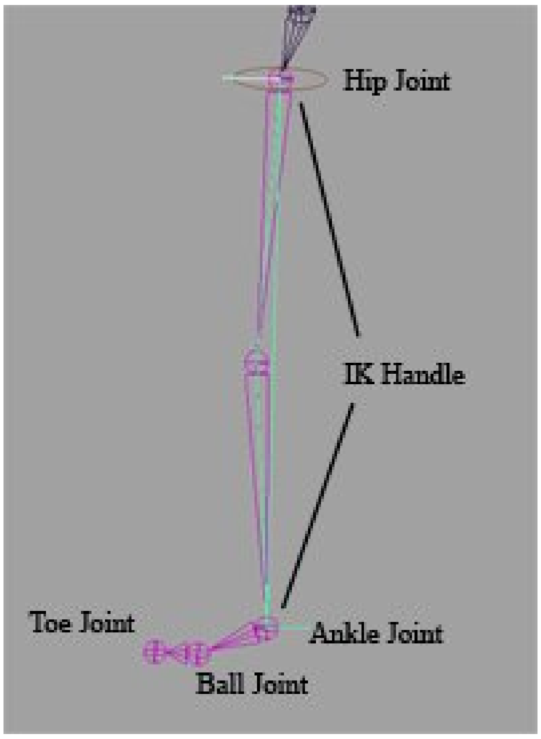


Figure 1.7 – The IK handle effects from the hip to the ankle giving the ankle control of the leg movement.

The Arm Setup

Setting up arms can vary between inverse or forward kinematics (FK) or sometimes even both. FK allows the skeleton to be animated as a normal hierarchy i.e. when a joint is rotated all those below in the hierarchy will follow. FK is used for creating arcs of animation such as swinging arms when walking, whereas IK arms would be needed when pushing against a wall. Accomplished riggers who want the benefits of both IK and FK may implement an IK/FK switch in the form of a driven key or an expression. Although this is useful I feel it is probably over complicated and unnecessary for beginners. I shall setup the arms of my rig using FK as IK has already been used on the legs and therefore users will have chance to learn both methods. For the fingers I shall add attributes to the wrist control that will allow for basic curl/spread animations.

The Body Setup

After the arms and legs have been setup the remaining joints make up the spinal column. There are several different ways of setting this up for animation and as always it depends on the flexibility needed. As the body is made of a long chain of joints an IK spline could be use integrating a NURBS curve into the back of the rig that will calculate the animation as control vertices of the curve are moved. However I feel that this setup would be overcomplicated for explanation in an auto rig tool. Instead further FK will be added to the spinal joints allowing them to rotate with freedom.

I have tried to choose the setup that I feel will produce the most general purpose, stable rig. A lot has been based on my own personal preferences and experiences with rigging but I feel the setup I have chosen will be most effective for beginners as well as demonstrating a variety of techniques that can be used again in future. It was important not to overload the rig with pointless tools that may never be used.

Writing the Tutorial

The tutorial part of my rigging script was a key feature to differentiate it from other available auto rigs, but also to help users learn about the process of rigging. Alongside scripting my auto rig, I wrote out a step by step tutorial broken down into small digestible pieces covering all the principles that the auto rig would use. My tutorial also included how to use the script and I considered what buttons and images would be included in the final interface.

Writing out the tutorial was my first opportunity at collating all the important information I thought would be essential for inclusion. I then rewrote it in a language that was simple, concise, with definitions for tools and processes broken down into clear explanations to help users with very little knowledge of rigging.

Writing the tutorial was challenging as I found it hard to explain complicated concepts in such a proscribed format. To aid my tutorial I intended to add links to other useful websites and Maya help guides so that more information was readily available. I asked several people, both familiar and unfamiliar with rigging, to read through my initial tutorial and give me feedback. Some said that the level of detail was just enough; others said parts left them needing to know more. The key throughout was to find a suitable level with enough information to hopefully build confidence and spark interest to learn.

Making the Script

Scripting was always the most challenging part of this project for me owing to my somewhat limited knowledge of MEL scripting language: I learnt as I progressed. At the beginning the idea of creating a tool using a scripting language seemed like an impossible brief but I found by breaking down every problem into small solvable tasks I could gradually put these together to build something bigger and more useful.

My first task was how to setup the skeleton, considering the character could be of any proportion. Thanks to my research into auto rigs beforehand I had decided that a locator based setup would be simplest to implement and most effective. This worked by creating a series of locators representing every joint in one half of the skeleton, the idea of half coming from the fact that the locators would be mirrored to the other side later. Each locator had co-ordinates of a point in space. By having the user move these locators to the correct position inside their character's mesh you could find the co-ordinates of where every joint should be created. The locators could then be mirrored and joints created. This was a very efficient way of solving the problem.

Another important factor I had to consider was that the skeleton needed to have a specific hierarchy of joints where the root is at the centre of gravity, all other joints radiating out, forming their own sub-hierarchies. This way the root controls the entire skeleton whilst the hip would control the knee and ankle below it. To create joints in a hierarchy required individual procedures to be made for separate hierarchical joint chains i.e., leg hierarchy, arm, fingers, spine and jaw. I then wrote a procedure to parent all the separate hierarchies together. I hoped that by having the joint chains made in separate steps it would also help explain the idea of hierarchies to the user.

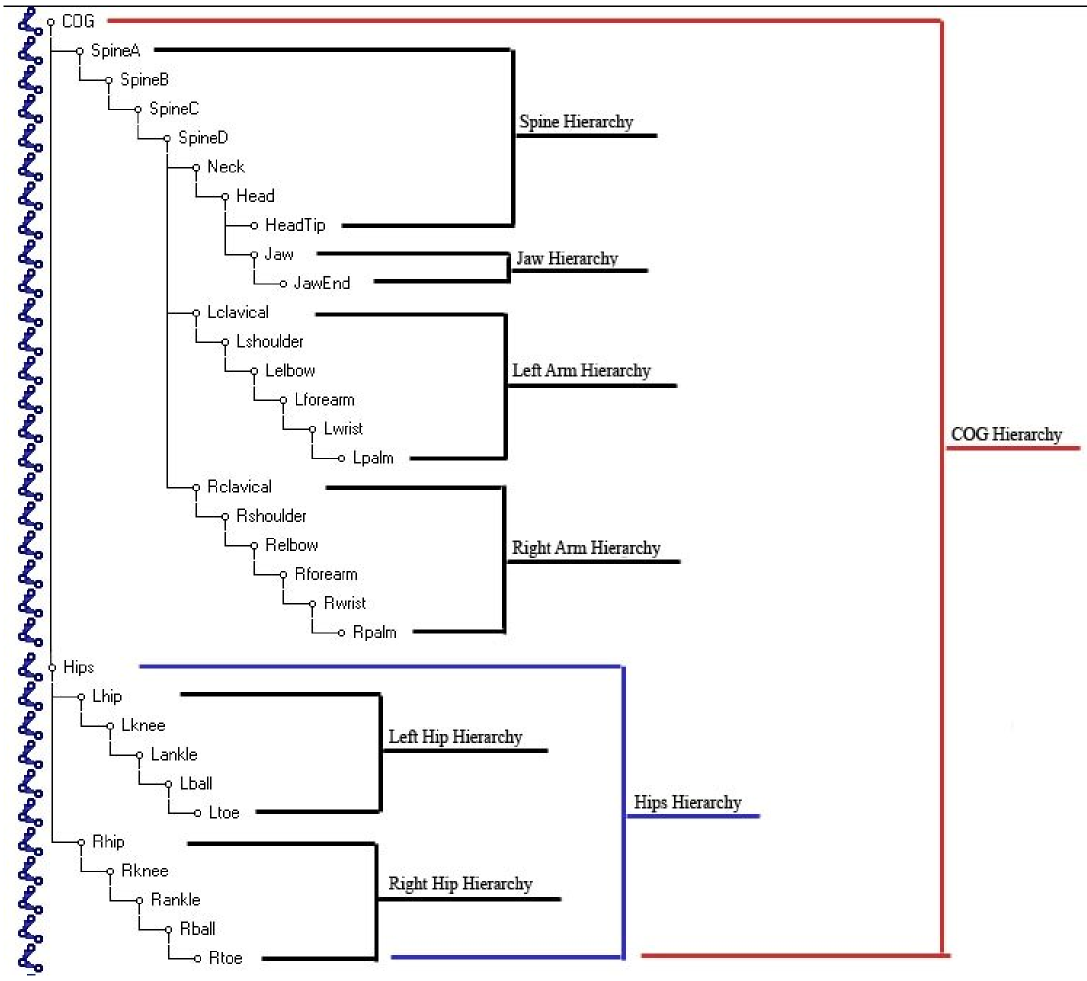
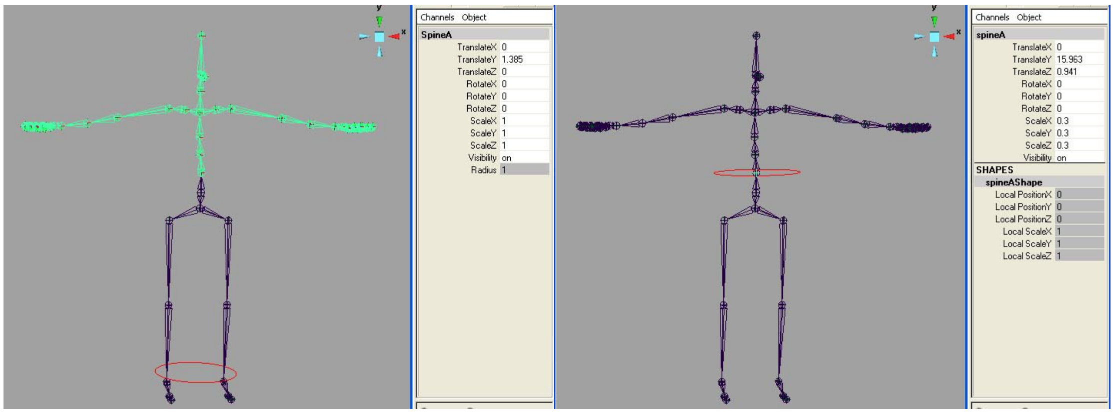


Figure 1.8 – The basic skeletal hierarchy that my script produces. Finger hierarchies are not shown.

Another challenge when making my script was how to create the control curves in the correct locations. Control curves are a way of animating the character without having to go inside the body and select individual joints and IK handles. Originally I used the co-ordinates of the joints to position my curves correctly, however, I found they were never created in the right position. I found that every joint in my skeleton did not in fact have its origin position at the world space origin but at the position of the joint above it in the hierarchy. This meant my control curves were being created according to the origin of the previous joint and therefore were never in the correct location. To solve this problem I used the co-ordinates of the original locators as my markers for positions of the joints in world space. As the locators have been made separate to each other and were in no form of hierarchy they took their position from the origin.



SpineA	
TranslateX	0
TranslateY	1.385
TranslateZ	0
RotateX	0
RotateY	0
RotateZ	0
ScaleX	1
ScaleY	1
ScaleZ	1
Visibility	on
Radius	1

spineA	
TranslateX	0
TranslateY	15.963
TranslateZ	0.941
RotateX	0
RotateY	0
RotateZ	0
ScaleX	0.3
ScaleY	0.3
ScaleZ	0.3
Visibility	on
SHAPES	
spineAShape	
Local PositionX	0
Local PositionY	0
Local PositionZ	0
Local ScaleX	1
Local ScaleY	1
Local ScaleZ	1

Figure 1.9 – Although SpineA (joint) and spineA (locator) appear to be in the same location, they both display very different co-ordinates causing the red control curve to be made in two different positions.

The global scale, rotate and translate curve (globalSRT) is used to position the whole character and rig in a scene. I wanted to have this curve surrounding the character's feet as a base. This meant there was no locator to specify the origin of the curve and I could not always rely on the character being positioned at the origin. I also needed a way to determine the radius of the curve so that for each case it would be large enough to surround the feet. I therefore used the x and z position from the cog locator and the y position from the toe locator as the cog is the centre of gravity and will always be positioned central to the rig and the toe locator will always by at the lowest point on the floor. To find the radius I took the distance from the origin to the x position of the toe locator and then added a constant value of two units so the curve would sit around the feet. The final code looks as follows:

```
//gets xyz translate values for the toe locator and returns as an array
float $lankle[] = `getAttr lankle.translate`;
//gets xyz translate values for the toe locator and returns as an array
float $ltoe[] = `getAttr ltoe.translate`;
//gets xyz translate values for the cog locator and returns as an array
float $cog[] = `getAttr cog.translate`;
//creates a circle with normal aligned with Y axis, centre point x value of the
cog locator, y value of ltoe locator and z value of cog locator, radius x value
of ltoe locator +2 and name GlobalSRT
circle -nr 0 1 0 -c $cog[0] $ltoe[1] $cog[2] -r ($ltoe[0]+2) -n GlobalSRT;
//centres the pivot
xform -cp;
```

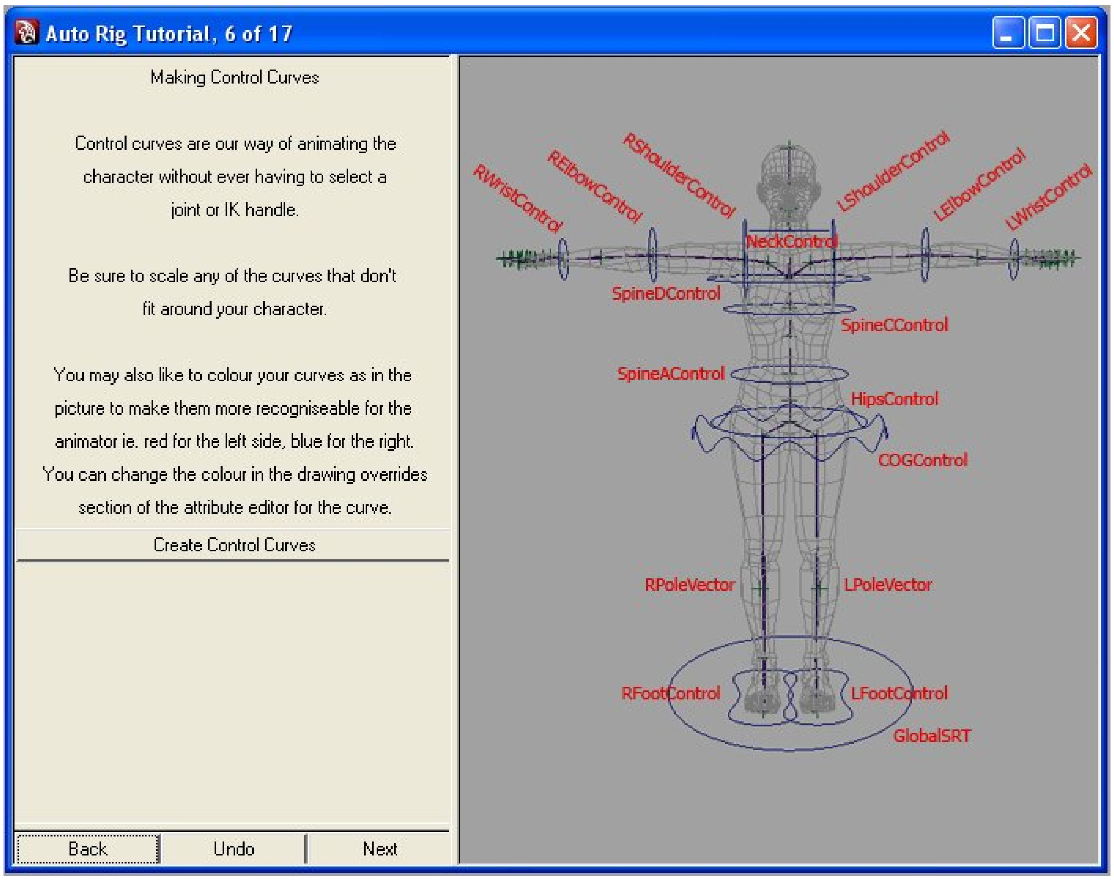
After overcoming the problems of creating the skeleton and setting up the control curves, the task of creating the rig was simple. This was down to being able to use Maya to perform a function and then use the script editor to view the code that made that function happen. I took great advantage of this and used it to create the majority of the rig setup. The important factor when creating the rigging section of the script was to break down each setup task into a separate procedure so that later on the rig could be reproduced step by step. I would have a procedure for creating the IK handles and another for setting the driven keys, this way both could be called separately and explained individually. This is where my rigging script would differ from those I had tested that created an entire rig setup with the push of one button. I needed this to be a step by step process to go alongside my tutorial.

I finally commented my code so that users who were interested would be able to understand it. Although my code may not be the most efficiently written or use the most technical of procedures, I found it more important that the direction the code lead you through the rigging process was laid out in the same steps as the tutorial.

Interface

The graphical user interface (GUI) for my script was always going to be a very important component as I had not been overly impressed with some of the others I had tested. I knew that making my script easy to use as well as getting across all the information and having useful pictures was going to be a key factor in marketing my product. None of the auto rigs I had looked at had made use of images, but this was something that I felt was most important to help the user understand what the script was doing.

The layout of my first GUI was 50% text, 50% image with a button that would call a procedure and then open the next window. This worked on a practical level, but for user friendliness I felt it was lacking so I added "back", "next" and "undo" buttons to aid simplicity of use. I created images with beginners in mind and included pictures of icons to effectively aid understanding and help the user become familiar with the appropriate tools.


Auto Rig Tutorial, 6 of 17

Making Control Curves

Control curves are our way of animating the character without ever having to select a joint or IK handle.

Be sure to scale any of the curves that don't fit around your character.

You may also like to colour your curves as in the picture to make them more recogniseable for the animator ie. red for the left side, blue for the right. You can change the colour in the drawing overrides section of the attribute editor for the curve.

Figure 1.10 – A window from the GUI. The layout is basic with a large labelled image, informative text, creation button as well as back, undo and next buttons.

Overall making the GUI was one of the most time consuming and frustrating parts of the project. This is due to the fact that MEL's window function leaves a lot to be desired and can be quite confusing. I also found that transferring my script between Windows and Linux machines would slightly alter the layout of my GUI. Nevertheless I did finally overcome this, although much of the time felt like pure trial and error to achieve the look I wanted.

Testing and Feedback

The testing stage filled me with great anticipation as it was a chance for others to see what I had been doing and give me their opinions. I obviously knew my tool very well, but how would others take to it?

I had students who knew about rigging and more importantly students who were unconfident beginners at rigging, to test my script. I was looking for their opinions on the overall ease of use of this rig and script and specifically on the clarity of the language and the usefulness of the tool. The overall feedback I received was positive with many helpful suggestions on areas that could be improved. Testing my script instantly showed me points of weakness previously unnoticed. Some problems found and suggestions made during the testing stage are below.

- Difficult to get locators into a 45 degree T-pose from the default 90 degree. You have to move every finger locator into position and this is tedious. There should be a way of moving/rotating the arm as a whole, or different start pose setups.
- The images could be improved by including labels to important points of interest. For example labelling of the locator names when shown inside the character mesh.
- Some locators could be better named for beginners, i.e. cog, for people who don't know what it stands for and lclavicle, for people who don't know where the clavicle bone is.
- The control curve for the COG joint should be bigger to distinguish it from the nearby hips curve.
- At the end of each text window add "You can test this by….", so users can check what they have just done.
- Terminology such as "delete the history" may not be understood by beginners.
- Accidentally clicking buttons twice was a problem. Buttons should be deactivated after use.

The feedback for the language and understanding of the tutorial's text was good and everyone seemed to be able to follow the tutorial without getting stuck on technical terminology.

I arranged a tutorial with Henry Bush who teaches MART to get his opinion on whether my script would be suitable for first years. The feedback I received was very positive indeed and Henry commented that such a tutorial, designed to further the understanding of rigging for beginners was going to be of great benefit. Overall the use of language was good though some areas needed further explanation.

Modifications

After testing I implemented the changes suggested. I added two radio buttons which act as on/off switches, to give the user the choice of 90 or 45 degree arms. I also made sure you could not create the locators until you had chosen one of these options.

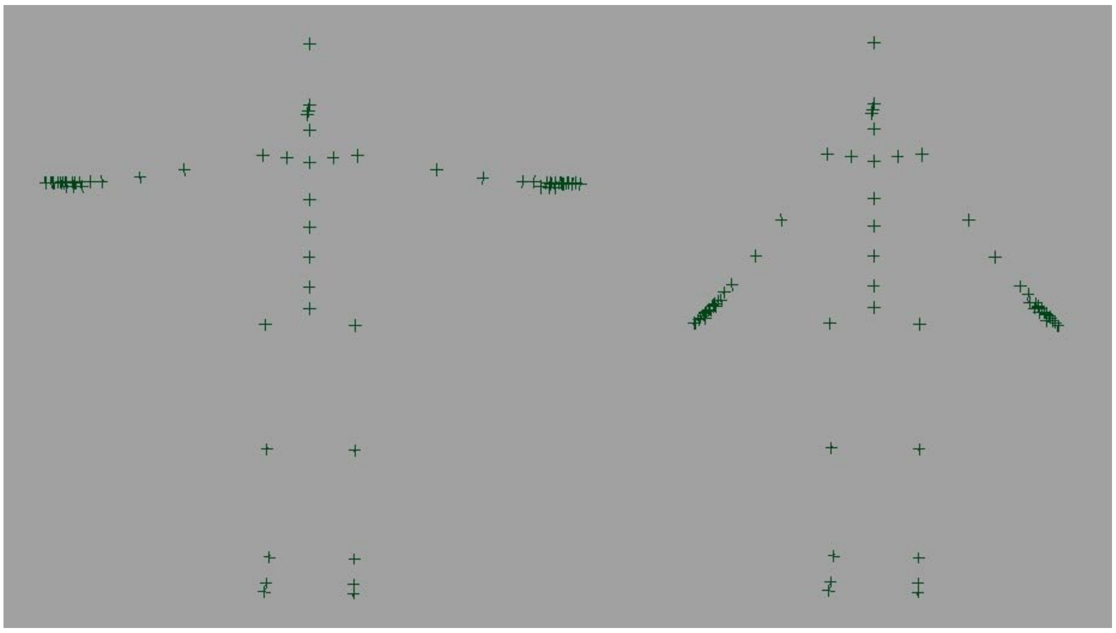


Figure 1.11 – Locators can be created in the two most popular rigging poses, 90 and 45 degree T-pose.

I changed my code so buttons could only be pressed once before being deactivated but also made sure that pressing undo would reactivate the button.

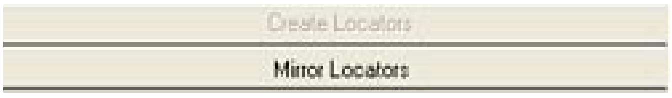


Figure 1.12 – An example of a button that has been pressed and a button waiting to be pressed.

Many images were remade or modified to add captions and helpful labels which I feel added greatly to the overall understanding of the tutorial.

At the end of each window’s text I added information about how to test what was being set up at each stage. This really helped as the user would no longer just be reading and pushing buttons but using the software and seeing how things worked. Hopefully this would help give users more confidence when building their own rig in future.

Overall I modified the language and added a few better explanations of terms like “deleting history”, as well as making the COG curve bigger as requested and renaming the original “cog” joint to “root” to make things clearer.

Future Work

One of the exciting aspects of this project is that there is always room for further development. Originally when first drafting my idea I had planned to give the user more options during the rigging process similar to some of the auto rigs I had tested. I would have liked to have given users the choice of IK or FK arms, or multiple spine joints. However, there was just not enough time to implement this, which is not necessarily a bad thing because had

I given users more options, the less the script becomes about the tutorial. Although I feel this is a slight deviation from my initial concept, it in no way detracts from the end product.

Another feature I would have liked to implement was the option of turning the tutorial on and off so more confident riggers would not need to continue through so many steps in the process. However, as I have worked through the project the tutorial has become so integrated into the rigging script that I feel it would be pointless to separate the two; this is not after all an auto rig designed to create every possible rig, but a learning aid.

If I had more time I would like to redesign the GUI of my script and this is something I intend to implement myself later. The MEL GUI feature somewhat limited my creativity when it came to the layout design stage and I feel my design could be improved. Currently a different window is created every time “next” or “back” is clicked. I would like to have implemented a single window system where only one window is drawn with the text and image updating depending on the button pressed. Tabs would also have been a nice feature so users can immediately see the stages of the process and go to specific areas in which they are interested.

Conclusion

I am delighted with the tutorial tool that I have made. As my first attempt at using MEL properly I am pleased to have created a product that works and can be used in future to help people learn and gain more confidence in rigging. This project has improved my confidence in using MEL and I would definitely hope to use this again for my major project and in the future. I have also learnt more about rigging and although this is an area in which I have always felt confident, this project has forced me to re-evaluate everything I know, to look at concepts from a beginner’s point of view and to always think “Why am I doing this?”, “How does this work?” I now have an increased understanding of the process of rigging and hopefully my script will allow others to achieve similar success.

References

Maraffi, Chris, *Maya Character Creation – Modeling and Animation Controls*, (New Riders Publishing, United States, 2004)

Meade, Tom and Arima, Shinsaku *The Complete Reference Maya 6*, (Osborne Publishing, United States, 2004)

Ritchie, K, Callery, J and Biri, K *The Art of Rigging Volume 1,* (CGToolkit, United States, 2005)

Websites

2005 Quick MEL GUI [online] Available from:
http://accad.osu.edu/~xchang/753/hw3/mel_tutorial.html

McCullough, W, 2006 Shanes Auto Rig – Animation Mel Scripts for Maya [online] Available from:
http://www.highend3d.com/maya/downloads/mel_scripts/animation/4049.html

2006 NCCA | National Centre for Computer Animation [online] Available from:
http://ncca.bournemouth.ac.uk/gallery/

2007 Morten Moen [online] Available from:
http://www.morten-moen.net/tutorials/MEL-Windows/WindowShopping.html

2007 Side Effects Software Announces Houdini 9.1 Public Beta [online] Available from:
http://www.sidefx.com/index.php?option=com_content&task=view&id=470&Itemid=55

2005 Simply CG Network :: View topic – AutoRig with Instructions [online] Available from:
http://www.simplycg.net/viewtopic.php?p=3682

Bush, H, 2006 MART Notes Index [online] Available from:
http://www.spookypeanut.co.uk/

2007 Film & Video | Riggers Gone Wild! [online] Available from:
http://www.studiodaily.com/filmandvideo/technique/casestudies/4467.html

Maya 8.0 Help File Documentation

Appendix

MEL Script

```
//Innovations Project 2007
//Automated Rigging Tutorial
//Rachael Hender

int $val = 0; //declare global variable

string $path;
$path = `getenv "HOME"`; //locate users home directory so images can be sourced
string $folder = $path+"/maya/8.0/prefs/icons/Images/";

proc begin() //displays the 1st window
{
	global string $folder;
	string $image = $folder+"picture00.bmp"; //find the image
	if (`window -exists Start`)
		deleteUI Start; //if the window previously exists, delete it
	if (`window -exists AutoRigTutorial`)
		deleteUI AutoRigTutorial;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 1 of 17"
AutoRigTutorial; //window layout
	paneLayout -cn "vertical2" -ps 2 60 100;
	columnLayout -columnAttach "both" 0 -columnWidth 275 -rowSpacing 5;
	text -label "Auto Rig Tutorial"; //text
	text -label "by Rachael Hender";
	text -label "Rigging is the process of adding a skeleton";
	text -label "and controls to a character that will be used";
	text -label "for animation.";
	text -label "This auto rig script aims to explain the basics";
	text -label "of rigging whilst being a quick and effective way";
	text -label "of creating a simple biped character rig.";
	text -label "Click \"Start\" to begin.";
	button -label "Start" -command "Next"; //create a button
	setParent ..;/*columnLayout*/
	image -image $image;
	showWindow AutoRigTutorial; //show the window
}

proc Next() //displays the 2nd window
{
	global string $folder;
	string $image = $folder+"picture01.bmp";
	if (`window -exists TheSkeleton`)
	deleteUI TheSkeleton;
	if (`window -exists AutoRigTutorial`)
	deleteUI AutoRigTutorial;
	if (`window -exists Start`)
		deleteUI Start;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 2 of 17"
Start;
	paneLayout -cn "left3" -ps 1 40 94 -ps 2 60 100;
	columnLayout -columnAttach "both" 0 -columnWidth 275 -rowSpacing 5;
	text -label "To start rigging make sure your character is";
	text -label "positioned at the origin with its axis of symmetry";
	text -label "aligned with Maya's YZ plane. This is to allow";
	text -label "for mirroring of joints later on.";
	text -label "This is also a good time to clean up your scene";
	text -label "by deleting your characters history,";
	text -label "(Edit>Delete by Type>History)";
	text -label "deleting unwanted objects and making sure";
	text -label "everything in the scene is named appropriately.";
	text -label "A good rigging pose for your chracter is also";
	text -label "important. This is normally a T-pose or some";
	text -label "varient of it with arms to the side and knees";
	text -label "slightly bent.";
	text -label "You are now ready to begin.";
	setParent ..;
	image -image $image;
	setParent ..;
	rowLayout -numberOfColumns 2
```

```
		-columnWidth2 138 138
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0;
		button -label "Back" -command begin;
		button -label "Next" -command FirstWindow;
	setParent..;
	showWindow Start;
}

proc FirstWindow() //displays the 3rd window
{
	global string $folder;
	string $image = $folder+"picture02.bmp";
	if (`window -exists Joints`)
	deleteUI Joints;
	if (`window -exists Start`)
	deleteUI Start;
	if (`window -exists TheSkeleton`)
		deleteUI TheSkeleton;
	window -tlc 395 575 -widthHeight 700 600 -title "Auto Rig Tutorial, 3 of 17"
TheSkeleton;
	paneLayout -cn "left3" -ps 1 50 96 -ps 2 50 100;
	columnLayout -columnAttach "both" 0 -columnWidth 355;
	text -label "Creating a Skeleton for the Character";
	text -label "Think of it as a simplified version of a human skeleton. It will";
	text -label "control the movement of the character and is made of \"joints\"";
	text -label "and \"bones\". A joint is a point where a character will be able";
	text -label "to bend and bones connect joints together.";
	iconTextButton -width 100 -style "iconAndTextHorizontal" //show the joint icon
	-image "kinJoint.xpm" -label "Joint tool in Maya";
	text -label "First we must add joints to all the places in the body where";
	text -label "the character might need to bend. Clicking \"Create Locators\"";
	text -label "will make a series of locators representing everywhere in the";
	text -label "left side of the body where you would want a joint, i.e. knee,";
	text -label "neck, elbow. You can create the locators in either a 90 or 45";
	text -label "degree T-pose.";
	text -label "Move these locators so they fit inside your character at the";
	text -label "places specified by their names. It might be useful to use";
	text -label "orthographic views to do this as well as wireframe mode.";
	text -label "An example of the locators in correct positions is shown";
	text -label "opposite";
	radioCollection; //radio buttons give the user a choice
	radioButton -label "90 degree T-pose" -onc "val"; //if on call procedure val
	radioButton -label "45 degree T-pose" -onc "val2"; //if on call procedure val2
	button -label "Create Locators" -command "createLocators" createlocators;
	text -label "Once you have the locators in position, click";
	text -label "\"Mirror Locators\" to mirror them to the other side.";
	button -label "Mirror Locators" -command "mirrorLocators" mirrorlocators;
	text -label "Finally click \"Rename Locators\" to rename them for the right";
	text -label "hand side.";
	button -label "Rename Locators" -command "renameLocators" renamelocators;
	setParent ..;
	image -image $image;
	setParent ..;
	rowLayout -numberOfColumns 3 //layout for back, undo and next buttons
		-columnWidth3 116 116 116
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAlign  3 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0
		-columnAttach 3 "both" 0;
		button -label "Back" -command Next;
		//if pressed recall 'Next' procedure
		button -label "Undo" -command undo1;
		//if pressed call 'undo1' procedure
		button -label "Next" -command Joints;
		//if pressed call 'Joints' procedure
	setParent..;
	showWindow TheSkeleton;
}

proc val() //sets the value of $val to 1
```

```
{
	global int $val;
	$val = 1;
}
proc val2() //sets the value of $val to 2
{
	global int $val;
	$val = 2;
}

//separate undo functions for each so that buttons will be re-enabled
proc undo1()
{undo;button -e -en 1 createlocators;button -e -en 1 mirrorlocators;button -e -en 1
renamelocators;}
proc undo2()
{undo;button -e -en 1 makeleg;}
proc undo3()
{undo;button -e -en 1 parentjoints;button -e -en 1 mirrorjoints;button -e -en 1
renamejoints;}
proc undo4()
{undo;button -e -en 1 controlcurves;}
proc undo5()
{undo;button -e -en 1 preferredangle;}
proc undo6()
{undo;button -e -en 1 firstik;}
proc undo7()
{undo;button -e -en 1 setuplegs;button -e -en 1 parenting;}
proc undo8()
{undo;button -e -en 1 setuparms;}
proc undo9()
{undo;button -e -en 1 armparenting;}
proc undo10()
{undo;button -e -en 1 setupfingers;}
proc undo11()
{undo;button -e -en 1 setupbody;}
proc undo12()
{undo;button -e -en 1 bodyparenting;}
proc undo13()
{undo;button -e -en 1 globalsrt;}
proc undo14()
{undo;button -e -en 1 cleanup;}

proc createLocators() //creates the locators, either 90 or 45 degree pose depending on
the value of $val
{
global int $val;
if($val == 2) //if 45 degree arms were chosen do this
{
matrix $locs[41][3] = <<2.138, 0.0, 2.437;2.138, 0.0, 1.485;        //ltoe, lball
                      2.138, 0.653, 0.0;2.138, 6.509, 0.513;        //lankle,lknee
                      2.138, 12.637, 0.299;0.0, 13.487, 0.684;      //lhip, hips
                      0.0, 14.578, 0.941;0.0, 15.963, 0.941;        //root,spineA
                      0.0, 17.27, 0.385;0.0, 18.544, 0.0;           //spineB,spineC
                      0.0, 20.246, 0.0;0.0, 21.717, 0.0;            //spineD,neck
                      0.0, 22.825, 0.0;0.0, 25.525, 0.0;            //head,headEnd
                      0.0, 22.382, 1.504;1.07, 20.445, 0.0;         //jawEnd,lclavicle
                      2.191, 20.54, 0.0;4.393, 17.538,0.0;          //lshoulder,lelbow
                      5.687, 15.785, 0.0;6.891, 14.384,0.0;         //lforearm,lwrist
                      7.276, 14.013, 0.0;7.275, 13.738, 0.714;      //lpalm,lthumb1
                      7.499, 13.523, 1.041;7.684, 13.345, 1.256;    //lthumb2,lthumb3
                      7.74, 13.473, 0.444;8.142, 13.087, 0.444;     //lindex1,lindex2
                      8.389, 12.807, 0.444;8.678, 12.572, 0.444;    //lindex3,lindex4
                      7.751, 13.462, 0.1;8.148, 13.081, 0.1;        //lmiddle1,lmiddle2
                      8.471, 12.771, 0.1;8.733, 12.519, 0.1;        //lmiddle3,lmiddle4
                      7.808, 13.407, -0.235;8.102, 13.125, -0.235; //lring1,lring2
                      8.4, 12.839, -0.235;8.641, 12.607, -0.235;    //lring3,lring4
                      7.711, 13.501, -0.562;7.991, 13.232, -0.562; //lbaby1,lbaby2
                      8.222, 13.01, -0.562;8.394, 12.845, -0.562    //lbaby3,lbaby4
                      ;0.0, 22.56, 0.786>>;                         //jaw

	for($i=0; $i<41; $i++)
	{
		spaceLocator; //make the 41 locators needed for half the body
	}
	for($j=1; $j<=41; $j++)
```

```
        {
setAttr ("locator"+$j+".translate") $locs[$j-1][0] $locs[$j-1][1] $locs[$j-1][2];
setAttr ("locator"+$j+".scale") 0.3 0.3 0.3; //move each one to the position in the
matrix
}
}
if($val == 1) //if 90 degree arms were chosen do this
{
matrix $locs[41][3] = <<2.138, 0.0, 2.437;2.138, 0.0, 1.485;       //ltoe, lball
                        2.138, 0.653, 0.0;2.138, 6.509, 0.513;       //lankle,lknee
                        2.138, 12.637, 0.299;0.0, 13.487, 0.684;     //lhip, hips
                        0.0, 14.578, 0.941;0.0, 15.963, 0.941;       //root,spineA
                        0.0, 17.27, 0.385;0.0, 18.544, 0.0;          //spineB,spineC
                        0.0, 20.246, 0.0;0.0, 21.717, 0.0;           //spineD,neck
                        0.0, 22.825, 0.0;0.0, 25.525, 0.0;           //head,headEnd
                        0.0, 22.382, 1.504;1.07, 20.445, 0.0;        //jawEnd,lclavicle
                        2.191, 20.54, 0.0;5.858, 19.9, 0.0;          //lshoulder,lelbow
                        8.006, 19.532, 0.00;9.845, 19.355, 0.005;    //lforearm,lwrist
                        10.38, 19.354, 0.005;10.569, 19.155, 0.715;  //lpalm,lthumb1
                        10.88, 19.155, 1.042;11.136, 19.155,1.257;   //lthumb2,lthumb3
                        11.088, 19.286, 0.444;11.646, 19.286, 0.444; //lindex1,lindex2
                        12.05, 19.286, 0.444;12.389, 19.286, 0.444;  //lindex3,lindex4
                        11.104, 19.286, 0.101;11.654, 19.286, 0.101; //lmiddle1,lmiddle2
                        12.102, 19.286, 0.101;12.465, 19.286, 0.101; //lmiddle3,lmiddle4
                        11.183, 19.286, -0.234;11.59, 19.286, -0.234;//lring1,lring2
                        12.003, 19.286, -0.234;12.338, 19.286, -0.234;//lring3,lring4
                        11.048, 19.286, -0.561;11.436, 19.286, -0.561;//lbaby1,lbaby2
                        11.756, 19.286, -0.561;11.995, 19.286, -0.561;//lbaby3,lbaby4
                        0.0, 22.56, 0.786>>;                         //jaw

        for($i=0; $i<41; $i++)
        {
                spaceLocator;
        }
        for($j=1; $j<=41; $j++)
        {
setAttr ("locator"+$j+".translate") $locs[$j-1][0] $locs[$j-1][1] $locs[$j-1][2];
        setAttr ("locator"+$j+".scale") 0.3 0.3 0.3;
        }
}


rename"locator1""ltoe"; rename"locator2""lball"; rename"locator3""lankle";
rename"locator4""lknee"; rename"locator5""lhip"; //leg locators
rename"locator6""hips"; rename"locator7""root"; rename"locator8""spineA";
rename"locator9""spineB"; rename"locator10""spineC"; rename"locator11""spineD";
rename"locator12""neck"; rename"locator13""head"; rename"locator14""headEnd"; //body
locators
rename"locator15""jawend"; rename"locator41""jaw"; //jaw locators
rename"locator16""lclavicle"; rename"locator17""lshoulder"; rename"locator18""lelbow";
rename"locator19""lforearm"; rename"locator20""lwrist"; rename"locator21""lpalm";
//arm locators
rename"locator22""lthumb1"; rename"locator23""lthumb2"; rename"locator24""lthumb3";
//thumb locators
rename"locator25""lindex1"; rename"locator26""lindex2"; rename"locator27""lindex3";
rename"locator28""lindex4"; //index finger locators
rename"locator29""lmiddle1"; rename "locator30""lmiddle2";
rename"locator31""lmiddle3"; rename"locator32""lmiddle4"; //middle finger locators
rename"locator33""lring1"; rename"locator34""lring2"; rename"locator35""lring3";
rename"locator36""lring4"; //ring finger locators
rename"locator37""lbaby1"; rename"locator38""lbaby2"; rename"locator39""lbaby3";
rename"locator40""lbaby4"; //baby finger locators
button -e -en 0 createlocators; //disable the button after the procedure has been
called
}

proc mirrorLocators() //mirror locators to the other side
{
        group -n legGroup ltoe lball lankle lknee lhip; xform -os -piv 0 0 0;
        //create a leg group
        group -n armGroup lbaby4 lbaby3 lbaby2 lbaby1 lring4 lring3 lring2 lring1
lmiddle4 lmiddle3 lmiddle2    lmiddle1 lindex4 lindex3 lindex2 lindex1 lthumb3 lthumb2
lthumb1 lpalm lwrist lforearm lelbow lshoulder lclavicle; xform -os -piv 0 0 0;
//create an arm group
```

```
        select -r legGroup; duplicate -rr -renameChildren; scale -r -1 1 1; //duplicate
leg group locators
        select -r armGroup; duplicate -rr -renameChildren; scale -r -1 1 1; //duplicate
arm group locators
        button -e -en 0 mirrorlocators; //disable button after procedure
}

proc renameLocators() //rename the new right side locators
{
        rename "ltoe1" "rtoe";rename "lball1" "rball";rename "lankle1" "rankle";rename
"lknee1" "rknee";rename       "lhip1" "rhip"; //leg locators
        rename "lclavicle1" "rclavicle";rename "lshoulder1" "rshoulder";rename
"lelbow1" "relbow";rename     "lforearm1" "rforearm";rename "lwrist1" "rwrist";rename
"lpalm1" "rpalm"; //arm locators
        rename "lthumb6" "rthumb1";rename "lthumb5" "rthumb2";rename "lthumb4"
"rthumb3"; //thumb locators
        rename "lindex8" "rindex1";rename "lindex7" "rindex2";rename "lindex6"
"rindex3";rename "lindex5"    "rindex4"; //index finger locators
        rename "lmiddle8" "rmiddle1";rename "lmiddle7" "rmiddle2";rename "lmiddle6"
"rmiddle3";rename      "lmiddle5" "rmiddle4"; //middle finger locators
        rename "lring8" "rring1";rename "lring7" "rring2";rename "lring6"
"rring3";rename "lring5" "rring4";
        //ring finger locators
        rename "lbaby8" "rbaby1";rename "lbaby7" "rbaby2";rename "lbaby6"
"rbaby3";rename "lbaby5" "rbaby4";
        //baby finger locators
        button -e -en 0 renamelocators; //disable button

}

proc Joints() //display the 4th window
{
        global string $folder;
        string $image = $folder+"picture03.bmp";
        if (`window -exists Joints`)
               deleteUI Joints;
        if (`window -exists TheSkeleton`)
               deleteUI TheSkeleton;
        if (`window -exists FinishingSkeleton`)
               deleteUI FinishingSkeleton;
        window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 4 of 17"
Joints;
        paneLayout -cn "left3" -ps 1 40 94 -ps 2 60 100;
        columnLayout -columnAttach "both" 0 -columnWidth 280 -rowSpacing 5;
        text -label "Making Joints";
        text -label "When making joints it is important to think about";
        text -label "the hierarchy they are in. We want joints higher";
        text -label "in the hierarchy to control those below it like";
        text -label "in our own body. You can't move your shoulder";
        text -label "without effecting your elbow below it. It is usual";
        text -label "to have the root of the skeleton placed at the";
        text -label "centre of gravity. This is usually just above the";
        text -label "hips.";
        text -label "Clicking \"Make Joints\" will create separate joint";
        text -label "chains that will later be parented together to form ";
        text -label "the joint hierarchy. Joint chains will be created";
        text -label "for the leg, spine, jaw, arm, thumb as well as";
        text -label "index, middle, ring and baby fingers.";
        button -label "Make Joints" -command makeLeg makeleg;
        setParent ..;
        image -image $image;
        setParent ..;
        rowLayout -numberOfColumns 3
               -columnWidth3 93 93 93
               -columnAlign  1 "center"
               -columnAlign  2 "center"
               -columnAlign  3 "center"
               -columnAttach 1 "both" 0
               -columnAttach 2 "both" 0
               -columnAttach 3 "both" 0;
               button -label "Back" -command FirstWindow;
               button -label "Undo" -command undo2;
               button -label "Next" -command Skeleton;
        setParent..;
        showWindow Joints;
```

```
}

proc makeLeg() //making the leg joints
{
	float $hips[] = `getAttr hips.translate`; //get the x,y,z positions of the
following locators
	float $lhip[] = `getAttr lhip.translate`;
	float $lknee[] = `getAttr lknee.translate`;
	float $lankle[] = `getAttr lankle.translate`;
	float $lball[] = `getAttr lball.translate`;
	float $ltoe[] = `getAttr ltoe.translate`;
	joint -p $hips[0] $hips[1] $hips[2]; //create joints at the positons taken from
the locators
	joint -p $lhip[0] $lhip[1] $lhip[2];
	joint -p $lknee[0] $lknee[1] $lknee[2];
	joint -p $lankle[0] $lankle[1] $lankle[2];
	joint -p $lball[0] $lball[1] $lball[2];
	joint -p $ltoe[0] $ltoe[1] $ltoe[2];
	select -cl;

	makeSpine; //call the next procedure
}

proc makeSpine() //make the spine joints
{
	float $root[] = `getAttr root.translate`;
	float $spineA[] = `getAttr spineA.translate`;
	float $spineB[] = `getAttr spineB.translate`;
	float $spineC[] = `getAttr spineC.translate`;
	float $spineD[] = `getAttr spineD.translate`;
	float $neck[] = `getAttr neck.translate`;
	float $head[] = `getAttr head.translate`;
	float $headEnd[] = `getAttr headEnd.translate`;
	joint -p $root[0] $root[1] $root[2];
	joint -p $spineA[0] $spineA[1] $spineA[2];
	joint -p $spineB[0] $spineB[1] $spineB[2];
	joint -p $spineC[0] $spineC[1] $spineC[2];
	joint -p $spineD[0] $spineD[1] $spineD[2];
	joint -p $neck[0] $neck[1] $neck[2];
	joint -p $head[0] $head[1] $head[2];
	joint -p $headEnd[0] $headEnd[1] $headEnd[2];
	select -cl  ;

	makeArm; //call the next procedure
}

proc makeArm() //makes the arm joints
{
	float $lclavicle[] = `getAttr lclavicle.translate`;
	float $lshoulder[] = `getAttr lshoulder.translate`;
	float $lelbow[] = `getAttr lelbow.translate`;
	float $lforearm[] = `getAttr lforearm.translate`;
	float $lwrist[] = `getAttr lwrist.translate`;
	float $lpalm[] = `getAttr lpalm.translate`;
	joint -p $lclavicle[0] $lclavicle[1] $lclavicle[2];
	joint -p $lshoulder[0] $lshoulder[1] $lshoulder[2];
	joint -p $lelbow[0] $lelbow[1] $lelbow[2];
	joint -p $lforearm[0] $lforearm[1] $lforearm[2];
	joint -p $lwrist[0] $lwrist[1] $lwrist[2];
	joint -p $lpalm[0] $lpalm[1] $lpalm[2];
	select -cl;

	makeThumb; //call the next procedure
}

proc makeThumb() //makes the thumb joints
{
	float $lthumb1[] = `getAttr lthumb1.translate`;
	float $lthumb2[] = `getAttr lthumb2.translate`;
	float $lthumb3[] = `getAttr lthumb3.translate`;
	joint -p $lthumb1[0] $lthumb1[1] $lthumb1[2];
	joint -p $lthumb2[0] $lthumb2[1] $lthumb2[2];
	joint -p $lthumb3[0] $lthumb3[1] $lthumb3[2];
	select -cl;
```

```
        makeIndex; //calls the next procedure
}

proc makeIndex() //make the index finger joints
{
        float $lindex1[] = `getAttr lindex1.translate`;
        float $lindex2[] = `getAttr lindex2.translate`;
        float $lindex3[] = `getAttr lindex3.translate`;
        float $lindex4[] = `getAttr lindex4.translate`;
        joint -p $lindex1[0] $lindex1[1] $lindex1[2];
        joint -p $lindex2[0] $lindex2[1] $lindex2[2];
        joint -p $lindex3[0] $lindex3[1] $lindex3[2];
        joint -p $lindex4[0] $lindex4[1] $lindex4[2];
        select -cl;

        makeMiddle; //call the next procedure
}

proc makeMiddle() //make the middle finger  joints
{
        float $lmiddle1[] = `getAttr lmiddle1.translate`;
        float $lmiddle2[] = `getAttr lmiddle2.translate`;
        float $lmiddle3[] = `getAttr lmiddle3.translate`;
        float $lmiddle4[] = `getAttr lmiddle4.translate`;
        joint -p $lmiddle1[0] $lmiddle1[1] $lmiddle1[2];
        joint -p $lmiddle2[0] $lmiddle2[1] $lmiddle2[2];
        joint -p $lmiddle3[0] $lmiddle3[1] $lmiddle3[2];
        joint -p $lmiddle4[0] $lmiddle4[1] $lmiddle4[2];
        select -cl;

        makeRing; //call the next procedure
}

proc makeRing() //make the ring finger joints
{
        float $lring1[] = `getAttr lring1.translate`;
        float $lring2[] = `getAttr lring2.translate`;
        float $lring3[] = `getAttr lring3.translate`;
        float $lring4[] = `getAttr lring4.translate`;
        joint -p $lring1[0] $lring1[1] $lring1[2];
        joint -p $lring2[0] $lring2[1] $lring2[2];
        joint -p $lring3[0] $lring3[1] $lring3[2];
        joint -p $lring4[0] $lring4[1] $lring4[2];
        select -cl;

        makeBaby; //call the next procedure
}

proc makeBaby() //make the baby finger joints
{
        float $lbaby1[] = `getAttr lbaby1.translate`;
        float $lbaby2[] = `getAttr lbaby2.translate`;
        float $lbaby3[] = `getAttr lbaby3.translate`;
        float $lbaby4[] = `getAttr lbaby4.translate`;
        joint -p $lbaby1[0] $lbaby1[1] $lbaby1[2];
        joint -p $lbaby2[0] $lbaby2[1] $lbaby2[2];
        joint -p $lbaby3[0] $lbaby3[1] $lbaby3[2];
        joint -p $lbaby4[0] $lbaby4[1] $lbaby4[2];
        select -cl;

        makeJaw; //call the next procedure
}

proc makeJaw() //make the jaw joitns
{
        float $jawend[] = `getAttr jawend.translate`;
        float $jaw[] = `getAttr jaw.translate`;
        joint -p $jaw[0] $jaw[1] $jaw[2];
        joint -p $jawend[0] $jawend[1] $jawend[2];
        select -cl  ;
        button -e -en 0 makeleg; //disable the original button that was pressed
}

proc Skeleton() //creates the 5th window
{
	global string $folder;
	string $image = $folder+"picture04.bmp";
	if (`window -exists Joints`)
		deleteUI Joints;
	if (`window -exists FinishingSkeleton`)
		deleteUI FinishingSkeleton;
	if (`window -exists ControlCurves`)
		deleteUI ControlCurves;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 5 of 17"
FinishingSkeleton;
	paneLayout -cn "left3" -ps 1 60 94 -ps 2 40 100;
	columnLayout -columnAttach "both" 0 -columnWidth 420 -rowSpacing 5;
text -label "Finishing the Skeleton";
text -label "Each joint chain must be parented into the hierarchy. The COG is the
joint";
text -label "at the base of the spine chain.Every other joint will be parented below";
text -label "that one. The joint hierarchy should look like the picture opposite.";
button -label "Parent Joints" -command "parentJoints" parentjoints;
text -label "Maya allows us to mirror joints over to the other side.";
iconTextButton -width 100 -style "iconAndTextHorizontal"
-image "kinMirrorJoint_S.xpm" -label "Mirror Joint tool in Maya ";
button -label "Mirror Joints" -command "mirrorJoints" mirrorjoints;
text -label "Naming conventions are very important in rigging so things don't get";
text -label "confusing. We have similar joints on the right and left sides,
therefore";
text -label "we shall give everything on the left side the prefix \"L\" and
everything";
text -label "on the right side the prefix \"R\". Centre joints are named hips, neck,";
text -label "head, spineA,B etc.";
button -label "Rename Joints" -command "renameJoints" renamejoints;
text -label "Try looking in the outliner to see the new joints in their hierarchy.";
	setParent ..;
	image -image $image;
	rowLayout -numberOfColumns 3
		-columnWidth3 140 140 140
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAlign  3 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0
		-columnAttach 3 "both" 0;
		button -label "Back" -command Joints;
		button -label "Undo" -command undo3;
		button -label "Next" -command ControlCurves;
	setParent..;
	showWindow FinishingSkeleton;
}

proc parentJoints() //parents the joint chains together
{
	parent joint1 joint7;
	parent joint15 joint11;
	parent joint21 joint20;
	parent joint24 joint20;
	parent joint28 joint20;
	parent joint32 joint20;
	parent joint36 joint20;
	parent joint40 joint13;

	button -e -en 0 parentjoints; //disable button
}

proc mirrorJoints() //mirror arm and leg joints over to the other side
{
	mirrorJoint -mirrorYZ -mirrorBehavior joint2;
	mirrorJoint -mirrorYZ -mirrorBehavior joint15;
	button -e -en 0 mirrorjoints;
}

proc renameJoints() //rename all the joints
{
	rename "joint1" "Hips";
	rename "joint2" "Lhip";rename "joint3" "Lknee";rename "joint4" "Lankle";rename
"joint5" "Lball";rename	"joint6" "Ltoe";
```

```
	rename "joint7" "COG";rename "joint8" "SpineA";rename "joint9" "SpineB";rename 
"joint10"	"SpineC";rename "joint11" "SpineD";
	rename "joint12" "Neck";rename "joint13" "Head";rename "joint14" "HeadTip";
	rename "joint15" "Lclavicle"; rename "joint16" "Lshoulder";rename "joint17" 
"Lelbow";rename "joint18"	"Lforearm";rename "joint19" "Lwrist";rename "joint20" 
"Lpalm";
	rename "joint21" "Lthumb1";rename "joint22" "Lthumb2";rename "joint23" 
"Lthumb3";
	rename "joint24" "Lindex1";rename "joint25" "Lindex2";rename "joint26" 
"Lindex3";rename "joint27"	"Lindex4";
	rename "joint28" "Lmiddle1";rename "joint29" "Lmiddle2";rename "joint30" 
"Lmiddle3";rename "joint31"	"Lmiddle4";
	rename "joint32" "Lring1";rename "joint33" "Lring2";rename "joint34" 
"Lring3";rename "joint35" "Lring4";
	rename "joint36" "Lbaby1";rename "joint37" "Lbaby2";rename "joint38" 
"Lbaby3";rename "joint39"	"Lbaby4";
	rename "joint40" "Jaw"; rename "joint41" "JawEnd";
	rename "joint42" "Rhip";rename "joint43" "Rknee";rename "joint44" 
"Rankle";rename "joint45"	"Rball";rename "joint46" "Rtoe";
	rename "joint47" "Rclavicle"; rename "joint48" "Rshoulder";rename "joint49" 
"Relbow";rename "joint50"	"Rforearm";rename "joint51" "Rwrist";rename "joint52" 
"Rpalm";
	rename "joint53" "Rthumb1";rename "joint54" "Rthumb2";rename "joint55" 
"Rthumb3";
	rename "joint56" "Rindex1";rename "joint57" "Rindex2";rename "joint58" 
"Rindex3";rename "joint59"	"Rindex4";
	rename "joint60" "Rmiddle1";rename "joint61" "Rmiddle2";rename "joint62" 
"Rmiddle3";rename "joint63"	"Rmiddle4";
	rename "joint64" "Rring1";rename "joint65" "Rring2";rename "joint66" 
"Rring3";rename "joint67" "Rring4";
	rename "joint68" "Rbaby1";rename "joint69" "Rbaby2";rename "joint70" 
"Rbaby3";rename "joint71"	"Rbaby4";
	button -e -en 0 renamejoints;
}

proc ControlCurves() //creates the 6th window
{
	global string $folder;
	string $image = $folder+"picture05.bmp";
	if (`window -exists FinishingSkeleton`)
		deleteUI FinishingSkeleton;
	if (`window -exists ControlCurves`)
		deleteUI ControlCurves;
	if (`window -exists OrientJoints`)
		deleteUI OrientJoints;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 6 of 17" 
ControlCurves;
	paneLayout -cn "left3" -ps 1 40 94 -ps 2 60 100;
	columnLayout -columnAttach "both" 0 -columnWidth 280 -rowSpacing 5;
	text -label "Making Control Curves";
	text -label "Control curves are our way of animating the";
	text -label "character without ever having to select a";
	text -label "joint or IK handle.";
	text -label "Be sure to scale any of the curves that don't ";
	text -label "fit around your character.";
	text -label "You may also like to colour your curves to";
	text -label "make them more recogniseable for the animator";
	text -label "ie. red for the left side, blue for the";
	text -label "right. You can change the colour in the drawing";
	text -label "overrides section of the attribute editor for the";
	text -label "curve";
	button -label "Create Control Curves" -command "makeControlCurves" 
controlcurves;
	setParent ..;
	image -image $image;
	setParent..;
	rowLayout -numberOfColumns 3
		-columnWidth3 93 93 93
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAlign  3 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0
		-columnAttach 3 "both" 0;
		button -label "Back" -command Skeleton;
```

```
                button -label "Undo" -command undo4;
                button -label "Next" -command SetAngle;
        setParent..;
        showWindow ControlCurves;

}

proc makeControlCurves()
{
        //creating the GlobalSRT curve
        float $lankle[] = `getAttr lankle.translate`;
        float $ltoe[] = `getAttr ltoe.translate`;
        float $root[] = `getAttr root.translate`;
        //create a circle at the positions taken from the locators
        circle -nr 0 1 0 -c $root[0] $ltoe[1] $root[2] -r ($ltoe[0]+2) -n GlobalSRT;
        xform -cp GlobalSRT;
        //creating the COG curve
        float $root[] = `getAttr root.translate`;
        circle -nr 0 1 0 -c 0 0 0 -r 5 -n COGControl -s 20;
        //move the cv's to make the shape
        select -r COGControl.cv[2] COGControl.cv[4] COGControl.cv[6] COGControl.cv[8]
COGControl.cv[10]     COGControl.cv[12] COGControl.cv[14] COGControl.cv[16]
COGControl.cv[18] COGControl.cv[0];
        move -r 0 -3 0;
        move -r $root[0] $root[1] $root[2] COGControl;xform -cp COGControl;
        makeIdentity -apply true -t 1 -r 1 -s 1 -n 0 COGControl; //freeze
transformations

        //make the hips control curve
        float $hips[] = `getAttr hips.translate`;
        circle -nr 0 1 0 -c $hips[0] $hips[1] $hips[2] -r 4 -n HipsControl;xform -cp;

        //make the spineA control curve
        float $spineA[] = `getAttr spineA.translate`;
        circle -nr 0 1 0 -c $spineA[0] $spineA[1] $spineA[2] -r 3 -n
SpineAControl;xform -cp;

        //make the spineC control curve
        float $spineC[] = `getAttr spineC.translate`;
        circle -nr 0 1 0 -c $spineC[0] $spineC[1] $spineC[2] -r 2.5 -n
SpineCControl;xform -cp;

        //make the spineD control curve
        float $spineD[] = `getAttr spineD.translate`;
        circle -nr 0 1 0 -c $spineD[0] $spineD[1] $spineD[2] -r 3 -n
SpineDControl;xform -cp;

        //make the neck control curve
        float $neck[] = `getAttr neck.translate`;
        circle -nr 0 1 0 -c $neck[0] $neck[1] $neck[2] -r 2 -n NeckControl;xform -cp;

        //make the lshoulder control curve
        float $lshoulder[] = `getAttr lshoulder.translate`;
        circle -nr 1 0 0 -c $lshoulder[0] $lshoulder[1] $lshoulder[2] -r 1 -n
LShoulderControl;xform-cp;

        //make the lelbow control curve
        float $lelbow[] = `getAttr lelbow.translate`;
        circle -nr 1 0 0 -c $lelbow[0] $lelbow[1] $lelbow[2] -r 1 -n
LElbowControl;xform-cp;

        //make the lwrist control curve
        float $lwrist[] = `getAttr lwrist.translate`;
        circle -nr 1 0 0 -c $lwrist[0] $lwrist[1] $lwrist[2] -r 1 -n
LWristControl;xform-cp;

        //make the rshoulder control curve
        float $rshoulder[] = `getAttr rshoulder.translate`;
        circle -nr -1 0 0 -c (-$rshoulder[0]) $rshoulder[1] $rshoulder[2] -r 1 -n
RShoulderControl;xform-cp;

        //make the relbow control curve
        float $relbow[] = `getAttr relbow.translate`;
        circle -nr -1 0 0 -c (-$relbow[0]) $relbow[1] $relbow[2] -r 1 -n
RElbowControl;xform-cp;
```

```
        //make the rwrist control curve
        float $rwrist[] = `getAttr rwrist.translate`;
        circle -nr -1 0 0 -c (-$rwrist[0]) $rwrist[1] $rwrist[2] -r 1 -n
RWristControl;xform-cp;

        //make the left foot control curve
        float $lankle[] = `getAttr lankle.translate`;
        float $ltoe[] = `getAttr ltoe.translate`;
        float $lball[] = `getAttr lball.translate`;
        float $var;
        $var = ($ltoe[2] - $lankle[2]); //take the z pos of the ankle locator away from
toe locator to get length
        circle -nr 0 1 0 -c $ltoe[0] $ltoe[1] $lball[2] -r ($var) -n LFootControl;xform
-cp;
        scale -r 0.2 1 1 LFootControl.cv[3] LFootControl.cv[7];

        //make the right foot control curve
        float $rankle[] = `getAttr rankle.translate`;
        float $rtoe[] = `getAttr rtoe.translate`;
        float $rball[] = `getAttr rball.translate`;
        float $var;
        $var = ($rtoe[2] - $rankle[2]); //take the z pos of the ankle locator away from
toe locator to get length
        circle -nr 0 1 0 -c (-$rtoe[0]) $rtoe[1] $rball[2] -r ($var) -n
RFootControl;xform -cp;
        scale -r 0.2 1 1 RFootControl.cv[3] RFootControl.cv[7];

        //use two exisiting knee locators as pole vectors
        float $rknee[] = `getAttr rknee.translate`;
        float $lknee[] = `getAttr lknee.translate`;
        move -a (-$rknee[0]) $rknee[1] ($rknee[2]+5) rknee;
        move -a $lknee[0] $lknee[1] ($lknee[2]+5) lknee;

        button -e -en 0 controlcurves; //disable button
}

proc SetAngle() //displays the 7th window
{
        global string $folder;
        string $image = $folder+"picture06.bmp";
        if (`window -exists ControlCurves`)
        deleteUI ControlCurves;
        if (`window -exists OrientJoints`)
                deleteUI OrientJoints;
        if (`window -exists MakingTheLeg`)
                deleteUI MakingTheLeg;
        window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 7 of 17"
OrientJoints;
        paneLayout -cn "left3" -ps 1 40 94 -ps 2 60 100;
        columnLayout -columnAttach "both" 0 -columnWidth 280 -rowSpacing 5;
        text -label "Orientating Joints and Control Curves";
        text -label "To ensure normal behaviour we want the";
        text -label "joint's local rotation axis to be aligned";
        text -label "with the bone, pointing down the joint chain.";
        text -label "We also want the control curves lined up";
        text -label "perpendicular to the joints so the animator ";
        text -label "can always see the rotation without needing to";
        text -label "see the joints.";
        button -label "Orient Joints and Curves" -command "preferredAngle"
preferredangle;
        text -label "You should see that the control curves are now";
        text -label "in alignment with the joints.";
        setParent..;
        image -image $image;
        setParent..;
        rowLayout -numberOfColumns 3
                -columnWidth3 93 93 93
                -columnAlign  1 "center"
                -columnAlign  2 "center"
                -columnAlign  3 "center"
                -columnAttach 1 "both" 0
                -columnAttach 2 "both" 0
                -columnAttach 3 "both" 0;
                button -label "Back" -command ControlCurves;
```

```
		button -label "Undo" -command undo5;
		button -label "Next" -command Leg1;
	setParent..;
	showWindow OrientJoints;
}

proc preferredAngle() //orientates joints and curves
{
	joint -e -oj xyz -secondaryAxisOrient yup -ch -zso Rclavicle; //orient joints
from Rclavicle down
	joint -e -oj xyz -secondaryAxisOrient yup -ch -zso Lclavicle; //orient joints
from Lclavicle down

	//to get control curves to align with the joints local rotation axis create a
group and orient constrain that group to the joint, then delete the orient constraint
leaving the curve oriented
	group -n LShoulderControlGrp LShoulderControl;
	orientConstraint -offset 0 0 0 -weight 1 Lshoulder LShoulderControlGrp;
	delete LShoulderControlGrp_orientConstraint1;

	group -n LElbowControlGrp LElbowControl;
	orientConstraint -offset 0 0 0 -weight 1 Lelbow LElbowControlGrp;
	delete LElbowControlGrp_orientConstraint1;

	group -n LWristControlGrp LWristControl;
	orientConstraint -offset 0 0 0 -weight 1 Lwrist LWristControlGrp;
	delete LWristControlGrp_orientConstraint1;

	group -n RShoulderControlGrp RShoulderControl;
	orientConstraint -offset 0 0 0 -weight 1 Rshoulder RShoulderControlGrp;
	delete RShoulderControlGrp_orientConstraint1;

	group -n RElbowControlGrp RElbowControl;
	orientConstraint -offset 0 0 0 -weight 1 Relbow RElbowControlGrp;
	delete RElbowControlGrp_orientConstraint1;

	group -n RWristControlGrp RWristControl;
	orientConstraint -offset 0 0 0 -weight 1 Rwrist RWristControlGrp;
	delete RWristControlGrp_orientConstraint1;

	button -e -en 0 preferredangle; //disable button
}

proc Leg1() //displays the 8th window
{
	global string $folder;
	string $image = $folder+"picture07.bmp";
	if (`window -exists OrientJoints`)
		deleteUI OrientJoints;
	if (`window -exists MakingTheLeg`)
		deleteUI MakingTheLeg;
	if (`window -exists MakingTheLeg2`)
		deleteUI MakingTheLeg2;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 8 of 17"
MakingTheLeg;
	paneLayout -cn "left3" -ps 1 50 94 -ps 2 50 100;
	columnLayout -columnAttach "both" 0 -columnWidth 350 -rowSpacing 5;
	text -label "Setting up the Leg";
	text -label "We are going to setup the legs with inverse kinematics. This";
	text -label "is controlled by an IK handle that allows you to animate the";
	text -label "body while keeping the foot on the floor. The joints in the";
	text -label "IK chain will follow the joint lowest in the IK hierarchy, hence";
	text -label "the name inverse kinematics.";
	iconTextButton -width 100 -style "iconAndTextHorizontal"
	-image "kinHandle.xpm" -label "IK handle tool in Maya";
	text -label "There are two types of IK handle that we will use, rotate plane";
	text -label "and single chain. First we will attach a rotate plane IK handle";
	text -label "between the hip and the ankle. This type of IK handle allows";
	text -label "us to also create a pole vector constraint which controls the";
	text -label "direction the knee points in.";
	text -label "Next we will create two single chain IK handles between the";
	text -label "ankle and ball joints, and the ball and toe joints. These help";
	text -label "to hold the foot in place.";
	button -label "Create IK handle" -command "firstIK" firstik;
	text -label "To test this try selecting an IK handle and moving the";
```

```
        text -label "leg around.";
        setParent ..;
        image -image $image;
        rowLayout -numberOfColumns 3
                -columnWidth3 116 116 116
                -columnAlign  1 "center"
                -columnAlign  2 "center"
                -columnAlign  3 "center"
                -columnAttach 1 "both" 0
                -columnAttach 2 "both" 0
                -columnAttach 3 "both" 0;
                button -label "Back" -command SetAngle;
                button -label "Undo" -command undo6;
                button -label "Next" -command Leg2;
        setParent..;
        showWindow MakingTheLeg;
}

proc firstIK() //sets up IK handles on the legs
{
        //setup left IK handles
        ikHandle -n LballIK -sj Lankle -ee Lball -sol ikSCsolver;
        ikHandle -n LtoeIK -sj Lball -ee Ltoe -sol ikSCsolver;
        ikHandle -n LankleIK -sj Lhip -ee Lankle -sol ikRPsolver;

        //create pole vector
        poleVectorConstraint -weight 1 -n lkneePoleVector lknee LankleIK;

        //create pivot groups
        group -n peelHeelGrp LankleIK; xform -os -piv 0 0 0;
        float $lball[] = `getAttr lball.translate`;
        move $lball[0] $lball[1] $lball[2] peelHeelGrp.scalePivot
peelHeelGrp.rotatePivot ;

        //create foot roll groups
        group -n toeTapGrp LballIK LtoeIK; xform -os -piv 0 0 0;
        move $lball[0] $lball[1] $lball[2] toeTapGrp.scalePivot toeTapGrp.rotatePivot ;

        group -n toePivotGrp peelHeelGrp toeTapGrp; xform -os -piv 0 0 0;
        float $ltoe[] = `getAttr ltoe.translate`;
        move $ltoe[0] $ltoe[1] $ltoe[2] toePivotGrp.scalePivot toePivotGrp.rotatePivot
;

        group -n heelPivotGrp toePivotGrp; xform -os -piv 0 0 0;
        float $lankle[] = `getAttr lankle.translate`;
        move $lankle[0] $lankle[1] $lankle[2] heelPivotGrp.scalePivot
heelPivotGrp.rotatePivot ;

        //setup right IK handles
        ikHandle -n RballIK -sj Rankle -ee Rball -sol ikSCsolver;
        ikHandle -n RtoeIK -sj Rball -ee Rtoe -sol ikSCsolver;
        ikHandle -n RankleIK -sj Rhip -ee Rankle -sol ikRPsolver;

        //create pole vector
        poleVectorConstraint -weight 1 -n rkneePoleVector rknee RankleIK;

        //create pivot groups
        group -n peelHeelGrp1 RankleIK; xform -os -piv 0 0 0;
        float $rball[] = `getAttr rball.translate`;
        move (($rball[0])*-1) $rball[1] $rball[2] peelHeelGrp1.scalePivot
peelHeelGrp1.rotatePivot ;

        //setup foot roll groups
        group -n toeTapGrp1 RballIK RtoeIK; xform -os -piv 0 0 0;
        move (($rball[0])*-1) $rball[1] $rball[2] toeTapGrp1.scalePivot
toeTapGrp1.rotatePivot ;

        group -n toePivotGrp1 peelHeelGrp1 toeTapGrp1; xform -os -piv 0 0 0;
        float $rtoe[] = `getAttr rtoe.translate`;
        move (($rtoe[0])*-1) $rtoe[1] $rtoe[2] toePivotGrp1.scalePivot
toePivotGrp1.rotatePivot ;

        group -n heelPivotGrp1 toePivotGrp1; xform -os -piv 0 0 0;
        float $rankle[] = `getAttr rankle.translate`;
        move (($rankle[0])*-1) $rankle[1] $rankle[2] heelPivotGrp1.scalePivot
heelPivotGrp1.rotatePivot ;

        //setup pole vectors
        float $lknee[] = `getAttr lknee.translate`;
        move -a $lknee[0] $lknee[1] ($lknee[2]+5) lknee;
        rename "lknee" "LkneePoleVector";

        float $rknee[] = `getAttr rknee.translate`;
        move -a (-$rknee[0]) $rknee[1] ($rknee[2]+5) rknee;
        rename "rknee" "RkneePoleVector";

        button -e -en 0 firstik; //disable button
}

proc Leg2() //displays the 9th window
{
        global string $folder;
        string $image = $folder+"picture08.bmp";
        if (`window -exists MakingTheLeg`)
                deleteUI MakingTheLeg;
        if (`window -exists MakingTheLeg2`)
                deleteUI MakingTheLeg2;
        if (`window -exists MakingTheArm`)
                deleteUI MakingTheArm;
        window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 9 of 17"
MakingTheLeg2;
        paneLayout -cn "left3" -ps 1 60 94 -ps 2 40 100;
        columnLayout -columnAttach "both" 0 -columnWidth 420 -rowSpacing 5;
        text -label "Finishing the Leg";
        text -label "We are going to connect the movements of the IK handles to
attributes that";
        text -label "we will add to the foot curve. The foot curve will then easily
control the";
        text -label "movements of the foot setup. This will be useful for walk cycle
animations";
        text -label "where you need foot roll.";
        text -label "The set driven key tool requires input from a driver that will
drive a";
        text -label "driven attribute. In this case the foot curve is the driver and
the";
        text -label "movements of the IK handles will be driven.";
        text -label "Added attributes will inclued \"Toe Tap\" and \"Stand Tip\". These
attributes";
        text -label "are keyable. The set driven key tool can be found in the
\"Animate\" menu.";
        button -label "Set Driven Keys" -command "setupLegs" setuplegs;
        text -label "The IK handles must be parented under the foot curve for it to
have full";
        text -label "control over the leg. The hierarchy should look like the picture
opposite.";
        button -label "Parent" -command parenting parenting;
        text -label "Try selecting a foot curve and using the new attributes.";
        setParent ..;
        image -image $image;
                setParent ..;
                rowLayout -numberOfColumns 3
                -columnWidth3 140 140 140
                -columnAlign  1 "center"
                -columnAlign  2 "center"
                -columnAlign  3 "center"
                -columnAttach 1 "both" 0
                -columnAttach 2 "both" 0
                -columnAttach 3 "both" 0;
                button -label "Back" -command Leg1;
                button -label "Undo" -command undo7;
                button -label "Next" -command armWindow;
        setParent..;
        showWindow MakingTheLeg2;
}

proc setupLegs() //makes the driven keys
{
        //create left foot attributes
        addAttr -ln peelHeel -at double  -min 0 -max 10 |LFootControl;
```

```
setAttr -e -keyable true |LFootControl.peelHeel;
addAttr -ln standTip -at double  -min 0 -max 10 |LFootControl;
setAttr -e -keyable true |LFootControl.standTip;
addAttr -ln twistHeel -at double  -min -10 -max 10 |LFootControl;
setAttr -e -keyable true |LFootControl.twistHeel;
addAttr -ln twistToes -at double  -min -10 -max 10 |LFootControl;
setAttr -e -keyable true |LFootControl.twistToes;
addAttr -ln toeTap -at double  -min -10 -max 10 |LFootControl;
setAttr -e -keyable true |LFootControl.toeTap;

//create right foot attributes
addAttr -ln peelHeel -at double  -min 0 -max 10 |RFootControl;
setAttr -e -keyable true |RFootControl.peelHeel;
addAttr -ln standTip -at double  -min 0 -max 10 |RFootControl;
setAttr -e -keyable true |RFootControl.standTip;
addAttr -ln twistHeel -at double  -min -10 -max 10 |RFootControl;
setAttr -e -keyable true |RFootControl.twistHeel;
addAttr -ln twistToes -at double  -min -10 -max 10 |RFootControl;
setAttr -e -keyable true |RFootControl.twistToes;
addAttr -ln toeTap -at double  -min -10 -max 10 |RFootControl;
setAttr -e -keyable true |RFootControl.toeTap;

//set driven keys for left foot
setDrivenKeyframe -currentDriver LFootControl.peelHeel peelHeelGrp.rotateX;
setAttr "LFootControl.peelHeel" 10;
setAttr "peelHeelGrp.rotateX" 45;
setDrivenKeyframe -currentDriver LFootControl.peelHeel peelHeelGrp.rotateX;

setDrivenKeyframe -currentDriver LFootControl.toeTap toeTapGrp.rotateX;
setAttr "LFootControl.toeTap" -10;
setAttr "toeTapGrp.rotateX" 74;
setDrivenKeyframe -currentDriver LFootControl.toeTap toeTapGrp.rotateX;
setAttr "LFootControl.toeTap" 10;
setAttr "toeTapGrp.rotateX" -45;
setDrivenKeyframe -currentDriver LFootControl.toeTap toeTapGrp.rotateX;

setDrivenKeyframe -currentDriver LFootControl.standTip toePivotGrp.rotateX;
setAttr "LFootControl.standTip" 10;
setAttr "toePivotGrp.rotateX" 40;
setDrivenKeyframe -currentDriver LFootControl.standTip toePivotGrp.rotateX;

setDrivenKeyframe -currentDriver LFootControl.twistHeel heelPivotGrp.rotateX;
setAttr "LFootControl.twistHeel" -10;
setAttr "heelPivotGrp.rotateX" -50;
setDrivenKeyframe -currentDriver LFootControl.twistHeel heelPivotGrp.rotateX;
setAttr "LFootControl.twistHeel" 10;
setAttr "heelPivotGrp.rotateX" 50;
setDrivenKeyframe -currentDriver LFootControl.twistHeel heelPivotGrp.rotateX;

setDrivenKeyframe -currentDriver LFootControl.twistToes toePivotGrp.rotateY;
setAttr "LFootControl.twistToes" -10;
setAttr "toePivotGrp.rotateY" 50;
setDrivenKeyframe -currentDriver LFootControl.twistToes toePivotGrp.rotateY;
setAttr "LFootControl.twistToes" 10;
setAttr "toePivotGrp.rotateY" -45;
setDrivenKeyframe -currentDriver LFootControl.twistToes toePivotGrp.rotateY;

setAttr "LFootControl.peelHeel" 0;
setAttr "LFootControl.standTip" 0;
setAttr "LFootControl.twistHeel" 0;
setAttr "LFootControl.twistToes" 0;
setAttr "LFootControl.toeTap" 0;

//set driven keys for right foot
setDrivenKeyframe -currentDriver RFootControl.peelHeel peelHeelGrp1.rotateX;
setAttr "RFootControl.peelHeel" 10;
setAttr "peelHeelGrp1.rotateX" 45;
setDrivenKeyframe -currentDriver RFootControl.peelHeel peelHeelGrp1.rotateX;

setDrivenKeyframe -currentDriver RFootControl.toeTap toeTapGrp1.rotateX;
setAttr "RFootControl.toeTap" -10;
setAttr "toeTapGrp1.rotateX" 74;
setDrivenKeyframe -currentDriver RFootControl.toeTap toeTapGrp1.rotateX;
setAttr "RFootControl.toeTap" 10;
setAttr "toeTapGrp1.rotateX" -45;
```

```
        setDrivenKeyframe -currentDriver RFootControl.toeTap toeTapGrp1.rotateX;

        setDrivenKeyframe -currentDriver RFootControl.standTip toePivotGrp1.rotateX;
        setAttr "RFootControl.standTip" 10;
        setAttr "toePivotGrp1.rotateX" 40;
        setDrivenKeyframe -currentDriver RFootControl.standTip toePivotGrp1.rotateX;

        setDrivenKeyframe -currentDriver RFootControl.twistHeel heelPivotGrp1.rotateX;
        setAttr "RFootControl.twistHeel" -10;
        setAttr "heelPivotGrp1.rotateX" -50;
        setDrivenKeyframe -currentDriver RFootControl.twistHeel heelPivotGrp1.rotateX;
        setAttr "RFootControl.twistHeel" 10;
        setAttr "heelPivotGrp1.rotateX" 50;
        setDrivenKeyframe -currentDriver RFootControl.twistHeel heelPivotGrp1.rotateX;

        setDrivenKeyframe -currentDriver RFootControl.twistToes toePivotGrp1.rotateY;
        setAttr "RFootControl.twistToes" -10;
        setAttr "toePivotGrp1.rotateY" 50;
        setDrivenKeyframe -currentDriver RFootControl.twistToes toePivotGrp1.rotateY;
        setAttr "RFootControl.twistToes" 10;
        setAttr "toePivotGrp1.rotateY" -45;
        setDrivenKeyframe -currentDriver RFootControl.twistToes toePivotGrp1.rotateY;

        setAttr "RFootControl.peelHeel" 0;
        setAttr "RFootControl.standTip" 0;
        setAttr "RFootControl.twistHeel" 0;
        setAttr "RFootControl.twistToes" 0;
        setAttr "RFootControl.toeTap" 0;

        button -e -en 0 setuplegs;
}

proc parenting() //parent IK's under control curve
{
        parent heelPivotGrp LFootControl;
        parent heelPivotGrp1 RFootControl;
        button -e -en 0 parenting;
}

proc armWindow() //displays the 10th window
{
        global string $folder;
        string $image = $folder+"picture09.bmp";
        if (`window -exists MakingTheLeg2`)
                deleteUI MakingTheLeg2;
        if (`window -exists MakingTheArm`)
                deleteUI MakingTheArm;
        if (`window -exists ArmParenting`)
                deleteUI ArmParenting;
        window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 10 of 17"
MakingTheArm;
        paneLayout -cn "left3" -ps 1 50 96 -ps 2 50 100;
        columnLayout -columnAttach "both" 0 -columnWidth 350 -rowSpacing 2;
        text -label "Setting up the Arms";
        text -label "Forward Kinematics work by rotating a joint in a chain. It";
        text -label "allows the arms to move smoothly which is useful for walk";
        text -label "cycles.";
        text -label "To setup the arm we use orient constraints. An orient";
        text -label "constraint allows the rotations of one object, the constrained";
        text -label "object, to be controlled by another, the target object. The";
        text -label "three control curves of the arm will be target objects and the";
        text -label "shoulder, elbow and wrist joints will be the constrained";
        text -label "objects";
        iconTextButton -width 100 -style "iconAndTextHorizontal"
        -image "orientConstraint.xpm" -label "Orient Constraint tool in Maya";
        text -label "It is also important to freeze transformations on all curves";
        text -label "before adding constraints. This means resetting all translation";
        text -label "and rotation values to zero and is useful so you can always";
        text -label "get your curve back to its original position.";
        text -label "It is important to note that this is only one way of rigging the";
        text -label "arm. An IK handle could also be used for setup with a pole";
        text -label "vector to control the direction the elbow points in. This setup";
        text -label "would be very similar to the setup of the legs.";
        button -label "Setup Arms" -command "setupArms" setuparms;
        text -label "Try rotating an arm curve to see what happens.";
        setParent ..;
        image -image $image;
        setParent ..;
                rowLayout -numberOfColumns 3
                -columnWidth3 116 116 116
                -columnAlign  1 "center"
                -columnAlign  2 "center"
                -columnAlign  3 "center"
                -columnAttach 1 "both" 0
                -columnAttach 2 "both" 0
                -columnAttach 3 "both" 0;
                button -label "Back" -command Leg2;
                button -label "Undo" -command undo8;
                button -label "Next" -command armParentingWindow;
        setParent..;
        showWindow MakingTheArm;
}


proc setupArms() //orient constrain each arm curve to the joint
{
        orientConstraint -offset 0 0 0 -weight 1 LShoulderControl Lshoulder;
        orientConstraint -offset 0 0 0 -weight 1 LElbowControl Lelbow;
        orientConstraint -offset 0 0 0 -weight 1 LWristControl Lwrist;
        orientConstraint -offset 0 0 0 -weight 1 RShoulderControl Rshoulder;
        orientConstraint -offset 0 0 0 -weight 1 RElbowControl Relbow;
        orientConstraint -offset 0 0 0 -weight 1 RWristControl Rwrist;

        button -e -en 0 setuparms;
}

proc armParentingWindow() //displays the 11th window
{
        global string $folder;
        string $image = $folder+"picture10.bmp";
        if (`window -exists MakingTheArm`)
                deleteUI MakingTheArm;
        if (`window -exists ArmParenting`)
                deleteUI ArmParenting;
        if (`window -exists FingerSetup`)
                deleteUI FingerSetup;
        window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 11 of 17"
ArmParenting;
        paneLayout -cn "left3" -ps 1 50 94 -ps 2 50 100;
        columnLayout -columnAttach "both" 0 -columnWidth 350 -rowSpacing 5;
        text -label "Finishing the Arms";
        text -label "Rotating a shoulder curve now would result in rotating the arm";
        text -label "but leaving the elbow and wrist control curves behind. This is";
        text -label "because we need to build a control curve hierarchy. The wrist";
        text -label "curve should be parented to the elbow curve which in turn will";
        text -label "be parented to the shoulder curve.";
        text -label "The hierarchy should look like the picture opposite.";
        button -label "Parent Curves" -command armParenting armparenting;
        text -label "Try rotating an arm curve again to see how the parenting";
        text -label "has worked.";
        setParent ..;
        image -image $image;
        setParent ..;
                rowLayout -numberOfColumns 3
                -columnWidth3 116 116 116
                -columnAlign  1 "center"
                -columnAlign  2 "center"
                -columnAlign  3 "center"
                -columnAttach 1 "both" 0
                -columnAttach 2 "both" 0
                -columnAttach 3 "both" 0;
                button -label "Back" -command armWindow;
                button -label "Undo" -command undo9;
                button -label "Next" -command fingerSetup;
        setParent..;
        showWindow ArmParenting;
}

proc armParenting() //parents the arm control curves to the ones higher in the
hierarchy
```

```
{
	parent LWristControlGrp LElbowControl;
	parent LElbowControlGrp LShoulderControl;
	parent RWristControlGrp RElbowControl;
	parent RElbowControlGrp RShoulderControl;
	button -e -en 0 armparenting;
}

proc fingerSetup() //displays the 12th window
{
	global string $folder;
	string $image = $folder+"picture11.bmp";
	if (`window -exists ArmParenting`)
		deleteUI ArmParenting;
	if (`window -exists FingerSetup`)
		deleteUI FingerSetup;
	if (`window -exists BodySetup`)
		deleteUI BodySetup;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 12 of 17" 
FingerSetup;
	paneLayout -cn "left3" -ps 1 40 94 -ps 2 60 100;
	columnLayout -columnAttach "both" 0 -columnWidth 280 -rowSpacing 5;
	text -label "Setting up the Fingers and Jaw";
	text -label "By using driven keys again we shall make the";
	text -label "wrist curve manage the rotations of the finger";
	text -label "joints. The wrist curve is the driver and the";
	text -label "finger joint rotations will be driven. Extra";
	text -label "finger attributes will be added to the wrist";
	text -label "curve allowing the fingers to curl, make a fist,";
	text -label "spread etc.";
	text -label "We will also add a jaw open attribute to the neck";
	text -label "curve and add a driven key for this in the";
	text -label "same way.";
	button -label "Setup Fingers and Jaw" -command setupFingers setupfingers;
	text -label "Test the finger setup by selecting a wrist curve";
	text -label "and using the new attributes. Similarly to test";
	text -label "the jaw setup select the neck curve.";
	setParent ..;
	image -image $image;
	setParent ..;
		rowLayout -numberOfColumns 3
		-columnWidth3 93 93 93
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAlign  3 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0
		-columnAttach 3 "both" 0;
		button -label "Back" -command armParentingWindow;
		button -label "Undo" -command undo10;
		button -label "Next" -command bodySetup;
	setParent..;
	showWindow FingerSetup;
}

proc setupFingers() //creates driven keys for the fingers and jaw
{
	//add attributes to the LWristControl
	addAttr -ln thumbCurl -at double  -min -5 -max 10 LWristControl;
	setAttr -e -keyable true LWristControl.thumbCurl;
	addAttr -ln indexCurl -at double  -min -5 -max 10 LWristControl;
	setAttr -e -keyable true LWristControl.indexCurl;
	addAttr -ln middleCurl -at double  -min -5 -max 10 LWristControl;
	setAttr -e -keyable true LWristControl.middleCurl;
	addAttr -ln ringCurl -at double  -min -5 -max 10 LWristControl;
	setAttr -e -keyable true LWristControl.ringCurl;
	addAttr -ln babyCurl -at double  -min -5 -max 10 LWristControl;
	setAttr -e -keyable true LWristControl.babyCurl;
	addAttr -ln fingerSpread -at double  -min -10 -max 10 LWristControl;
	setAttr -e -keyable true LWristControl.fingerSpread;

	//set index curl dirven key
	setDrivenKeyframe -currentDriver LWristControl.indexCurl Lindex1.rotateZ 
Lindex2.rotateZ Lindex3.rotateZ;
	setAttr "LWristControl.indexCurl" 10;
```

```
        setAttr "Lindex1.rotateZ" -30; setAttr "Lindex2.rotateZ" -80; setAttr
"Lindex3.rotateZ" -100;
        setDrivenKeyframe -currentDriver LWristControl.indexCurl Lindex1.rotateZ
Lindex2.rotateZ Lindex3.rotateZ;
        setAttr "LWristControl.indexCurl" -5;
        setAttr "Lindex1.rotateZ" 6; setAttr "Lindex2.rotateZ" 5.5; setAttr
"Lindex3.rotateZ" 5;
        setDrivenKeyframe -currentDriver LWristControl.indexCurl Lindex1.rotateZ
Lindex2.rotateZ Lindex3.rotateZ;

        //set middle curl driven key
        setDrivenKeyframe -currentDriver LWristControl.middleCurl Lmiddle1.rotateZ
Lmiddle2.rotateZ       Lmiddle3.rotateZ;
        setAttr "LWristControl.middleCurl" 10;
        setAttr "Lmiddle1.rotateZ" -30; setAttr "Lmiddle2.rotateZ" -80; setAttr
"Lmiddle3.rotateZ" -100;
        setDrivenKeyframe -currentDriver LWristControl.middleCurl Lmiddle1.rotateZ
Lmiddle2.rotateZ       Lmiddle3.rotateZ;
        setAttr "LWristControl.middleCurl" -5;
        setAttr "Lmiddle1.rotateZ" 6; setAttr "Lmiddle2.rotateZ" 5.5; setAttr
"Lmiddle3.rotateZ" 5;
        setDrivenKeyframe -currentDriver LWristControl.middleCurl Lmiddle1.rotateZ
Lmiddle2.rotateZ       Lmiddle3.rotateZ;

        //set ring curl driven key
        setDrivenKeyframe -currentDriver LWristControl.ringCurl Lring1.rotateZ
Lring2.rotateZ Lring3.rotateZ;
        setAttr "LWristControl.ringCurl" 10;
        setAttr "Lring1.rotateZ" -30; setAttr "Lring2.rotateZ" -80; setAttr
"Lring3.rotateZ" -100;
        setDrivenKeyframe -currentDriver LWristControl.ringCurl Lring1.rotateZ
Lring2.rotateZ Lring3.rotateZ;
        setAttr "LWristControl.ringCurl" -5;
        setAttr "Lring1.rotateZ" 6; setAttr "Lring2.rotateZ" 5.5; setAttr
"Lring3.rotateZ" 5;
        setDrivenKeyframe -currentDriver LWristControl.ringCurl Lring1.rotateZ
Lring2.rotateZ Lring3.rotateZ;

        //set baby curl driven key
        setDrivenKeyframe -currentDriver LWristControl.babyCurl Lbaby1.rotateZ
Lbaby2.rotateZ Lbaby3.rotateZ;
        setAttr "LWristControl.babyCurl" 10;
        setAttr "Lbaby1.rotateZ" -30; setAttr "Lbaby2.rotateZ" -80; setAttr
"Lbaby3.rotateZ" -100;
        setDrivenKeyframe -currentDriver LWristControl.babyCurl Lbaby1.rotateZ
Lbaby2.rotateZ Lbaby3.rotateZ;
        setAttr "LWristControl.babyCurl" -5;
        setAttr "Lbaby1.rotateZ" 6; setAttr "Lbaby2.rotateZ" 5.5; setAttr
"Lbaby3.rotateZ" 5;
        setDrivenKeyframe -currentDriver LWristControl.babyCurl Lbaby1.rotateZ
Lbaby2.rotateZ Lbaby3.rotateZ;

        //set thumb curl driven key
        setDrivenKeyframe -currentDriver LWristControl.thumbCurl Lthumb1.rotateX
Lthumb1.rotateY        Lthumb2.rotateY Lthumb1.rotateZ;
        setAttr "LWristControl.thumbCurl" 10;
        setAttr "Lthumb1.rotateX" 30; setAttr "Lthumb1.rotateY" 35; setAttr
"Lthumb1.rotateZ" -40; setAttr        "Lthumb2.rotateY" 35;
        setDrivenKeyframe -currentDriver LWristControl.thumbCurl Lthumb1.rotateX
Lthumb1.rotateY        Lthumb2.rotateY Lthumb1.rotateZ;
        setAttr "LWristControl.thumbCurl" -5;
        setAttr "Lthumb1.rotateX" 0; setAttr "Lthumb1.rotateY" -20; setAttr
"Lthumb1.rotateZ" 0; setAttr  "Lthumb2.rotateY" -5;
        setDrivenKeyframe -currentDriver LWristControl.thumbCurl Lthumb1.rotateX
Lthumb1.rotateY        Lthumb2.rotateY Lthumb1.rotateZ;

        //set finger spread driven key
        setDrivenKeyframe -currentDriver LWristControl.fingerSpread Lthumb1.rotateY
Lindex1.rotateY        Lring1.rotateY Lbaby1.rotateY;
        setAttr "LWristControl.fingerSpread" 10;
        setAttr "Lthumb1.rotateY" 30; setAttr "Lindex1.rotateY" 2; setAttr
"Lring1.rotateY" -5; setAttr  "Lbaby1.rotateY" -6;
        setDrivenKeyframe -currentDriver LWristControl.fingerSpread Lthumb1.rotateY
Lindex1.rotateY        Lring1.rotateY Lbaby1.rotateY;
        setAttr "LWristControl.fingerSpread" -10;
        setAttr "Lthumb1.rotateY" -25; setAttr "Lindex1.rotateY" -20; setAttr
"Lring1.rotateY" 16; setAttr  "Lbaby1.rotateY" 35;
        setDrivenKeyframe -currentDriver LWristControl.fingerSpread Lthumb1.rotateY
Lindex1.rotateY        Lring1.rotateY Lbaby1.rotateY;

        //add attributes to the RWristControl curve
        addAttr -ln thumbCurl -at double  -min -5 -max 10 RWristControl;
        setAttr -e -keyable true RWristControl.thumbCurl;
        addAttr -ln indexCurl -at double  -min -5 -max 10 RWristControl;
        setAttr -e -keyable true RWristControl.indexCurl;
        addAttr -ln middleCurl -at double  -min -5 -max 10 RWristControl;
        setAttr -e -keyable true RWristControl.middleCurl;
        addAttr -ln ringCurl -at double  -min -5 -max 10 RWristControl;
        setAttr -e -keyable true RWristControl.ringCurl;
        addAttr -ln babyCurl -at double  -min -5 -max 10 RWristControl;
        setAttr -e -keyable true RWristControl.babyCurl;
        addAttr -ln fingerSpread -at double  -min -10 -max 10 RWristControl;
        setAttr -e -keyable true RWristControl.fingerSpread;

        //set index curl driven key
        setDrivenKeyframe -currentDriver RWristControl.indexCurl Rindex1.rotateZ
Rindex2.rotateZ Rindex3.rotateZ;
        setAttr "RWristControl.indexCurl" 10;
        setAttr "Rindex1.rotateZ" -30; setAttr "Rindex2.rotateZ" -80; setAttr
"Rindex3.rotateZ" -100;
        setDrivenKeyframe -currentDriver RWristControl.indexCurl Rindex1.rotateZ
Rindex2.rotateZ Rindex3.rotateZ;
        setAttr "RWristControl.indexCurl" -5;
        setAttr "Rindex1.rotateZ" 6; setAttr "Rindex2.rotateZ" 5.5; setAttr
"Rindex3.rotateZ" 5;
        setDrivenKeyframe -currentDriver RWristControl.indexCurl Rindex1.rotateZ
Rindex2.rotateZ Rindex3.rotateZ;

        //set middle curl driven key
        setDrivenKeyframe -currentDriver RWristControl.middleCurl Rmiddle1.rotateZ
Rmiddle2.rotateZ       Rmiddle3.rotateZ;
        setAttr "RWristControl.middleCurl" 10;
        setAttr "Rmiddle1.rotateZ" -30; setAttr "Rmiddle2.rotateZ" -80; setAttr
"Rmiddle3.rotateZ" -100;
        setDrivenKeyframe -currentDriver RWristControl.middleCurl Rmiddle1.rotateZ
Rmiddle2.rotateZ       Rmiddle3.rotateZ;
        setAttr "RWristControl.middleCurl" -5;
        setAttr "Rmiddle1.rotateZ" 6; setAttr "Rmiddle2.rotateZ" 5.5; setAttr
"Rmiddle3.rotateZ" 5;
        setDrivenKeyframe -currentDriver RWristControl.middleCurl Rmiddle1.rotateZ
Rmiddle2.rotateZ       Rmiddle3.rotateZ;

        //set ring curl driven key
        setDrivenKeyframe -currentDriver RWristControl.ringCurl Rring1.rotateZ
Rring2.rotateZ Rring3.rotateZ;
        setAttr "RWristControl.ringCurl" 10;
        setAttr "Rring1.rotateZ" -30; setAttr "Rring2.rotateZ" -80; setAttr
"Rring3.rotateZ" -100;
        setDrivenKeyframe -currentDriver RWristControl.ringCurl Rring1.rotateZ
Rring2.rotateZ Rring3.rotateZ;
        setAttr "RWristControl.ringCurl" -5;
        setAttr "Rring1.rotateZ" 6; setAttr "Rring2.rotateZ" 5.5; setAttr
"Rring3.rotateZ" 5;
        setDrivenKeyframe -currentDriver RWristControl.ringCurl Rring1.rotateZ
Rring2.rotateZ Rring3.rotateZ;

        //set baby curl driven key
        setDrivenKeyframe -currentDriver RWristControl.babyCurl Rbaby1.rotateZ
Rbaby2.rotateZ Rbaby3.rotateZ;
        setAttr "RWristControl.babyCurl" 10;
        setAttr "Rbaby1.rotateZ" -30; setAttr "Rbaby2.rotateZ" -80; setAttr
"Rbaby3.rotateZ" -100;
        setDrivenKeyframe -currentDriver RWristControl.babyCurl Rbaby1.rotateZ
Rbaby2.rotateZ Rbaby3.rotateZ;
        setAttr "RWristControl.babyCurl" -5;
        setAttr "Rbaby1.rotateZ" 6; setAttr "Rbaby2.rotateZ" 5.5; setAttr
"Rbaby3.rotateZ" 5;
        setDrivenKeyframe -currentDriver RWristControl.babyCurl Rbaby1.rotateZ
Rbaby2.rotateZ Rbaby3.rotateZ;
```

```
        //set thumb curl driven key
        setDrivenKeyframe -currentDriver RWristControl.thumbCurl Rthumb1.rotateX
Rthumb1.rotateY        Rthumb2.rotateY Rthumb1.rotateZ;
        setAttr "RWristControl.thumbCurl" 10;
        setAttr "Rthumb1.rotateX" 30; setAttr "Rthumb1.rotateY" 35; setAttr
"Rthumb1.rotateZ" -40; setAttr        "Rthumb2.rotateY" 35;
        setDrivenKeyframe -currentDriver RWristControl.thumbCurl Rthumb1.rotateX
Rthumb1.rotateY        Rthumb2.rotateY Rthumb1.rotateZ;
        setAttr "RWristControl.thumbCurl" -5;
        setAttr "Rthumb1.rotateX" 0; setAttr "Rthumb1.rotateY" -20; setAttr
"Rthumb1.rotateZ" 0; setAttr  "Rthumb2.rotateY" -5;
        setDrivenKeyframe -currentDriver RWristControl.thumbCurl Rthumb1.rotateX
Rthumb1.rotateY        Rthumb2.rotateY Rthumb1.rotateZ;

        //set finger spread driven key
        setDrivenKeyframe -currentDriver RWristControl.fingerSpread Rthumb1.rotateY
Rindex1.rotateY        Rring1.rotateY Rbaby1.rotateY;
        setAttr "RWristControl.fingerSpread" 10;
        setAttr "Rthumb1.rotateY" 30; setAttr "Rindex1.rotateY" 2; setAttr
"Rring1.rotateY" -5; setAttr  "Rbaby1.rotateY" -6;
        setDrivenKeyframe -currentDriver RWristControl.fingerSpread Rthumb1.rotateY
Rindex1.rotateY        Rring1.rotateY Rbaby1.rotateY;
        setAttr "RWristControl.fingerSpread" -10;
        setAttr "Rthumb1.rotateY" -25; setAttr "Rindex1.rotateY" -20; setAttr
"Rring1.rotateY" 16; setAttr  "Rbaby1.rotateY" 35;
        setDrivenKeyframe -currentDriver RWristControl.fingerSpread Rthumb1.rotateY
Rindex1.rotateY        Rring1.rotateY Rbaby1.rotateY;

        //set all attributes to 0
        setAttr "LWristControl.thumbCurl" 0;
        setAttr "LWristControl.indexCurl" 0;
        setAttr "LWristControl.middleCurl" 0;
        setAttr "LWristControl.ringCurl" 0;
        setAttr "LWristControl.babyCurl" 0;
        setAttr "LWristControl.fingerSpread" 0;
        setAttr "RWristControl.thumbCurl" 0;
        setAttr "RWristControl.indexCurl" 0;
        setAttr "RWristControl.middleCurl" 0;
        setAttr "RWristControl.ringCurl" 0;
        setAttr "RWristControl.babyCurl" 0;
        setAttr "RWristControl.fingerSpread" 0;

        //add jaw attribute to neck curve
        addAttr -ln JawOpen -at double  -min 0 -max 10 NeckControl;
        //set jaw open driven key
        setAttr -e -keyable true NeckControl.JawOpen;
        setDrivenKeyframe -currentDriver NeckControl.JawOpen Jaw.rotateX;
        setAttr "NeckControl.JawOpen" 10;
        setAttr "Jaw.rotateX" 45;
        setDrivenKeyframe -currentDriver NeckControl.JawOpen Jaw.rotateX;
        setAttr "NeckControl.JawOpen" 0;

        button -e -en 0 setupfingers; //disable button
}

proc bodySetup() //displays the 13th window
{
        global string $folder;
        string $image = $folder+"picture12.bmp";
        if (`window -exists FingerSetup`)
                deleteUI FingerSetup;
        if (`window -exists BodySetup`)
                deleteUI BodySetup;
        if (`window -exists BodyParenting`)
                deleteUI BodyParenting;
        window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 13 of 17"
BodySetup;
        paneLayout -cn "left3" -ps 1 50 94 -ps 2 50 100;
        columnLayout -columnAttach "both" 0 -columnWidth 350 -rowSpacing 5;
        text -label "Setting up the Body";
        text -label "For the body control curves (spine, hips & head) we shall";
        text -label "again use orientation constraints except for the COG curve.";
        text -label "The Centre of Gravity or \"COG\", is the curve that controls";
        text -label "the root of our joint hierarchy. We want this curve to be able";
        text -label "to rotate and translate so that the body can move forwards";
        text -label "and backwards while keeping the legs attached to the floor";
        text -label "with the IK handles.";
        text -label "A parent constraint will be used as this allows translations";
        text -label "and rotations to be controlled by the COG curve. Again";
        text -label "freezing transformations is important before adding constraints.";
        button -label "Setup Body" -command setupBody setupbody;
        text -label "Try selecting and spine curve to test the orientations. Select";
        text -label "the COG curve to see how a parent constraint differs.";
        setParent ..;
        image -image $image;
        setParent ..;
                rowLayout -numberOfColumns 3
                -columnWidth3 116 116 116
                -columnAlign  1 "center"
                -columnAlign  2 "center"
                -columnAlign  3 "center"
                -columnAttach 1 "both" 0
                -columnAttach 2 "both" 0
                -columnAttach 3 "both" 0;
                button -label "Back" -command fingerSetup;
                button -label "Undo" -command undo11;
                button -label "Next" -command finalParent;
        setParent..;
        showWindow BodySetup;
}

proc setupBody() //orient constrains the body curves to joints
{
        orientConstraint -offset 0 0 0 -weight 1 NeckControl Neck;
        orientConstraint -offset 0 0 0 -weight 1 SpineDControl SpineD;
        orientConstraint -offset 0 0 0 -weight 1 SpineCControl SpineC;
        orientConstraint -offset 0 0 0 -weight 1 SpineAControl SpineA;
        orientConstraint -offset 0 0 0 -weight 1 HipsControl Hips;
        setAttr "Hips_orientConstraint1.offsetX" 180; //add an offset so the hips don't
flip
        parentConstraint -mo -weight 1 COGControl COG; //give the COG a parent
constraint
        button -e -en 0 setupbody;
}

proc finalParent() //displays the 14th window
{
        global string $folder;
        string $image = $folder+"picture13.bmp";
        if (`window -exists BodySetup`)
                deleteUI BodySetup;
        if (`window -exists BodyParenting`)
                deleteUI BodyParenting;
        if (`window -exists GlobalSRT`)
                deleteUI GlobalSRT;
        window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 14 of 17"
BodyParenting;
        paneLayout -cn "left3" -ps 1 60 94 -ps 2 40 100;
        columnLayout -columnAttach "both" 0 -columnWidth 420 -rowSpacing 5;
        text -label "Parenting the Curves";
        text -label "The shoulder control curves should be parented below the spineD
curve.";
        text -label "This way when the back starts to bend forwards the arms will also
move.";
        text -label "Each spine curve should then be parented to the one above in the";
        text -label "hierarchy and finally the COG.";
        text -label "The hips will also be parented to the COG curve making it the
root";
        text -label "of our control rig.";
        text -label "Moving the COG curve moves all other curves apart from the feet
which";
        text -label "are held in place by the IK handles.";
        text -label "At this stage the hierarchy should look like the picture
opposite.";
        button -label "Parent Curves" -command bodyParenting bodyparenting;
        text -label "Try moving the COG curve to check the rest follow.";
        setParent ..;
        image -image $image;
        setParent ..;
                rowLayout -numberOfColumns 3
```

```
		-columnWidth3 140 140 140
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAlign  3 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0
		-columnAttach 3 "both" 0;
		button -label "Back" -command bodySetup;
		button -label "Undo" -command undo12;
		button -label "Next" -command globalSRTwin;
	setParent..;
	showWindow BodyParenting;
}

proc bodyParenting() //parent the hierarchy of body curves
{
	parent NeckControl SpineDControl;
	parent SpineDControl SpineCControl;
	parent SpineCControl SpineAControl;
	parent HipsControl SpineAControl COGControl;
	parent LFootControl RFootControl LkneePoleVector RkneePoleVector GlobalSRT;
	parent LShoulderControlGrp RShoulderControlGrp SpineDControl;
	button -e -en 0 bodyparenting;
}

proc globalSRTwin() //displays the 15th window
{
	global string $folder;
	string $image = $folder+"picture14.bmp";
	if (`window -exists BodyParenting`)
		deleteUI BodyParenting;
	if (`window -exists GlobalSRT`)
		deleteUI GlobalSRT;
	if (`window -exists CleanUp`)
		deleteUI CleanUp;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 15 of 17"
GlobalSRT;
	paneLayout -cn "left3" -ps 1 60 94 -ps 2 40 100;
	columnLayout -columnAttach "both" 0 -columnWidth 420 -rowSpacing 5;
	text -label "Finishing Off";
	text -label "The Global Scale, Rotate and Translate or \"GlobalSRT\" allows us
to";
	text -label "move the entire rig so the character can be positioned in a
scene.";
	text -label "The globalSRT needs to have all the control curves, IK handles
and";
	text -label "Pole Vectors parented beneath it.";
	text -label "The final hierarchy should look like the picture opposite.";
	button -label "Parent to globalSRT" -command globalSRT globalsrt;
	text -label "Try moving the GlobalSRT around your scene to check it moves the";
	text -label "entire rig.";
	setParent ..;
	image -image $image;
	setParent ..;
		rowLayout -numberOfColumns 3
		-columnWidth3 140 140 140
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAlign  3 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0
		-columnAttach 3 "both" 0;
		button -label "Back" -command finalParent;
		button -label "Undo" -command undo13;
		button -label "Next" -command cleanUpwin;
	setParent..;
	showWindow GlobalSRT;
}

proc globalSRT() //parent the COG curve to the GlobalSRT curve
{
	parent COGControl GlobalSRT;
	button -e -en 0 globalsrt;
}
```

```
proc cleanUpwin() //displays the 16th window
{
	global string $folder;
	string $image = $folder+"picture15.bmp";
	if (`window -exists GlobalSRT`)
		deleteUI GlobalSRT;
	if (`window -exists CleanUp`)
		deleteUI CleanUp;
	if (`window -exists links`)
		deleteUI links;
	window -tlc 395 575 -widthHeight 700 550 -title "Auto Rig Tutorial, 16 of 17"
CleanUp;
	paneLayout -cn "left3" -ps 1 40 94 -ps 2 60 100;
	columnLayout -columnAttach "both" 0 -columnWidth 280 -rowSpacing 5;
	text -label "Clean Up";
	text -label "The final stage is to clean up the scene by";
	text -label "removing attributes of the control curves that";
	text -label "we don't want to be keyed or modified. It is";
	text -label "unlikely that scale and visibility attributes";
	text -label "will need to be keyed and oriented curves will";
	text -label "not need their translation values changed.";
	text -label "These attributes will be locked and hidden from";
	text -label "view. This can be done by right clicking over the";
	text -label "attributes in the channel box and selecting";
	text -label "\"Lock and Hide Selected\", or by using the";
	text -label "Channel Control Editor found under the \"Window\"";
	text -label "menu. Here attributes that have been locked and";
	text -label "hidden can be unlocked and unhidden.";
	button -label "Clean Up" -command cleanUp cleanup;
	text -label "Selecting a curve should now reveal that it no";
	text -label "longer has all it's original attributes.";
	setParent ..;
	image -image $image;
	setParent ..;
		rowLayout -numberOfColumns 3
		-columnWidth3 93 93 93
		-columnAlign  1 "center"
		-columnAlign  2 "center"
		-columnAlign  3 "center"
		-columnAttach 1 "both" 0
		-columnAttach 2 "both" 0
		-columnAttach 3 "both" 0;
		button -label "Back" -command globalSRTwin;
		button -label "Undo" -command undo14;
		button -label "Next" -command links;
	setParent..;
	showWindow CleanUp;
}

proc cleanUp() //remove unwanted attributes
{
setAttr -lock true -keyable false "LFootControl.sx"; //clean up left foot control
	setAttr -lock true -keyable false "LFootControl.sy";
	setAttr -lock true -keyable false "LFootControl.sz";
	setAttr -lock true -keyable false "LFootControl.v";

setAttr -lock true -keyable false "RFootControl.sx"; //clean up right foot control
	setAttr -lock true -keyable false "RFootControl.sy";
	setAttr -lock true -keyable false "RFootControl.sz";
	setAttr -lock true -keyable false "RFootControl.v";

setAttr -lock true -keyable false "RShoulderControl.tx"; //clean up rshoulder control
	setAttr -lock true -keyable false "RShoulderControl.ty";
	setAttr -lock true -keyable false "RShoulderControl.tz";
	setAttr -lock true -keyable false "RShoulderControl.sx";
	setAttr -lock true -keyable false "RShoulderControl.sy";
	setAttr -lock true -keyable false "RShoulderControl.sz";
	setAttr -lock true -keyable false "RShoulderControl.v";

setAttr -lock true -keyable false "LShoulderControl.tx"; //clean up lshoulder control
	setAttr -lock true -keyable false "LShoulderControl.ty";
	setAttr -lock true -keyable false "LShoulderControl.tz";
	setAttr -lock true -keyable false "LShoulderControl.sx";
	setAttr -lock true -keyable false "LShoulderControl.sy";
	setAttr -lock true -keyable false "LShoulderControl.sz";
	setAttr -lock true -keyable false "LShoulderControl.v";

setAttr -lock true -keyable false "RElbowControl.tx"; //clean up right elbow control
	setAttr -lock true -keyable false "RElbowControl.ty";
	setAttr -lock true -keyable false "RElbowControl.tz";
	setAttr -lock true -keyable false "RElbowControl.rx";
	setAttr -lock true -keyable false "RElbowControl.rz";
	setAttr -lock true -keyable false "RElbowControl.sx";
	setAttr -lock true -keyable false "RElbowControl.sy";
	setAttr -lock true -keyable false "RElbowControl.sz";
	setAttr -lock true -keyable false "RElbowControl.v";

setAttr -lock true -keyable false "LElbowControl.tx"; //clean up left elbow control
	setAttr -lock true -keyable false "LElbowControl.ty";
	setAttr -lock true -keyable false "LElbowControl.tz";
	setAttr -lock true -keyable false "LElbowControl.rx";
	setAttr -lock true -keyable false "LElbowControl.rz";
	setAttr -lock true -keyable false "LElbowControl.sx";
	setAttr -lock true -keyable false "LElbowControl.sy";
	setAttr -lock true -keyable false "LElbowControl.sz";
	setAttr -lock true -keyable false "LElbowControl.v";

setAttr -lock true -keyable false "RWristControl.tx"; //clean up right wrist control
	setAttr -lock true -keyable false "RWristControl.ty";
	setAttr -lock true -keyable false "RWristControl.tz";
	setAttr -lock true -keyable false "RWristControl.sx";
	setAttr -lock true -keyable false "RWristControl.sy";
	setAttr -lock true -keyable false "RWristControl.sz";
	setAttr -lock true -keyable false "RWristControl.v";

setAttr -lock true -keyable false "LWristControl.tx"; //clean up left wrist control
	setAttr -lock true -keyable false "LWristControl.ty";
	setAttr -lock true -keyable false "LWristControl.tz";
	setAttr -lock true -keyable false "LWristControl.sx";
	setAttr -lock true -keyable false "LWristControl.sy";
	setAttr -lock true -keyable false "LWristControl.sz";
	setAttr -lock true -keyable false "LWristControl.v";

setAttr -lock true -keyable false "LkneePoleVector.rx"; //clean up lknee pole vector
	setAttr -lock true -keyable false "LkneePoleVector.ry";
	setAttr -lock true -keyable false "LkneePoleVector.rz";
	setAttr -lock true -keyable false "LkneePoleVector.sx";
	setAttr -lock true -keyable false "LkneePoleVector.sy";
	setAttr -lock true -keyable false "LkneePoleVector.sz";
	setAttr -lock true -keyable false "LkneePoleVector.v";

setAttr -lock true -keyable false "RkneePoleVector.rx"; //clean up rknee pole vector
	setAttr -lock true -keyable false "RkneePoleVector.ry";
	setAttr -lock true -keyable false "RkneePoleVector.rz";
	setAttr -lock true -keyable false "RkneePoleVector.sx";
	setAttr -lock true -keyable false "RkneePoleVector.sy";
	setAttr -lock true -keyable false "RkneePoleVector.sz";
	setAttr -lock true -keyable false "RkneePoleVector.v";

	setAttr -lock true -keyable false "NeckControl.tx"; //clean up neck curve
	setAttr -lock true -keyable false "NeckControl.ty";
	setAttr -lock true -keyable false "NeckControl.tz";
	setAttr -lock true -keyable false "NeckControl.sx";
	setAttr -lock true -keyable false "NeckControl.sy";
	setAttr -lock true -keyable false "NeckControl.sz";
	setAttr -lock true -keyable false "NeckControl.v";

	setAttr -lock true -keyable false "SpineDControl.tx"; //clean up spineD curve
	setAttr -lock true -keyable false "SpineDControl.ty";
	setAttr -lock true -keyable false "SpineDControl.tz";
	setAttr -lock true -keyable false "SpineDControl.sx";
	setAttr -lock true -keyable false "SpineDControl.sy";
	setAttr -lock true -keyable false "SpineDControl.sz";
	setAttr -lock true -keyable false "SpineDControl.v";

	setAttr -lock true -keyable false "SpineCControl.tx"; //clean up spineC curve
	setAttr -lock true -keyable false "SpineCControl.ty";
	setAttr -lock true -keyable false "SpineCControl.tz";
	setAttr -lock true -keyable false "SpineCControl.sx";
	setAttr -lock true -keyable false "SpineCControl.sy";
```

```
        setAttr -lock true -keyable false "SpineCControl.sz";
        setAttr -lock true -keyable false "SpineCControl.v";

        setAttr -lock true -keyable false "SpineAControl.tx"; //clean up spineA curve
        setAttr -lock true -keyable false "SpineAControl.ty";
        setAttr -lock true -keyable false "SpineAControl.tz";
        setAttr -lock true -keyable false "SpineAControl.sx";
        setAttr -lock true -keyable false "SpineAControl.sy";
        setAttr -lock true -keyable false "SpineAControl.sz";
        setAttr -lock true -keyable false "SpineAControl.v";

        setAttr -lock true -keyable false "HipsControl.tx"; //clean up hips curve
        setAttr -lock true -keyable false "HipsControl.ty";
        setAttr -lock true -keyable false "HipsControl.tz";
        setAttr -lock true -keyable false "HipsControl.sx";
        setAttr -lock true -keyable false "HipsControl.sy";
        setAttr -lock true -keyable false "HipsControl.sz";
        setAttr -lock true -keyable false "HipsControl.v";

        setAttr -lock true -keyable false "COGControl.sx"; //clean up COG curve
        setAttr -lock true -keyable false "COGControl.sy";
        setAttr -lock true -keyable false "COGControl.sz";
        setAttr -lock true -keyable false "COGControl.v";

        setAttr -lock true -keyable false "GlobalSRT.sx"; //clean up GlobalSRT
        setAttr -lock true -keyable false "GlobalSRT.sy";
        setAttr -lock true -keyable false "GlobalSRT.sz";
        setAttr -lock true -keyable false "GlobalSRT.v";

        delete hips root spineA spineB spineC spineD neck jaw head jawend legGroup
armGroup legGroup1     armGroup1 headEnd; //delete all locators

        button -e -en 0 cleanup;
}

proc links() //displays the 17th window
{
        if (`window -exists CleanUp`)
                deleteUI CleanUp;
        if (`window -exists Links`)
                deleteUI Links;
        window -tlc 395 575 -widthHeight 350 400 -title "Auto Rig Tutorial, 17 of 17"
Links;
        paneLayout;
        columnLayout -columnAttach "both" 0 -columnWidth 350 -rowSpacing 10;
        text -label "For further help with rigging, try the links below.";
        text -label "http://www.rigging101.com/"; //links to other web resources
        text -label "http://members.iinet.net.au/~zapo/pages/tutorials.htm";
        text -label "http://www.spookypeanut.co.uk/";
        text -label "Or try...";
        button -label "Maya Help!" -command Help; //opens maya help
        button -label "Close" -command end;
        button -label "Back" -command cleanUpwin;
        showWindow Links;
}

proc end() //deletes last window
{
        deleteUI Links;
}

begin
```