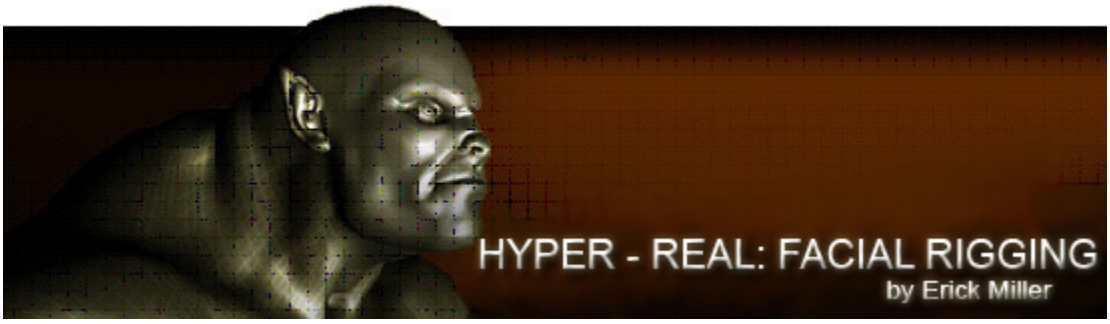


Hyper-Real Facial Rigging, Siggraph 2004 – Maya Masters Class Seminar
Instructor: Erick Miller, Lead Technical Director, Digital Domain

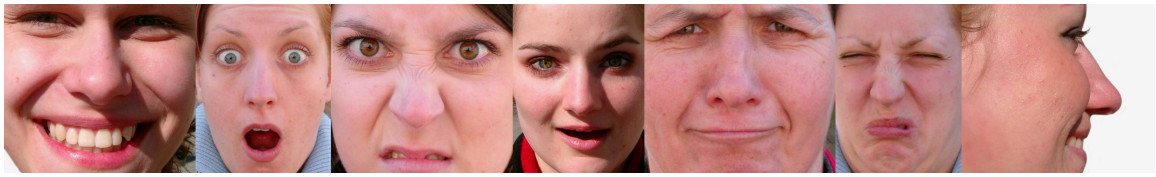
Introduction

This Masters Class will demonstrate effective, flexible and efficient techniques for creating hyper-real hero facial deformation rigs. With an emphasis on mixing rigging techniques to achieve fully realistic control, this class will explore the rigging process from the concept stage through to pipeline integration. You will learn strategies for creating realistic creature and character facial rigs with an emphasis on anatomy and realism. This class will teach effective techniques and workflows for creating rigs that can then be easily animated and rendered.

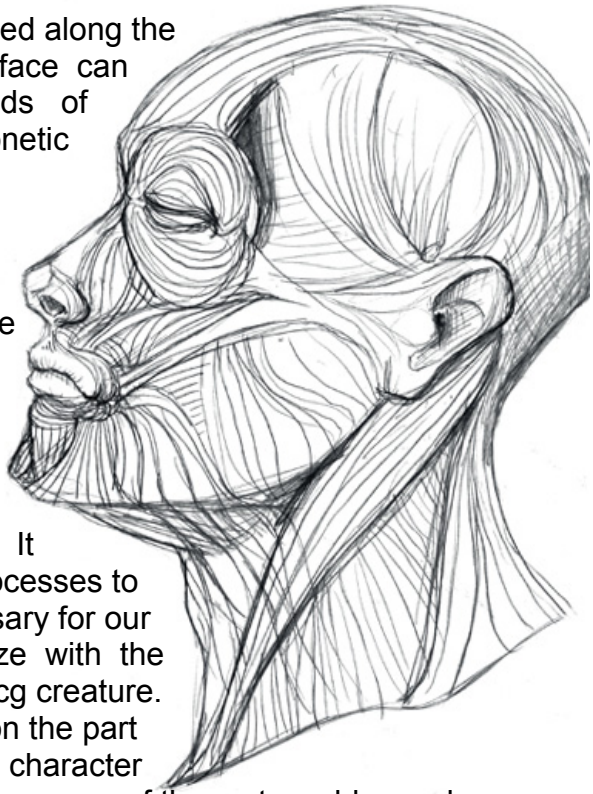
This class is the final part of a three part *Hyper-Real* series. Be sure to check out Jeff Unay's class, *Hyper-Real: Modeling and* Paul Thuriot's presentation, *Hyper-Real: Body Setup.*

I. The Amazing Face

The *face*: The single most expressive, complex and recognizable feature of almost any living being. The amazing wide range of emotions, ideas and suggestions that even a subtle change in facial expression can communicate are almost as broad and powerful as the very spoken language that it articulates. Millions of years of evolution have trained human beings to quickly and easily recognize and read the expressions of the face. The meaning of the spoken word can be completely redefined through the subtle change of facial expressions during speech.



Facial contractions directly communicate emotion to the viewer. How do you feel when you look at each image?
http://www.3d.sk



With over 40 free floating muscles intertwined along the bones of the jaw and the skull, the human face can produce literally thousands upon thousands of uniquely identifiable expressions and phonetic shapes.

The most important thing to remember about the face is that it naturally conveys emotion to its audience, therefore quite easily slipping the viewer into a state of suspended disbelief. A properly rigged [hyper-realistic] facial setup goes much beyond a few modeled blend shapes, and attempts to enter into a new realm of possible motion & rendering capabilities. It must allow the animation and rendering processes to achieve the heightened believability necessary for our brains to temporarily believe and empathize with the facial movements of an artificially created cg creature. Of course very careful skill must be taken on the part of the animation and lighting/shading of the character as well, but without the rig tying it all together none of the rest would even be possible.

Define "Hyper-Real"

Ok, so what does the term hyper-real mean, anyway?

Well, first off, we though it would be a catch sounding gimmicky name for a series of *Maya Master Class* presentations. All kidding aside, though, it is an important aspect to understand why certain anatomically based methods are needed or used in this series of classes, instead of using perhaps non-anatomy based approaches which could lead to a less dynamically realistic and potentially more canned or "cartoony" result in the final product.

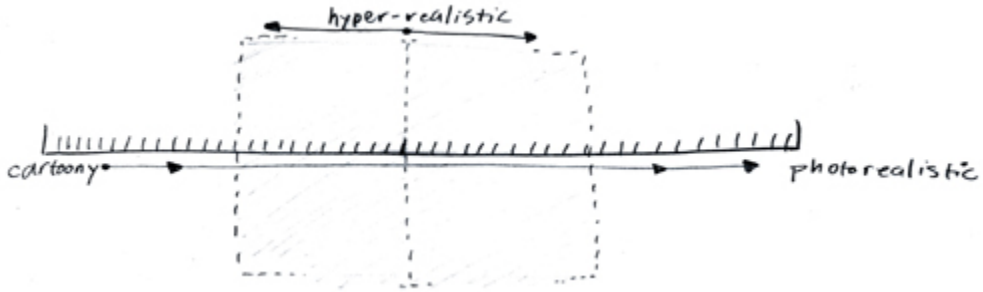
So, if there is someone from a dictionary looking to add some words, here's a shot at an official definition.

hy·per·re·al (hī'pər-rē'əl)
adj.

1. Above or beyond the quality of convincing existence, to the point of fantastic disbelief.
2. Something imaginary or incredible portrayed in a vividly realistic and believable manner.

Traditional clay sculpture of the "hyper-real" character that we are focusing on in this class.
Image and sculpture courtesy of Jeff Unay.

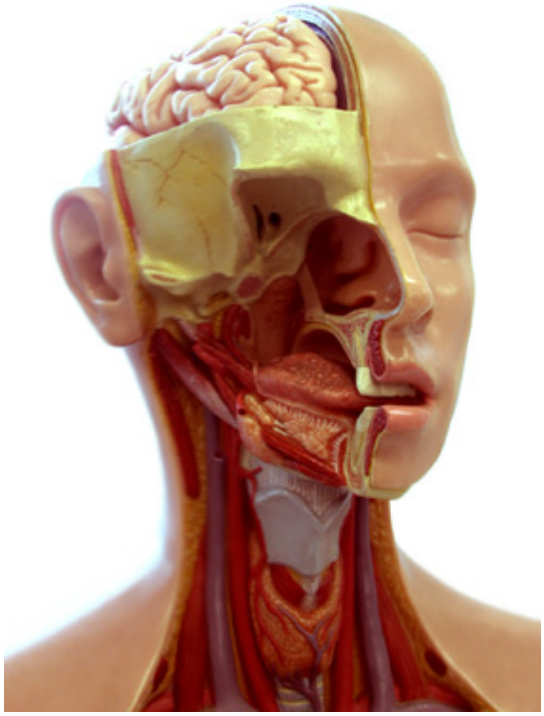
Hyper-real is believable, but, of course, is not photo-realism, although this is diving into somewhat muddied water. Some colleagues have joked that hyper-realistic is really just a nice way of saying "sub-par photo-realistic". This could be the case, depending on the quality and design goals of the final model, animation and rendering style, but this is most likely usually not the case.



Hyper-Real: Sub-par photo-real? Hopefully not!

On this topic, many have also argued that in order for photo-realism to be completely achieved, the rendered subject should most likely have a realistic quality of naturalism and a high potential for actual existence in real life as the viewer knows to already exist. Of course this is digressive and objective, so for now let's simply think of hyperrealism as "*nearing or over-achieving believable photo-realism, but when dealing with a creature or subject that is obviously not a known species in existence"*.

Specific Challenges of a Realistic Facial Setup



To attempt and build some sort of observed "analytical" model of something as complex as the human face, and engineer it in such a way that it's capable of a broad range of mechanical behavior based on a physical simulation of real anatomical structure, visually believable enough to pass as the real thing would be nearly impossible (if not completely insane) to do under the scrutiny and limitations of current reality (but could be a really great scientific research project).

In computer graphics today, however, the advances in techniques, technologies and graphics processing power have landed us close enough to say that we're able to achieve the believability part right now. Combine knowledge and technology with the age-old ancient magical art of *Chee-Ting**; which is so often a commodity in computer graphics, and the challenges can be overcome today in a nifty little bag of tricks.

The face has many unique challenges that must be carefully broken-down into components, each addressed with separate solutions. The most notable challenges we will address in this class are:

- *Anatomy*: Muscle based controls to achieve any expression or phoneme
- *Open Mouth*: Open mouth w/ volume & full range of realistic motion
- *Lips*: Fleshy/Squashy Rolling Lips and Sticky Lips
- *Eyes*: Lids, Blinks, Fleshy Eye Skin, Dilation, Iris Refraction.
- *Skin*: Pores, Fine Wrinkles, Subsurface Scatter Tricks

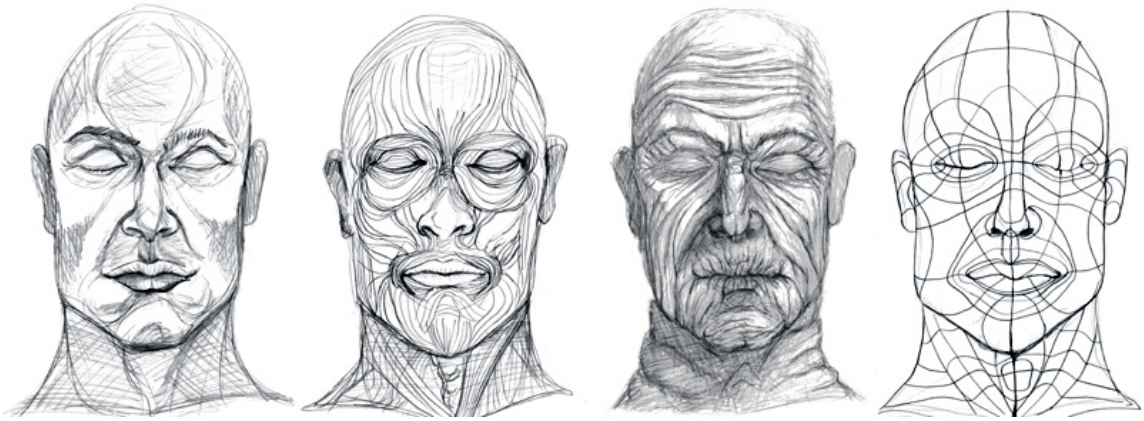
** Magical Art of Chee-Ting? …you know! I believe it's otherwise pronounced "the magical art of cheating"* ☺

II. OVERVIEW OF TECHNIQUES AND THEORY

The face has many unique challenges, and because of that fact, there is no golden solution to solve them all at once. Instead, each portion of the face must be analyzed and broken down into a catered solution motivated by each unique problem. First we will start with model integrity, and move on to the idea and reasons behind combining these multiple rigging techniques into one single rig.

Model Deformation Integrity and Good Edge Flow

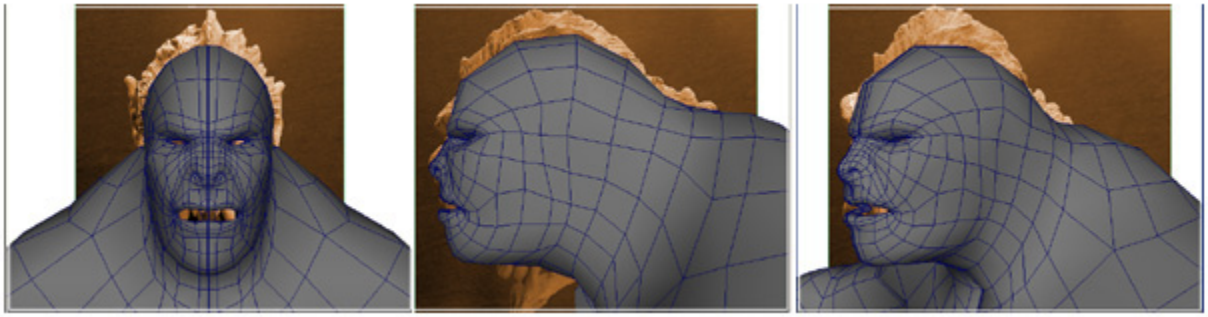
How the surface of the facial model is built is probably the single most important aspect to place under high scrutiny prior to and during the very first phase of the rigging process. This is true with character setup in general, but is particularly an issue of extreme importance with the head and face.



Examine the illustrations left to right. First is a simple face illustrating basic shadow planes, Next is the muscle structure beneath, Third exemplifies the course wrinkle lines of the face, and last shows the primary basic edge flow that your model can have based on the intersection between muscle lines and wrinkle lines. It is really important to notice that the fiber direction of muscles in the face run perpendicular and traverse to the direction of the wrinkle lines. Combining these two directions into your final polygon model is the goal.

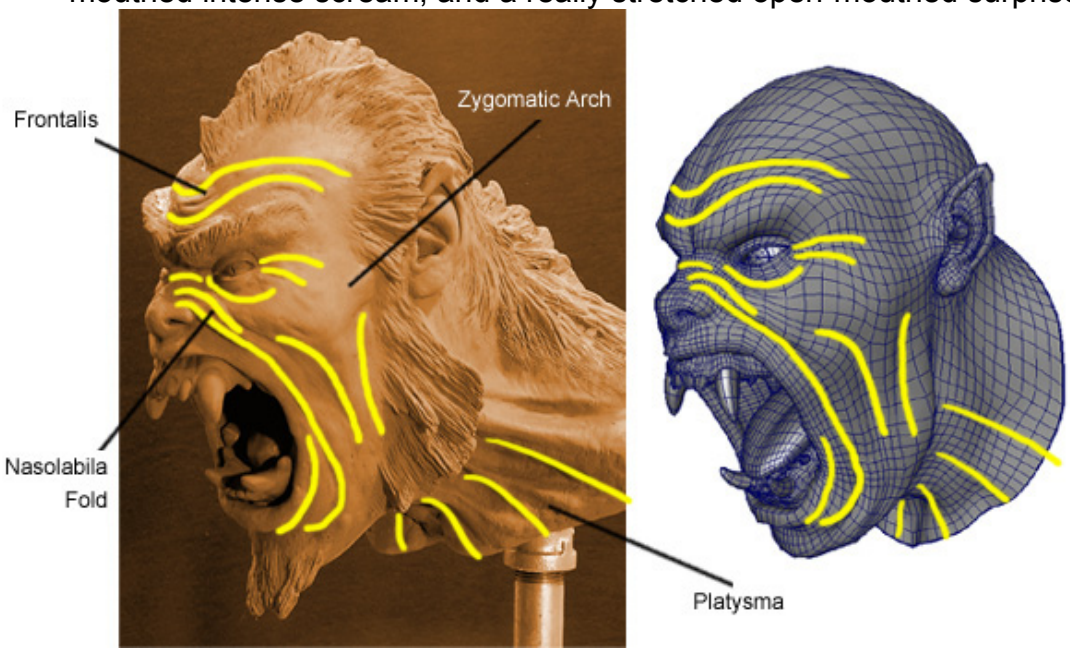
- First off, for modern facial rigging, a single mesh quad polygon control cage, optionally rendered as subdivision surfaces is the only way to go. I could probably leave this un-said, but for posterity I'll mention it anyway. For all sorts of painful reasons, patch models for facial rigging will only limit and prolong your rigging process, and are justifiably a thing of the past when complex facial animation setups are concerned.

- Secondly, the model must have a really good wrinkle-line based edge flow in it's creation, with radial circular edge loops radiating from the eyes, mouth, nose holes, and ears, as seen in the following diagrams.



Edge layout and flow is determined in the earliest stages of modeling. Notice how the edge circles radiate from the various regions of interest. Image and model courtesy of Jeff Unay.

- Third, the model must have plenty of resolution to support the most complex range of base expressions. If you are not sure about this, a good rule of thumb is to quickly model the face into a broad smile, a really wide-mouthed intense scream, and a really stretched open-mouthed surprise



Here is our poly mesh model, compared with the original sculptural maquette. Notice the edge layout and even mesh resolution on the model corresponds with wrinkle lines caused by the major muscle regions. Image and model courtesy of Jeff Unay.

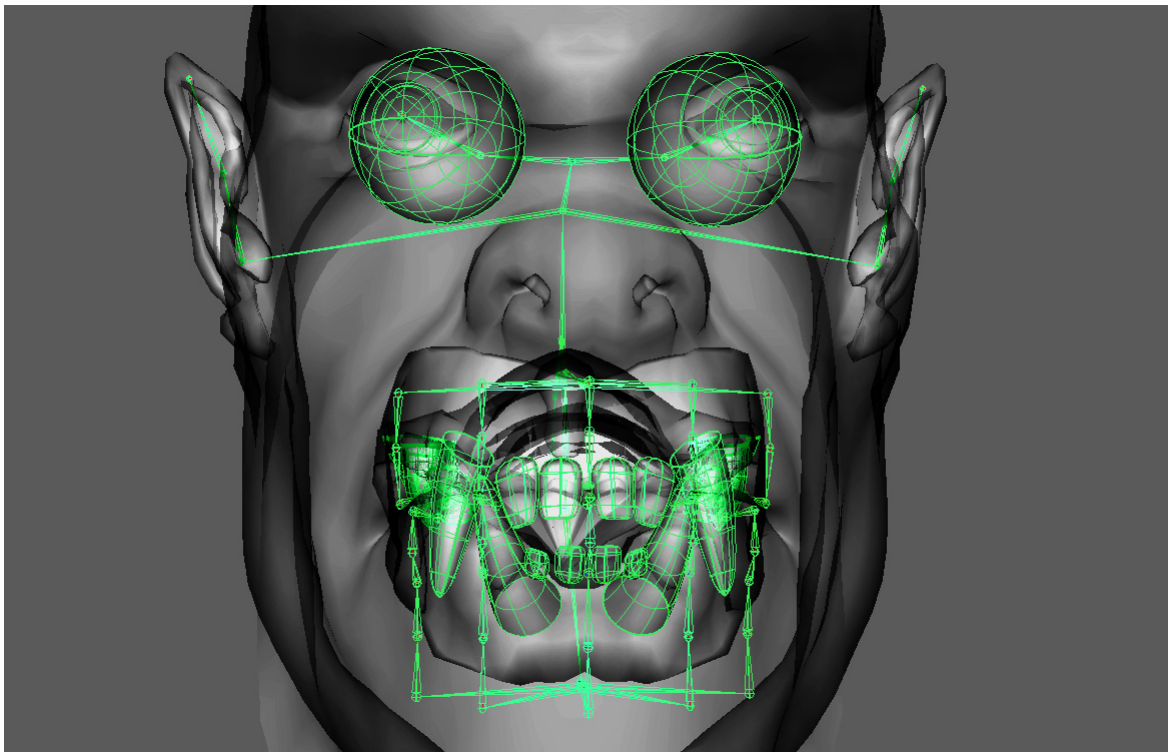
face. A lot of time the need arises to add a few extra rows of edges to the model at primary creasing points, primarily in laugh line, the upper nose, forehead and brow wrinkle lines, the crow wrinkle area around the eyes, as

well as possibly the area under the lower lip and above the chin where frown lines occur. These are all hot spots to police for sufficient resolution, and as you begin rigging your character, demanding extra resolution in those areas you need the most can sometimes make or break the ability to achieve the correct necessary range of facial poses.

Skeletal Placement Considerations & Using Joints

Deciding when to use hierarchies of joints to deform the face, and when to use modeled shapes is pretty simple. Joints should be used for portions of the face that need to either rotate, or have some sort of driven transformation controls that are more realistically and easier achieved using joints.

With this motivation, joints can be used to solve several of the cases listed above as our facial rigging challenges. The most obvious one of all would be the jaw, which is a skeletal structure in real life, which rotates and translates in an obviously arching manner.



Here is a combination of a couple different types of joint hierarchies that were used to create arc based rolling and rotational controls to deform our character's facial skin.

The other portions of the face that will be rigged separately using joint hierarchies are:

- Jaw

- Rolling / Fleshy lips
- Fleshy Eyes

Pros and Cons of Blend Shapes

The reason for not using modeled shapes to open the jaw is because blend shapes are linearly interpolated, this means that the vertex follows a straight line as you blend in a model, so there is never an arc occurring when dealing with blend shapes. This is problematic when an arc is required or a much more realistic of a motion path is needed for that expression. The face is naturally arced along the skull, so unfortunately this is one issue that can have a way of coming up more often that one would like.
On the other hand, blend shapes are really great deformers, and are probably one of the most useful and widely used nodes for facial rigging, so they should be very much respected, understood, and by all means, used! The problem of arcs can be alleviated by doing multiple in-between shapes, which cause the vertex to appear to follow more of an arc (even though it is really still just going from one straight line to another). It is also possible for an animator to creatively mix blend-shapes together in a very intuitive and powerful way, which far out-weighs any problem of arcs. The blend-shape is so powerful because of the fact that it is stable and linear, and what you see is what you get – all qualities that are absolutely invaluable when it comes to character deformations.

Understanding How Blend Shapes Work

A blend shape is really a very simple, but powerful linear deformer. The way that a blend shape works, is first it subtracts the original model from your newly modeled shape, leaving only the offsets for each point. It then simply multiplies the offset vector that it is storing by the blend shape's weight slider value for that shape. Next, it simply adds the result back with the original vertex point, and that's it.

The models shown above were each partially added into the center model using the blend shape algorithm (with a jaw joint to open the mouth) in order to achieve a mixture of available expressions combined into one single model.

So, the blend shape algorithm really is that simple. Lets take a look at what it would look like if we were to write it as a simple one line expression; where *Pa, Pb* and *Pc* are all new x,y,z shape modeled points, *aw, bw,* and *cw* are the single value blend shape control weight attributes, *Porig* is the original input point prior to receiving it's deformation, and *P* is the actual final deformed output point on the rendered character model.

For each vertex on the model, set *P* as follows:

P = (((Pa – Porig) * aw) + ((Pb – Porig) * bw) + ((Pc –Porig) * cw)) + Porig ;

Muscle Based Targets & Separate Symmetric Face Regions

The anatomic nature of the face will be carefully considered when deciding what the target shapes for using blend shapes will be. Probably the single most important factor in creating the modeled shape based portion of the facial rig is using a distinct combination of two very important methodologies.

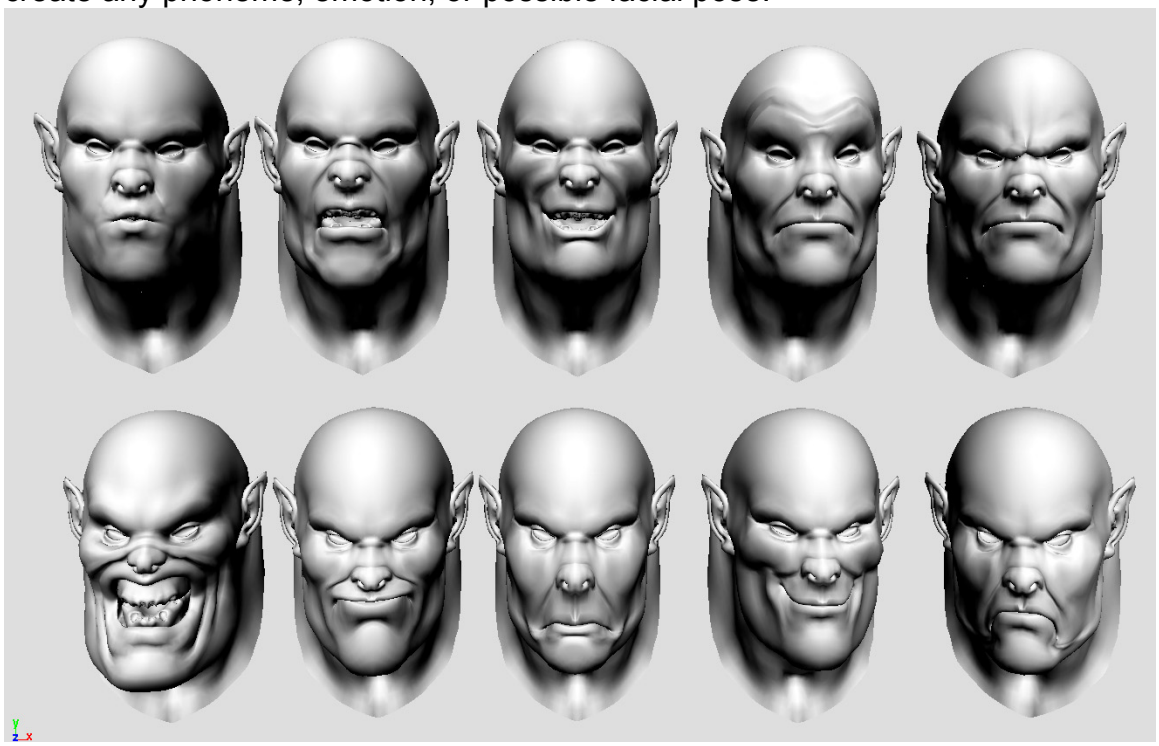
1. ***Muscle Based Facial Models*** (based on the contraction, pulling and expansion of the face resulting from the activation of major muscle groups) are modeled as fully articulated models first.

2. ***Symmetry Upper/Lower Left/Middle/Right region*** based splitting up of these muscle based models into separate blend shape models that can be blended in as separate sliders for ultimate asymmetrical muscle based "shaping" control happens next.

All of these models are used as blend shapes, giving course levels of control, as well as finer levels of control (not to mention tons and tons of sliders!).

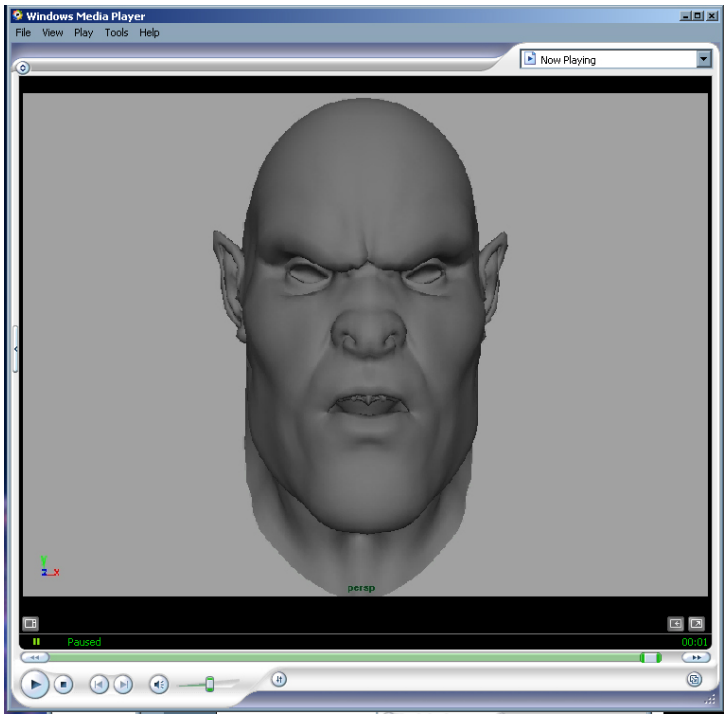
What does this mean exactly?

Well, for one, it means that there will be no spoken phoneme or emotional mood based modeling for the most part. All the models will be solely motivated by the fact that they are uniquely caused by a separate group of facial muscles that push or pull mostly independently to create the modeled shape. The models will all be created without any jaw-opening what so ever, and if the jaw must be opened to get the shape looking good, it should be subtracted back out from the shape before the model is finished and applied as one of the blend shapes. This way all the multiple shapes can be more easily combined together, in order to create any phoneme, emotion, or possible facial pose.



Here are just a few facial muscle based targets that were modeled with the muscle based group methodology in mind (please see list above for complete set that should be modeled). Notice that many of the shapes can be pushed to extreme poses for a wider range of control without the model tearing. It is also

important to note that although models attempt to isolate muscle groups, the fact that skin pulls with a falloff into other regions of the face should absolutely be considered. Examples of this are push/pull of the skin in the cheeks and chin (from lip muscles) and push/pull of the skin on the top of the head and ears (from forehead muscles).



See the movie file "mixing_some_blendshapes.avi" for a general idea of how easily the shapes shown above can be mixed together (just an example of mixing separate shapes – animation is not meant to look like anything in particular).

So, this is all the main idea and theory anyhow, and of course rules are always meant to be broken; but, let's start with a fairly extensive list of muscle based facial targets that can be modeled as a starting point for a really broad range of facial expression capabilities when properly symmetrically separated and then re-combined during animation:

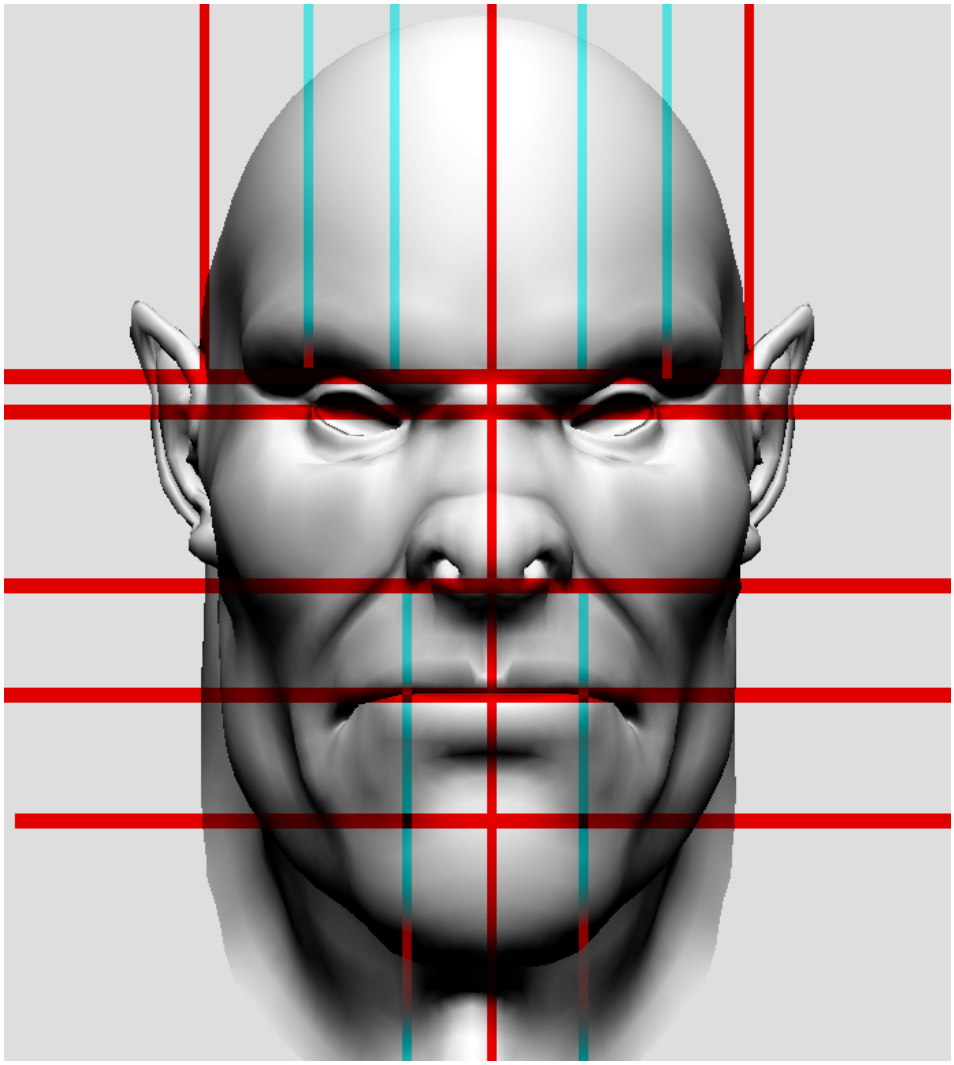
- Brow Down/Up (Furrow & Raise)
- Forehead Brow Sideways (Inward Pull / Outward Stretch)
- Eye Blinks (covered more in depth later)
- Wide-Open Eyes
- Tightly Closed/Squinted Eyes
- Nose Inward, Outward and Downward Flares
- Nose Sneer/Snarl
- Squinted Upper Cheeks/Lower Eyes (Including Crows Feet)
- Puffed Out and Sucked In Cheeks
- Lips Up/Down (upward and downward facial shrugs)
- Lip Corners Up/Down (pull of smile, push of frown)

- Lip Corners Sideways (Inward Pull / Outward Stretch)
- Vertically Flattened/Tightened Lips and Puffed out lips
- Open Mouthed Smile
- Oh-Ooo Mouth Shapes (several in-between targets required)
- Squeezed CloseTightened lips (closed lip purse/pucker)
- Upward Lip Sneer/Snarl & Downward Lip Sneer
- Jaw Clentch / Forehead Clentch / Neck Clentch

Next, these models are very carefully separated into individual facial areas. The areas are clearly based on the asymmetric changes that the face is highly capable of. We can analytically split the face up into upper, middle and lower as well as Left, Middle and Right regions. Each one of the muscle based shapes should then be split up into these separate regions carefully using the edit-membership tool on a blend shaped model, and then tweaking each shape so that it has a smooth fall-off around the edges of each region. Being able to paint weights on the blend shape deformer would be quite ideal (and easy to implement) for this process, indeed.

**Be sure to check out the "Condensed Reference Video Appendix" contained on the Hyper Real Facial Setup DVD which has high quality video footage of real human facial movement based on the above outlined muscle based shapes!*

The following diagram clearly shows these separate regions:



Face separated into symmetry based regions, where each shape should be roughly split up based on the region of the grid in which it falls. The red lined grid is the primary priority grid, and the light blue lines are secondary priority regions that give much finer grain separation, but often are overkill. For example, the separation of any single brow shape could range in size anywhere between 1 to 9 separate brow shapes: One for the whole, one for the left and right, and one for each individual split section – note that this fine level of distinction is usually not needed for very many shapes, but is often requested by the animator when a certain pose becomes un-achievable without it. It is a good idea to wait for specific requests before attempting to generate these fine level models, sticking with the red grid is usually quite a lot to start testing with anyhow.

Corrective Shapes for joint deformations

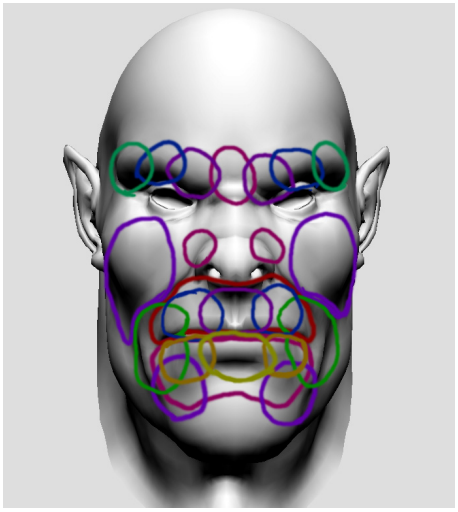
Modeling corrective shapes when joints deform the face that created into blend shapes which get set-driven key activated when the joints rotate is the perfect combination of techniques to achieve the exact modeled end pose, but using a rotational interpolation in-between. This topic is covered in both Jeff and Paul's

classes on Modeling and Body Rigging, but is definitely worth mentioning again here, due to the powerful results that can occur when using this technique. Probably the most obvious and expected corrective blend shape is one that will be driven by the jaw's rotation, which will allow the mouth's natural "o" shape to appear and stretch in a visually appropriate way as though it is skin that is maintaining some sort of volume around the structure of the jaw and skull as the mouth opens. Other corrective shapes will be used for example to get the rolling lip shaped up a little quicker.

Cluster Based Face-Shaping Controls

So, the final setup involves one more set of controls. These are the cluster based shape controls. These will be covered in more detail later, but it is important to note what they are and why they can and probably should be used. Clusters are very similar to joints, except they have some really nice transformation capabilities that give them a bit more power when it comes to the particular challenge of regionally based facial controls. First off, they can be used and rigged up in such a way that the controllers for the clusters stay attached to the mesh while it deforms. This is really useful because of all the deformation techniques that are being layered together would otherwise cause cluster controls to be floating in the wrong place for regions of the face that had already been animated with blend shapes.

The following diagram shows circled regions of the face for smoothly weighted cluster based facial controls to be applied onto.



Finally, we must understand that these controls are solely meant to give the animator a "sculptural" control over the face, but in a very smoothly weighted way. These clusters are not meant to crease wrinkle lines very much, and should drop off to zero around their edges very smoothly. This way the muscle based shapes will be used for the bulk of the realistic looking posing and the shaping can be used to add some rotational offsets, or to slide the pose over a little bit. These clusters are in no way meant to be a replacement for the blend shape controls, and definitely should be thought of as a very valuable secondary set of facial controls.

ALL the "theoretical" rules are now meant to be *broken*!

Ok, so once you have this entire army of blend shapes, all the joint and cluster controls you could ever wish for are set up, and everything is running well, you

must put the rig through a really extensive animation test, where the goal is to hand it off to multiple animators for testing; to have it speak all the possible phonemes, as well go through a full range of emotional shifts combined with dialogue. The combination of individually moving muscle based motion should begin to make more sense as you see the control it gives animators to realistically time and shape independent facial movements as well as adding subtle asymmetrical details quite quickly and naturally…. The process is a bit more based on reality, and the result is a much higher quality level of realism built into the face's skin…

B U T . . .

- Sometimes shapes will not "behave" exactly as expected. This can be the case particularly with shapes that move different muscles, but move vertices in the general same direction, but at differing rates. Un-expected and un-appealing double transformations that cause ripping can occur. We will cover some topics on how to deal with some of these using driven multipliers as a rigging topic later on, but first these problems should be addressed in the modeled shapes as much as possible. This is what I meant by breaking some of the rules. Splitting certain shapes out that don't really coincide directly with only a muscle group, but help create a very specific form in the face is sometimes necessary. Think of what you would need to do if you had to make a really slow motion scene of a face getting punched so hard that it broke the skin and fractured part of the skull beneath, or perhaps a slow-mo close-up of a suicidal gunshot from within the mouth. Although these are quite gruesome examples, this sort of thing gets done quite often in visual effects, because you just simply cannot do it to the real actor. These types of cases defiantly require custom shape models or additional controls to get the job done.

- Other reasons for additional models as blend shapes that are not purely muscle based are what I will call "signature" poses. This is the case with many famous celebrities – they have certain key facial expressions or looks that they are really well known for, which have helped make them famous and pretty much defines their persona (think of the Elvis or James Brown perhaps). Capturing this look exactly is often incredibly important, and if necessary, justifies an entirely new set of models just to achieve the "signature" pose. It is usually possible to get really close with the rig, and then just duplicate off the model, and do the subtle tweaks until it looks exactly correct. This pose can then be used to blend in at varying degrees to make the face look more "in-character".

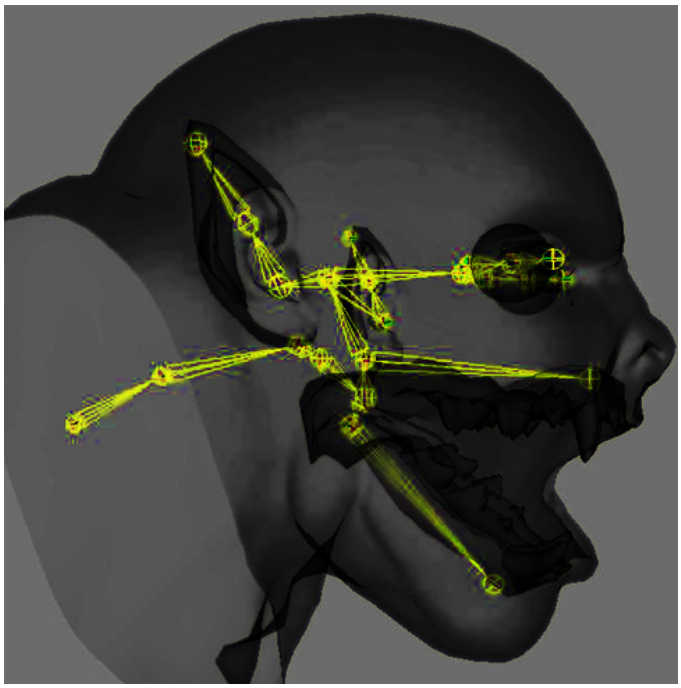
- Asymmetrical additive drooping shapes that include well-formed wrinkle lines can also be quite useful depending on the need of the character. What this means is that you make a few blend shapes that simply droop parts of the face very subtly, as though the skin is being slightly pulled up

and pushed around by a force such as gravity or by your finger (look in the mirror, and put your fingers on your cheek applying force downward towards the ground and upwards towards your ears). These sorts of shapes are pretty easy to whip out for the modeler, they can still be generically used in many circumstances, and are defiantly organically and reality based. For example they can be very useful for an animator in particular situations to make the face skin look more dynamic during a quick whipping the head around, or a forceful motion, and general subtle drooping can help add a look of tiredness or depravity to any facial expression.

III. DIRECT RIGGING TECHNIQUES

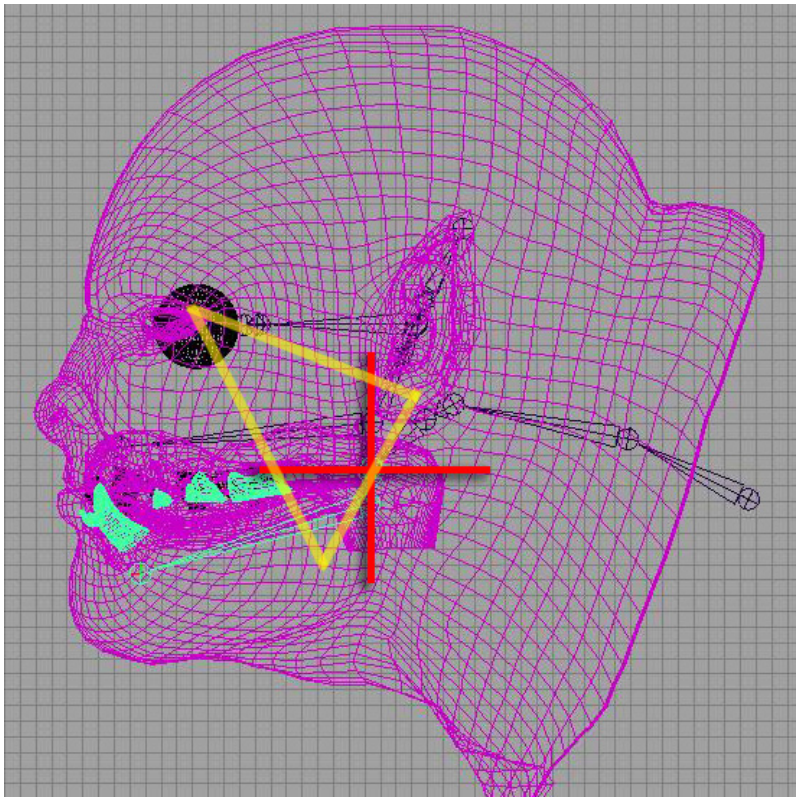
Now that the specific muscle-structural details, challenges and difficulties, as well as explanations for the motivations and techniques has been laid out, the rest of this document will be more detailed insight into the specific rigging techniques used to create these facial rigging and deformation effects.

Skeletal Hierarchy, Skinning and Weighting



Joints are deforming the face with a simple smooth bind between 2 joints as the jaw opens.

- The pivot of the jaw almost always exists in a clearly defined area just below the ear, along a line that runs perpendicular to another line that you could draw from the ear-lobe to the corner of the nearest eye; it is never really behind the ear, and never as far forward as the cheek. Sometimes it is worth experimenting with the pivot location in order to full “discover” what the best location for the jaw is, based on the complexity of your model or it’s derivation from being human.

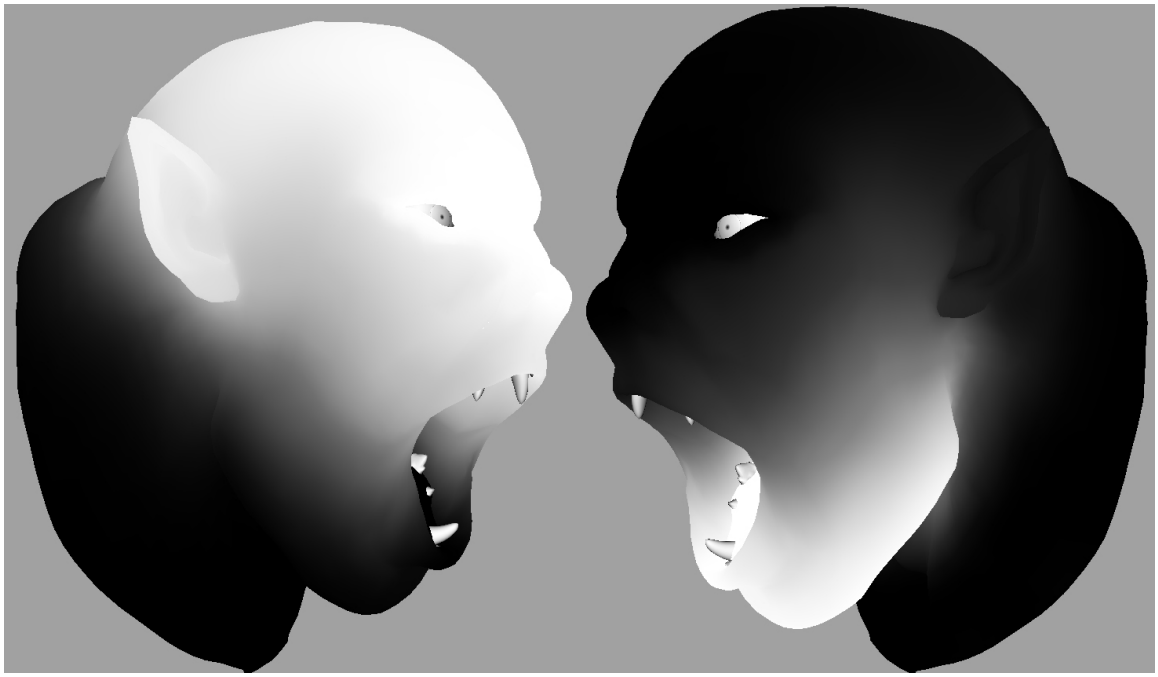


If your character has human enough features, this triangular technique shown above in yellow can help you locate the ideal pivot for your jaw joint. The Red Cross marks the spot. Pythagoras sure would be proud ☺

- The main skeletal hierarchy for weighting the jaw can really just consist of two joints, both having pivots in the exact same location, at the jaw, and both being children of the same parent, the main head joint. What this allows us to do is to rotate the head upwards from the same single pivot, not changing the jaw location, as well as conventionally rotating the jaw as well. See the movie file ***jaw_range_of_motion.avi*** in the *movies* folder for an example of the wide range that the final jaw setup can animate.

- The extra joints found in the face are not always necessarily needed for deformation, many joints are used simply as extra child or parent transforms to add additional controls to later, for example the extra joints above and below the jaw are used to parent the teeth to, and the eye joints to parent the eyes to (although they will help us add more deformations, and control some extra deformers later, when we get to the fleshy eyes section).

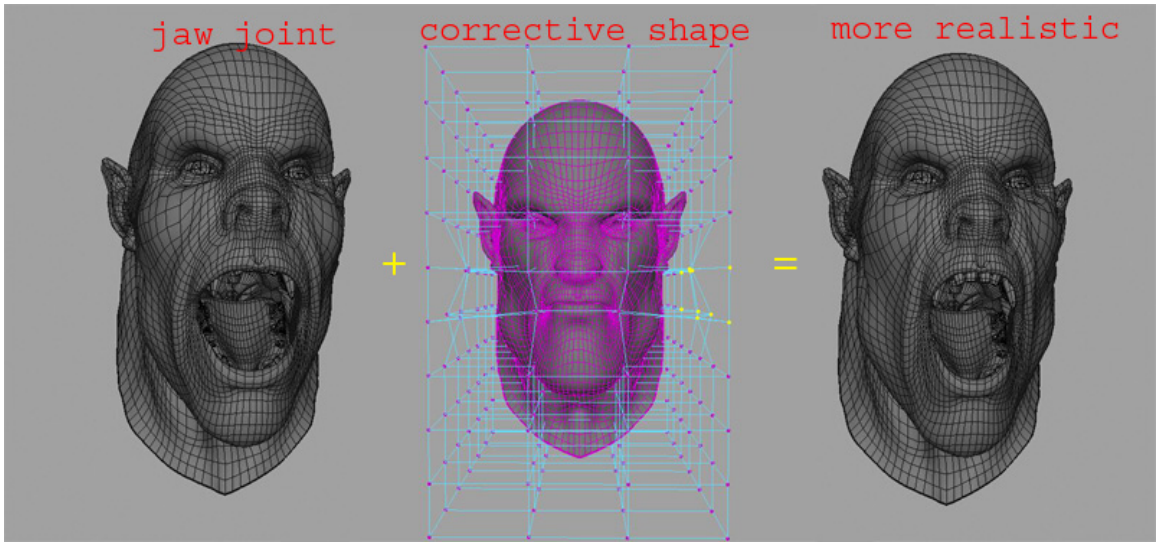
- Carefully and patiently painting really well distributed weights for the joint that opens the mouth is probably the single most important first step to creating a really believable facial deformation rig. Definitely spend as much time on this section (it really doesn't take that long) and concentrate on a 50/50 fall-off radiating from the corners of the lips, wrapping lightly upwards towards the temple and laugh-lines, and downward towards the chin. Add just really subtle amounts of weighting dabbed in with a low opacity setting weighted to the jaw into areas like the lower nose and the upper lip. In turn, weight portions of the lower chin just ever-so subtly more just below the lower lip weighted onto the jaw which controls the upper portion of the head. These little details are a really important part of having a mouth that appears realistically fleshy when the jaw opens, attempting to give the upper and lower regions above the lips a subtle feeling of being pulled by muscle and stretched over a structure that the lips are able to subtly wrap around just a little bit more (i.e. the jaw and the teeth).



Notice the smooth and subtle fall-off between weights, this causes a nicely averaged stretching of the entire facial skin giving a very fleshy appearance when the jaw is animated open. This will be the only mechanism for opening the

characters jaw, which will get layered in with the muscle and symmetry based blend-shape controls, as well as the regionally based cluster controls.

- After the weighting is really looking even (similar to the image above) then it is time to add the corrective blend shape that will allow the mouth to appear and stretch to maintain volume. This corrective is really easy to create by simply duplicating the base model (un-deformed) and making a front of chain blend shape. Next, set the blend shape to 1, and open the jaw of the model. Put a well good resolution lattice around the blend shape model, and begin to shape it so that the lip volume looks good. Now, do a couple set driven keys to control the weighting of the blend shape to be off when the jaw is closed, and on when the jaw is open.



Jaw Joint + Corrective Shape = Cool

Categorizing, Storing, and *MEL Scripting* the Blend Shape Setup

Ok, so you have all these shape models. There are probably so many that you have already completely lost track of which ones are which! Well, that is a challenge no matter what when dealing with the shear number of models you need for a hero facial rig.

So, how do you store them, and actually keep track of them to rig into your character so they are clearly saved and easy to update?
Well, I would strongly suggest creating a two pronged approach to solve this problem:

1. A nice clean directory structure of models, and organizing it based on major areas of the face (areas that an animator could relate to, like, “Cheeks”, “Mouth”, “Forehead”, “Brows”, etc.). Each one of the models saved out should be as light in data as possible, and should only contain

the single blend shape target model, no history, no uvs, nothing but the polygon modeled head.

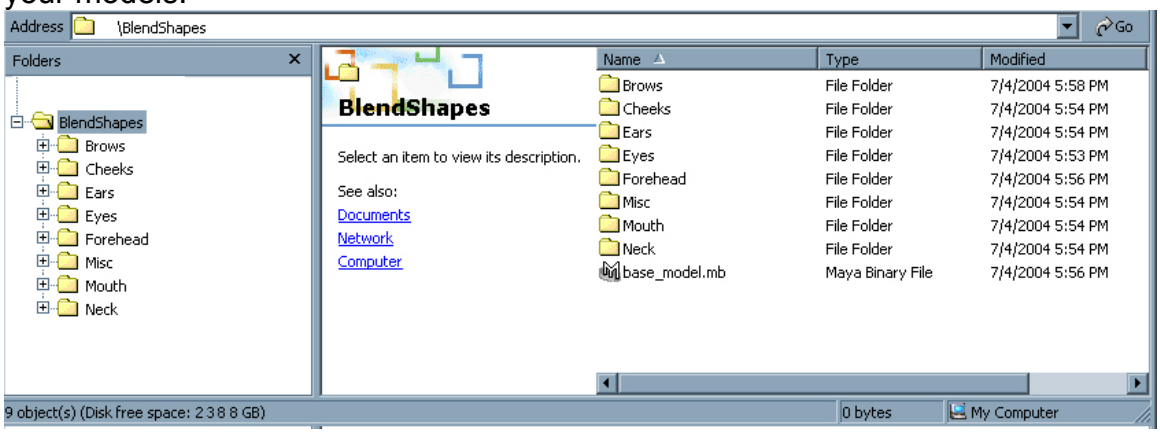
2. A really clear naming convention for the models, and most especially the ones that are split off into facial regions. The best way to come up with a naming convention is to make it simple to read, and contain all the necessary info in order to identify the model ("Full", "Left", "Right", "Center", etc.). I will leave the rest of these details as an exercise to the reader, because although having decent and logical naming conventions is extremely important, often times the details of the naming conventions themselves can spark very unnecessary tangential debates – which are not really related to the current problem hand*.

**I believe Paul (Thuriot) mentions naming conventions more specifically and more in context in his in-depth Hyper-Real Body Setup Course, which would be a very appropriate thing to read in regards to this issue, and I would recommend very much.*

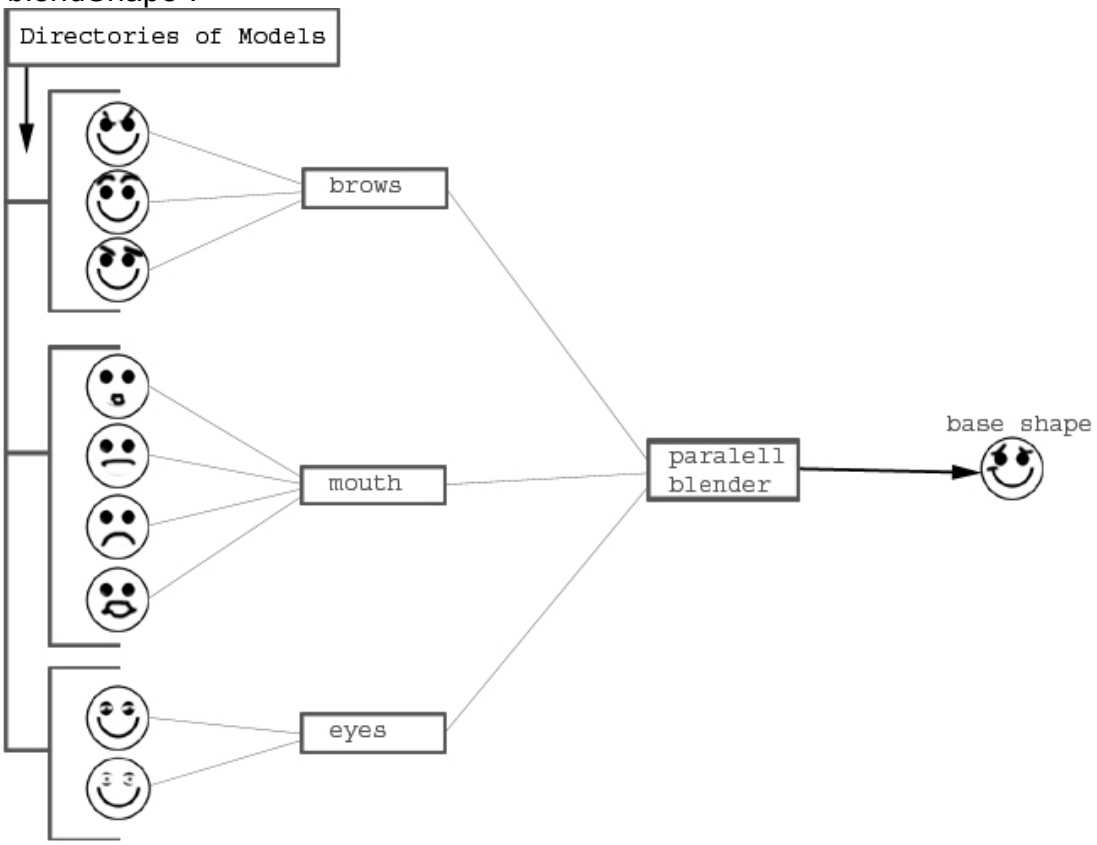
The huge reason for wanting to have a clearly defined directory structure and naming convention for your models is so that you (or your friendly neighborhood TD) can *script the rigging process** for adding the blend shape models to the facial rig. This means that when a bunch of models are updated, all that is necessary is to re-build the facial rig by running a MEL script, and getting all the updated models with no hassles or questions, and in a very clean way. Writing a script like this is relatively simple, and is even more simple when a clean directory structure and naming convention that the MEL script can "understand and read-from" is used. Next, each facial region that is agreed upon with the animators can have a subdirectory associated with it.

The process of scripted rigging, and many more advanced character rigging solutions similar to this nature are covered much more in depth in the Alias DVD, **"Maya Techniques™ | Custom Character Toolkit" which Paul Thuriot and I also taught as a Master Class at Siggraph. This DVD is available through http://www.alias.com/ -> click on the link for the on-line store.*

Here is very rough example of a directory structure that could be used for storing your models:



The base model for all the blend shapes can be saved at the root directory, and then using MEL commands, each subdirectory can be listed; each model can be imported, and added as a blend shape to the base node. Each subdirectory should represent it's own group, which makes it easy to use a separate blend shape node for each sub-directory, and use a front of chain "parallel blender" blend shape deformer (which is another use of a blend shape in Maya) to add each one of the blend shapes representing each sub-directory of models into the base model. What a parallel blender does is simply add all the deformations that are being applied together multiplied by a weight, and is literally a nodeType of "blendShape".



The MEL code that reads in the models and blend shapes them into the base mesh is pretty simple code that any entry to mid level MEL coder could get working pretty quickly.
Here is some pseudo code to show some simple sample logic that would do this:

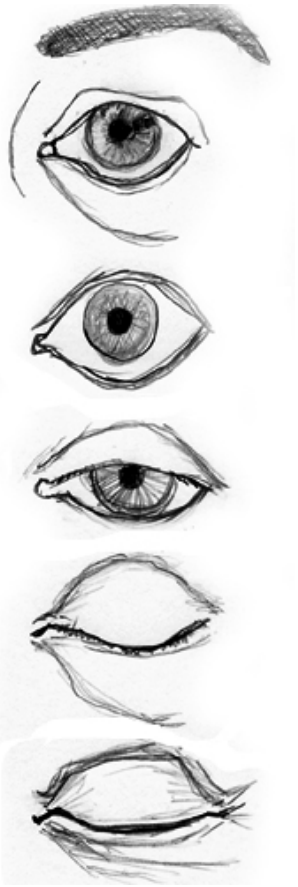
```
Import base_model.mb
For each ShapeFolder in BlendShapes folder{

        getFileList of ShapeFolder
        For each Model in ShapeFolder file list
        {
                import Model
                if( blendshape named ShapeFolder doesn't exist) {
                        make parallel blendshape ShapeFolder on base_model.
                }
                add Model to blend shape named ShapeFolder.
        }
}
```

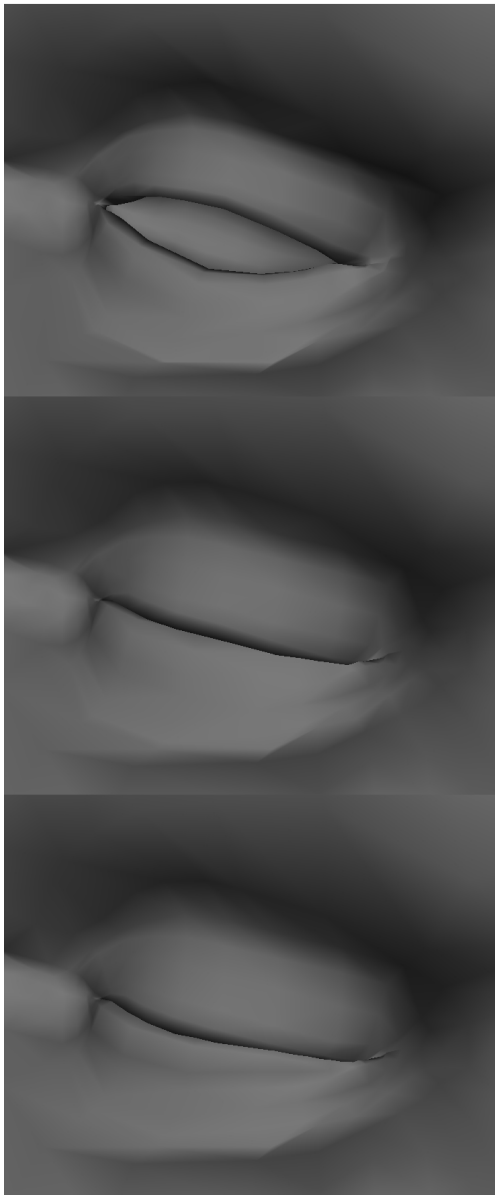
Tons of actual MEL and C++ source code, and many more advanced coded rigging solutions similar to this nature are covered much more in depth in the Alias DVD, **"Maya Techniques™ | Custom Character Toolkit"** which Paul Thuriot and myself taught as a Master Class world wide last year. This DVD is available through http://www.alias.com/ -> click on the link for the on-line store. If programming and scripting for character setup interests you, then I would highly recommend you or your organization get this DVD, which covers the entire process very carefully and in depth.

Rigging Anatomy of the Eyes

Eyes and Blinks

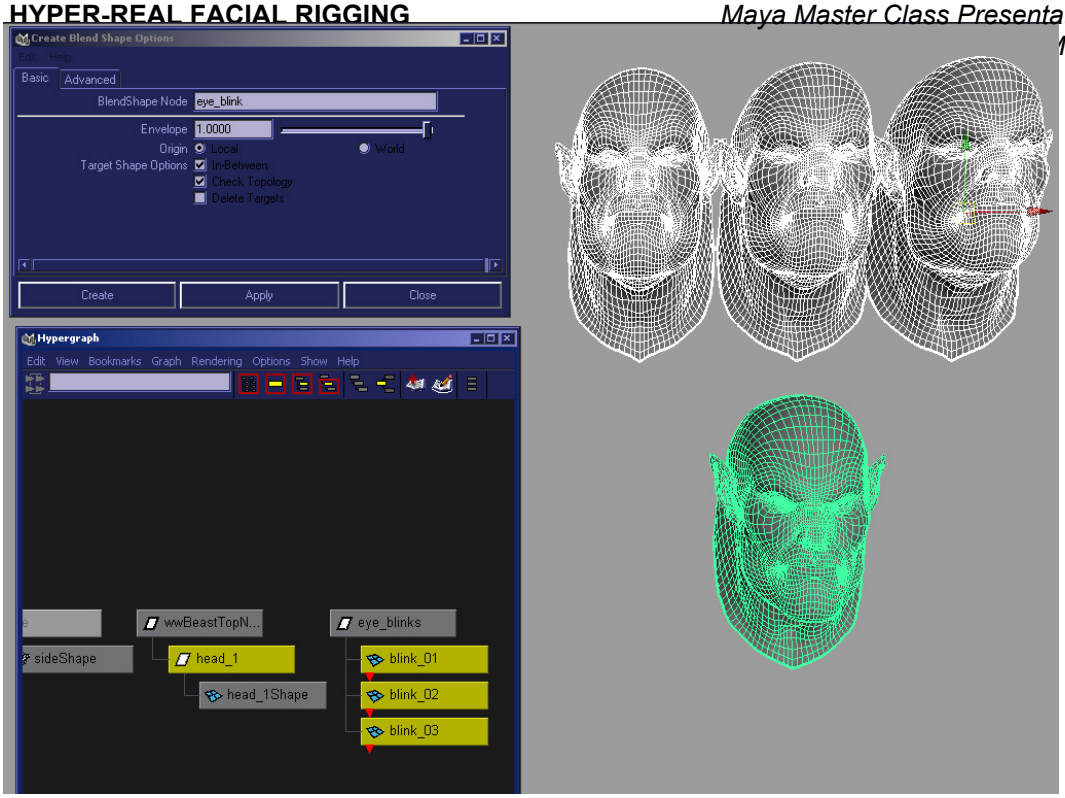
Eye blinks are actually quite straightforward set of shapes, as seen in the very illustration here to the right. The eye can be modeled as a whole and then split up into upper and lower shapes, or visa versa. Here are a few shapes that should be carefully modeled for the lids:



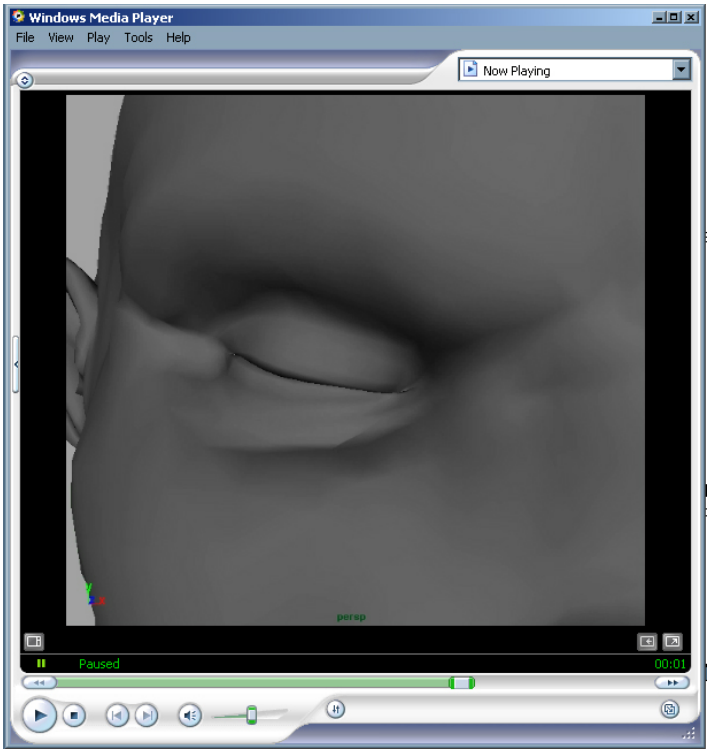


- *Wide-Eyed Open (not used for blink, but still very important)*
- Neutral Open (default- no extra work needed ☺)
- Half way closed
- Closed Blink (the moment right before the lids impact each other)
- Follow-through Blink (the moment right after the lids impact each other, and the skin forms a slight ripple just below the lower lid)

Notice the last two blink models are subtly different. The last one has compressed skin around the lids, attempting to create the effect of compression as the skin touches together during a blink.



Using the three blink models shown on the previous set of images, an in-between blend shape (shown above) was created to drive the blink.



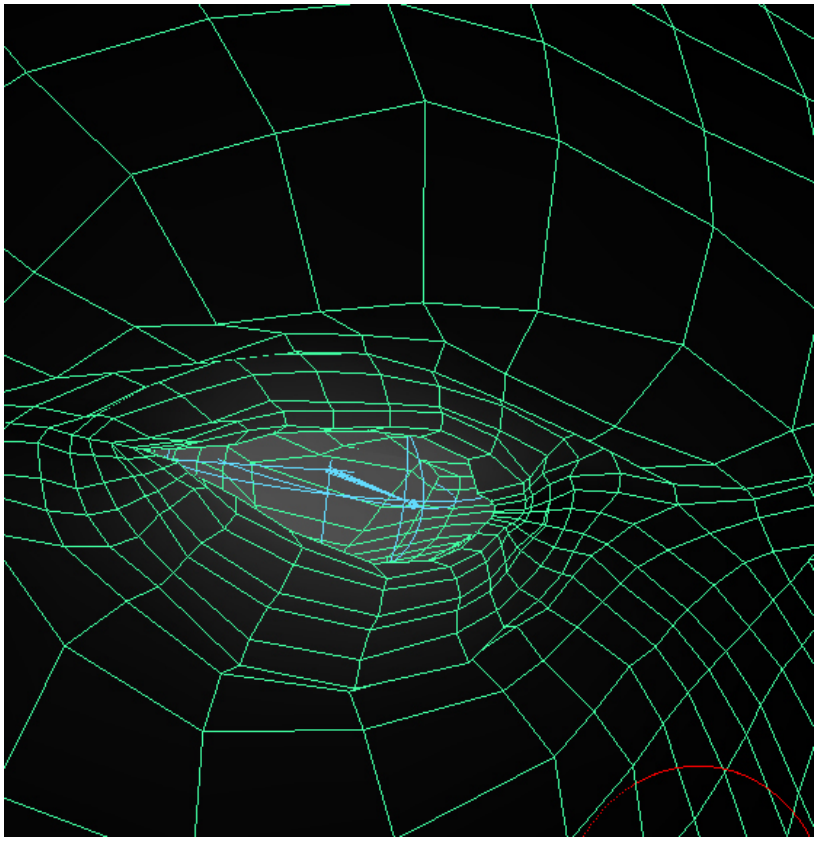
Check out the movie file, "eye_blink.avi", which shows how the mixture of shapes that cause the eye to appear more realistic as it closes.

Initially created models used to make the open and closed blink shapes can also be automatically driven as the eyes look up and down, opening and closing the lids, just as eyes do naturally, as a part of the fleshy eye setup which we will talk about next…

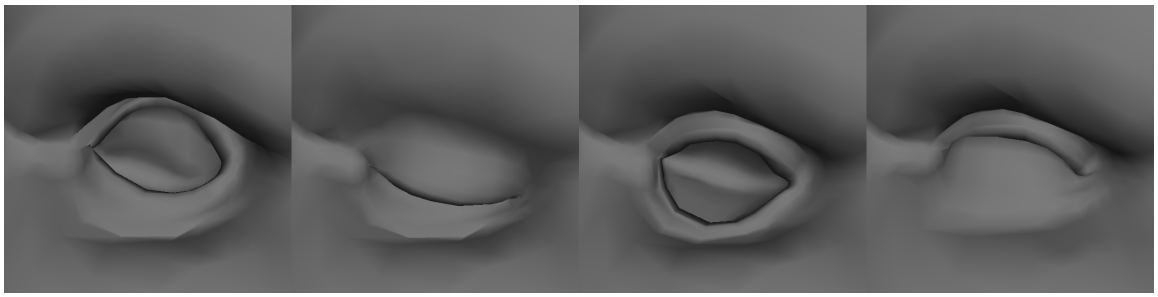
Fleshy Eyes

The fleshy eyes are a really cool part of the facial setup, because they add a really nice level of realism that is absolutely necessary. There are several ways to rig fleshy eyes, and some are more complete than others. The techniques displayed in this class will show a combination of techniques, using skinning, set-driven keys and vector utility nodes, and sculpt deformers, to drive the fleshy eye deformations.

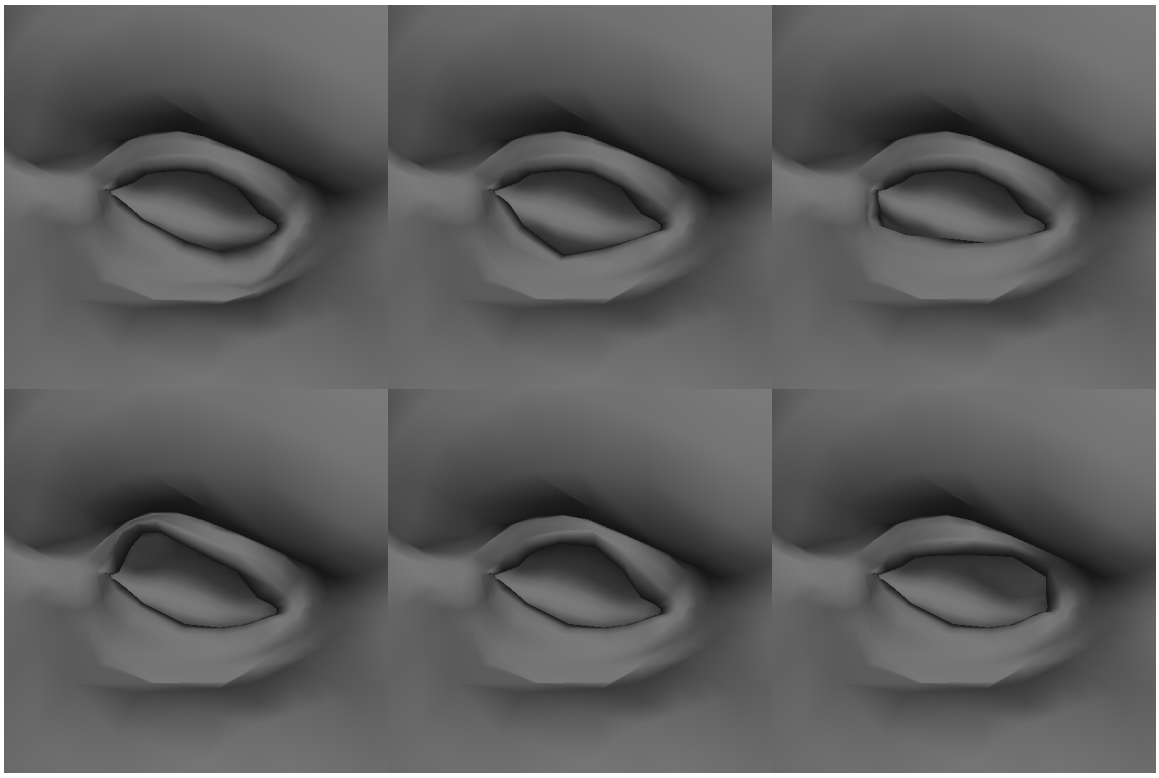
1. To start off, you absolutely need a single eye joint that has the same pivot as the exact center of the eyeball. This joint should be aligned with the plane of the face that falls across the eye socket so that one of it's rotational axis lies parallel with the eye socket, and the other two axis are perpendicular to it. This joint should be very subtly weighted onto the area around they eyes and eyelids. The weighting onto this joint should never be one, instead it should be more like one tenth, with a nice smooth fade off around the edges. Smooth flooding a few times with the paint skin weights tool usually will do the trick. This single joint is extremely important because the rest of your fleshy eye rig, including the rotation of the eyeball itself will be directly and indirectly driven by the rotation of this joint.



Notice the very subtle skin weights for the eye joint, this is just the beginning…

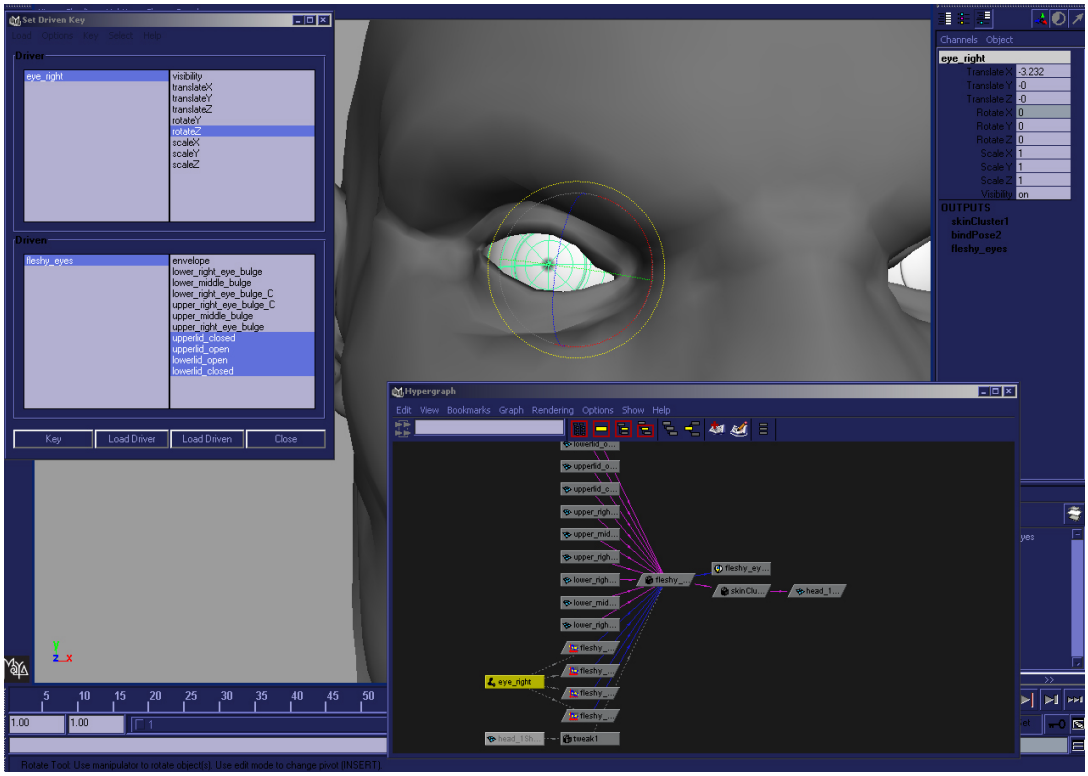


Before you start, above and below are the models you will need to create the full fleshy eyes effect. The four images on top are upper-lid and lower-lid extreme open and close.shapes. The next six images below will be used to achieve the stretching look of the skin as the eyeball looks around.



2. Now we will start off by making a front-of-chain blendshape from the above ten models, onto the skinned model that has the smoothly weighted eye joint as discussed in the previous page. Call the blendshape “fleshy_eyes”.

3. Next, lets do a set driven key based on the upward axis of the eyeball. When the eye angle is facing upwards, we will change the open close targets to move both the upper and lower lids up to maintain the natural shape of the eye. The same goes for the downward angle.

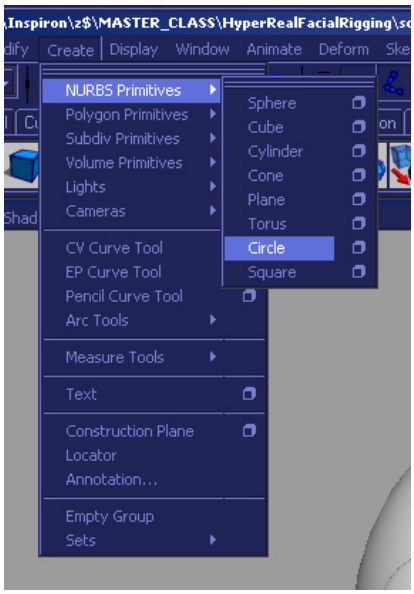


Set driven key of the open close blendshapes based on the eye angle.

Once that part is looking good, you should have a relatively decent looking fleshy eye. You could stop here, and call it quits if you don't need a really good looking eye setup, but, if you do, then read on…

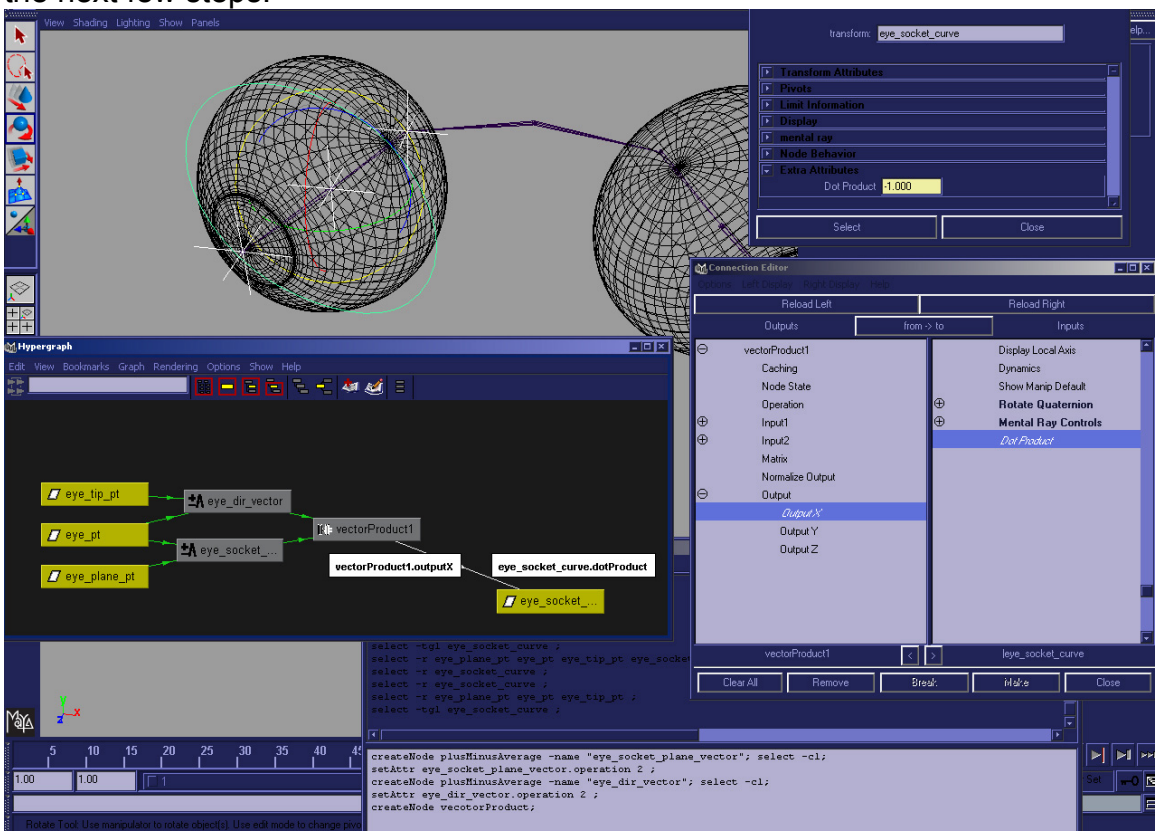
4. Next, we will use a combination of a couple subtraction nodes piped into a dot product along with the closest uv position on a nurbs surface to drive the rest of our fleshy eye shapes (the ones that look like eye is pushing the eyelid skin around).

5. First, create a nurbs circle, and position it and shape it to be roughly an outline of where the eye-socket on the skull would be.

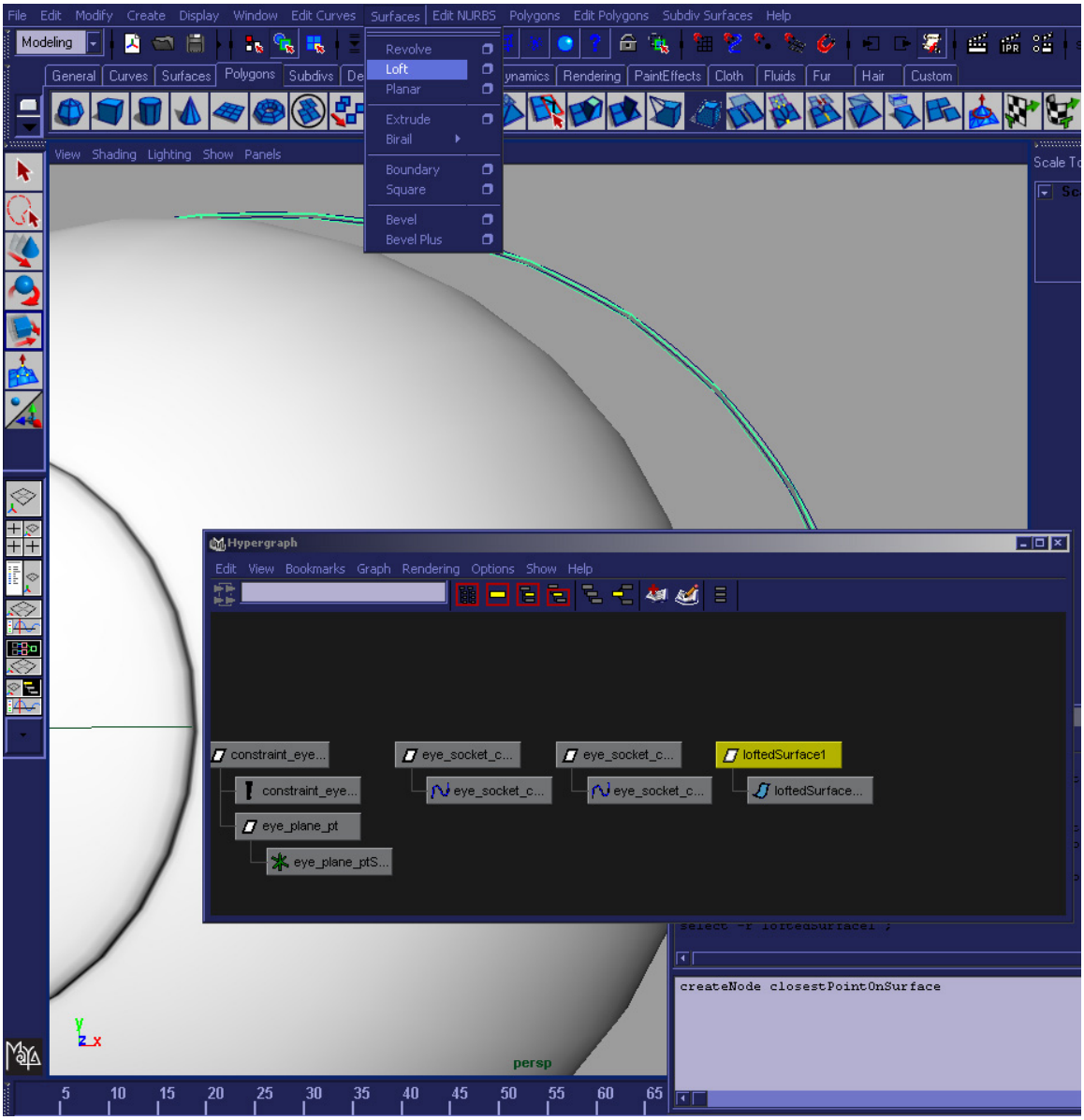


6. Next, create three locators, all should be in world space, and should NOT have their

transforms frozen. This is important because we will need the translate values from them. Point constrain the locators to the appropriate joints as seen in the image below. Next, connect the translate values of the outer locators into a subtraction node each with the locator that is in the center of the eye point. This will give you two vectors which you can simply use to get a dot product from (using the vector product node as shown) -- in order to multiply the eye shapes by a number between zero and one in the next few steps.

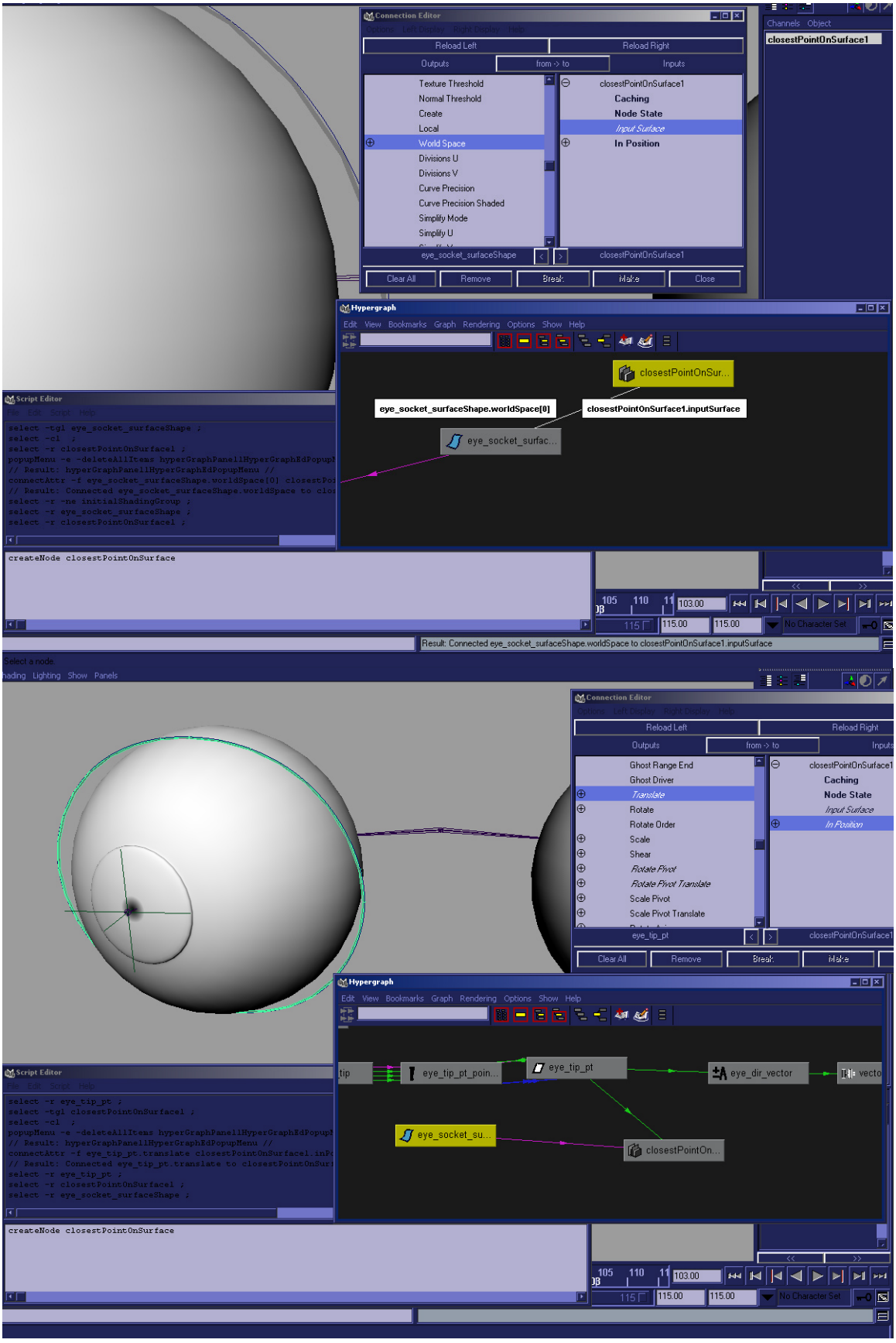


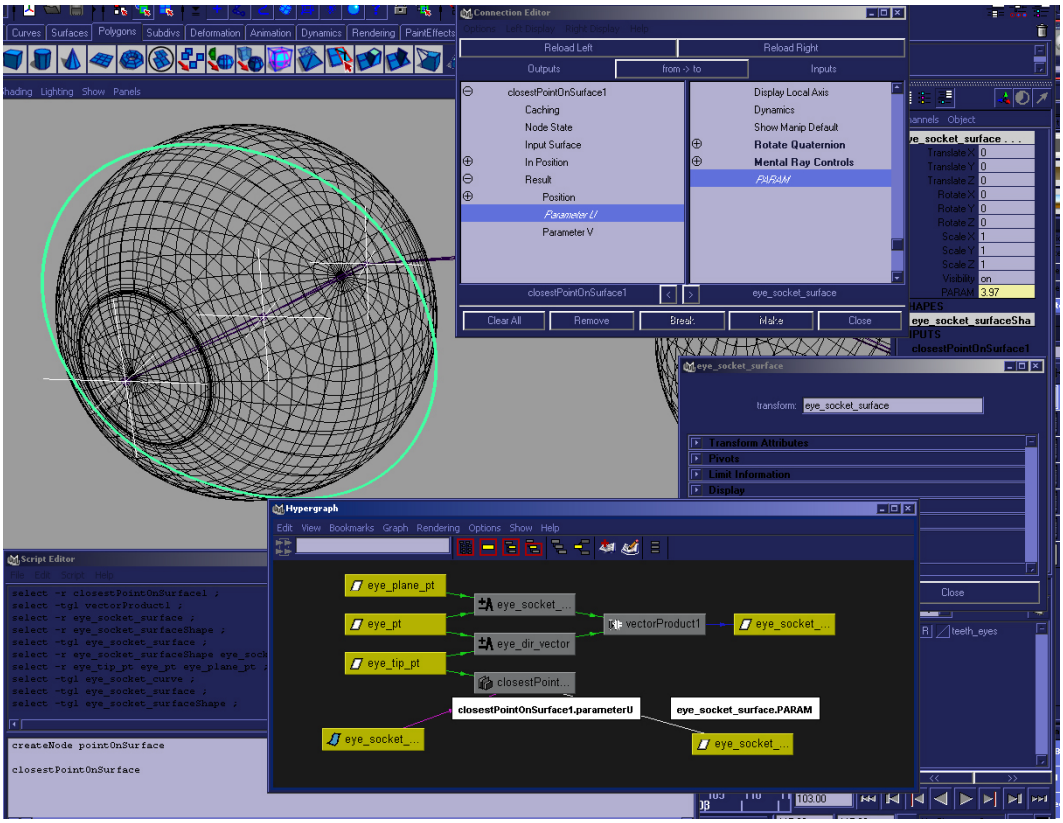
7. Before we use the dot product, we need to use something to reliably drive the blendhapes of the fleshy eyes using simple set driven keys. Since the eye is restricted to 2 axis (ie it doesn't spin), the closest uv values of a nurbs cylinder that is in the shape of the eye socket will work perfectly. So, first, create the nurbs surface however you want, I did it by lofting the curve I had previously created in the last few steps (as shown in the next image).



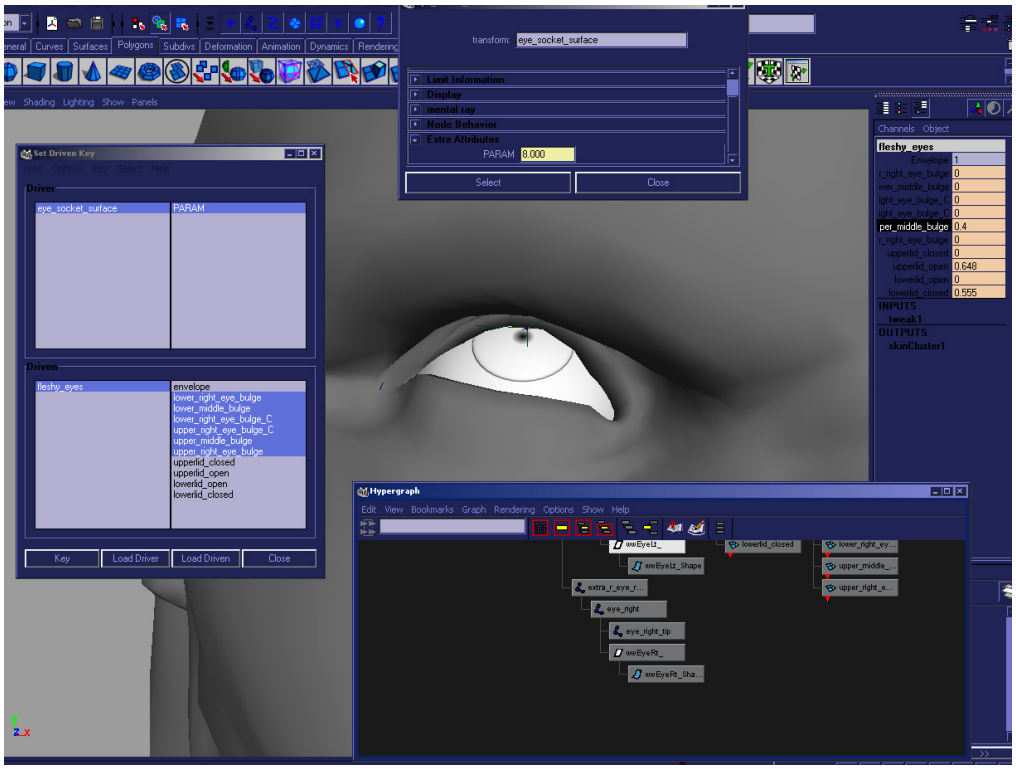
Next, create a closest point on surface node, and input the .worldSpace[0] attribute from the nurbs cylinder that you just created along with the eye-tip locator in order to calculate what the closest point is on the surface from the tip of the eyeball when it rotates around – this is a useful technique because it is angle independent, and outputs a single value which interpolates properly, and can directly represent the blend shape target that the eyeball should be driven to. The only problem is that it needs to be multiplied off as the eye rotates back toward it's center default position, and this is exactly what we have the dot product node sitting there waiting for.

8. Make the connections between nodes as seen in the following diagrams:

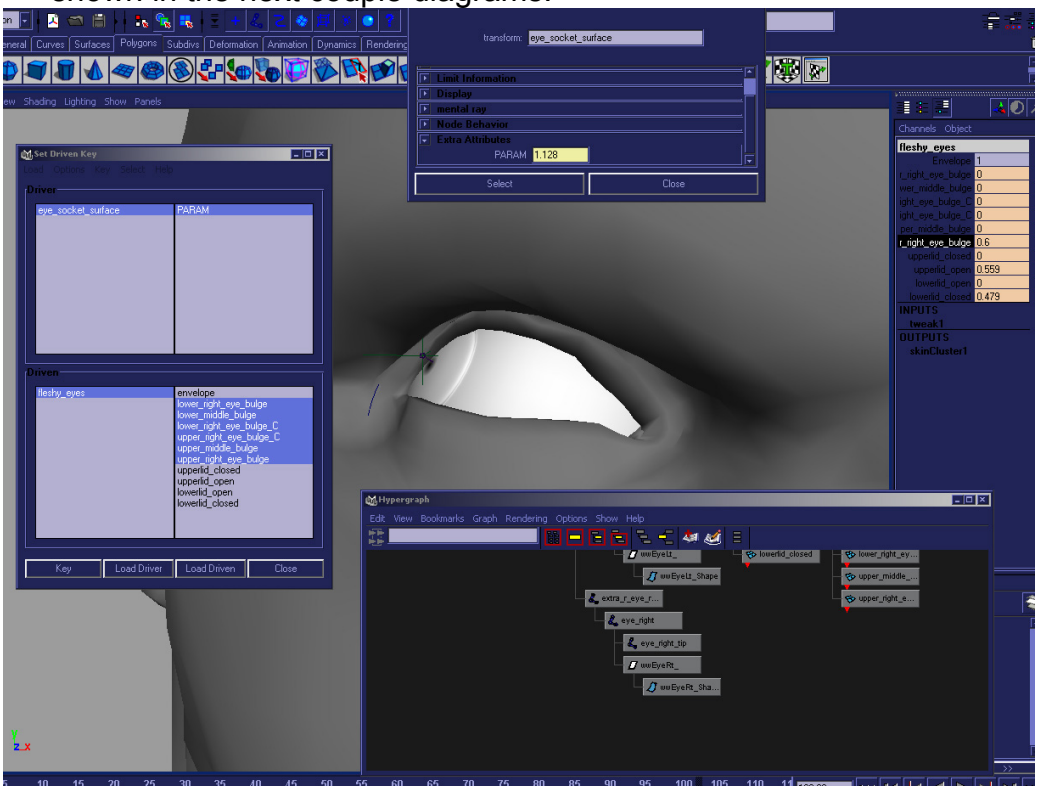

Connection Editor
Reload Left
Reload Right
Outputs
from -> to
Inputs
Texture Threshold
Normal Threshold
Create
Local
World Space
Divisions U
Divisions V
Curve Precision
Curve Precision Shaded
Simplify Mode
Simplify U
closestPointOnSurface1
Caching
Node State
Input Surface
In Position
eye_socket_surfaceShape
Clear All
Remove
Break
Make
Close
Hypergraph
Edit View Bookmarks Graph Rendering Options Show Help
closestPointOnSur...
eye_socket_surfaceShape.worldSpace[0]
closestPointOnSurface1.inputSurface
eye_socket_surfac...
Script Editor
createNode closestPointOnSurface
Result: Connected eye_socket_surfaceShape.worldSpace to closestPointOnSurface1.inputSurface
Ghost Range End
Ghost Driver
Translate
Rotate
Rotate Order
Scale
Shear
Rotate Pivot
Rotate Pivot Translate
Scale Pivot
Scale Pivot Translate
eye_tip_pt
eye_tip_pt_poin...
eye_dir_vector
eye_socket_su...
closestPointOn...



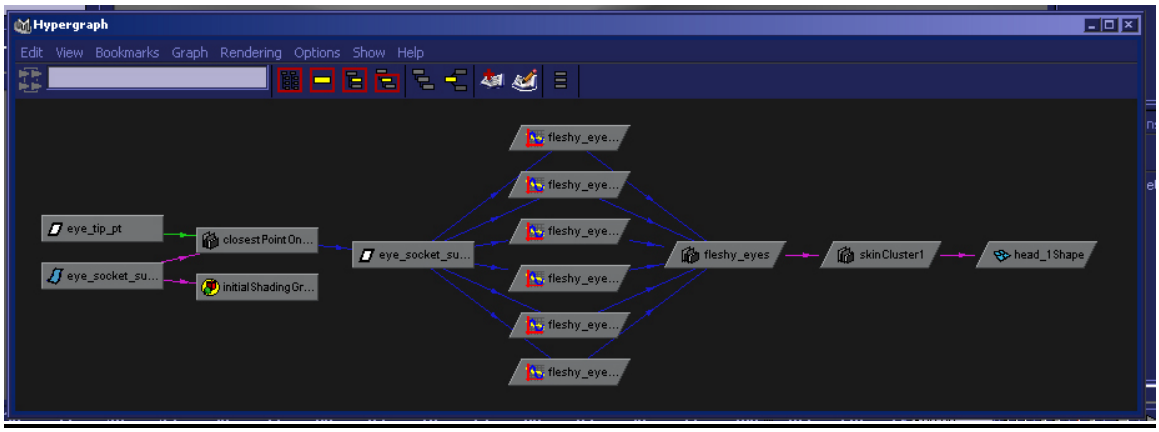
9. Now, move the eyeball around w/ the eye joint, but use set driven key to drive the different blend shape values depending on the u value of the



closest point output from the tip of the eye ball to the nurbs cylinder as shown in the next couple diagrams.



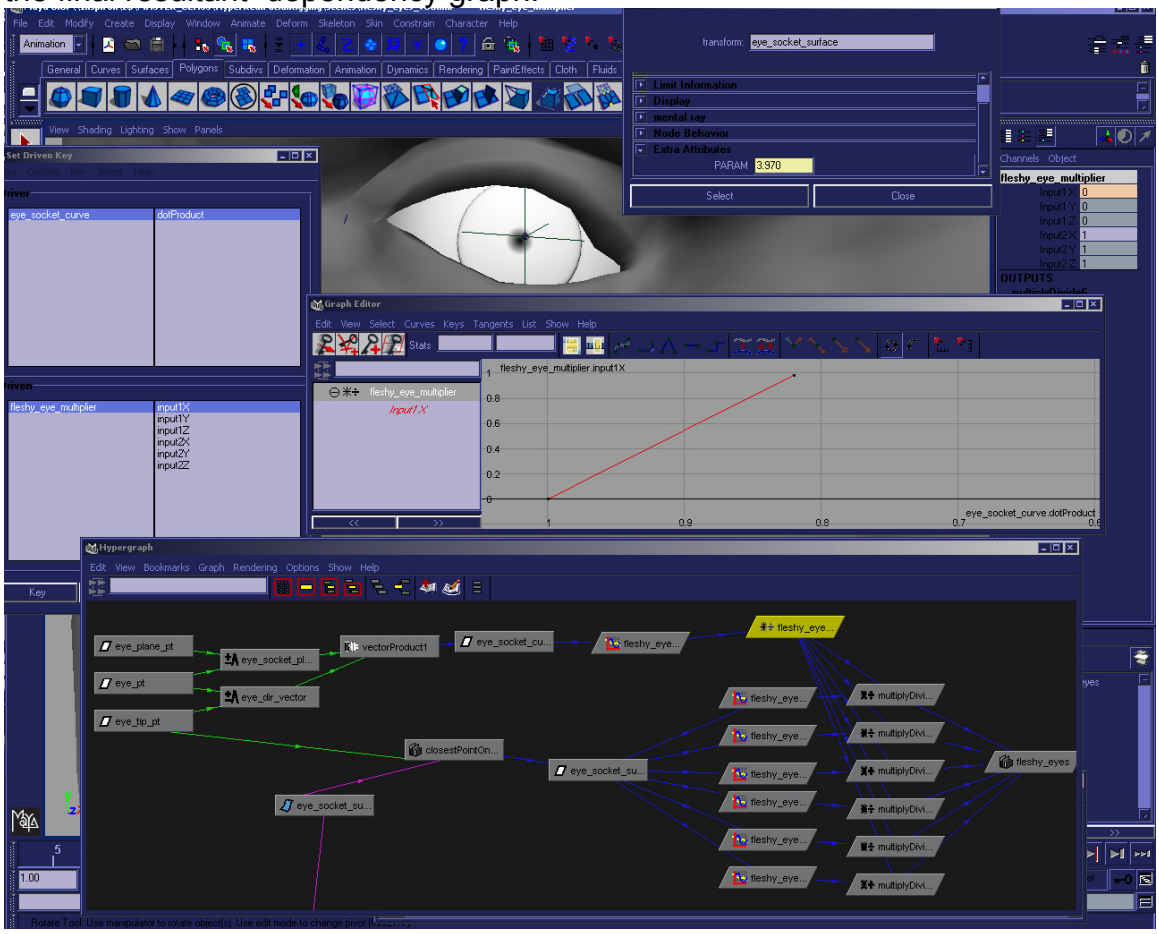
The resulting input/output node network from the set driven keys should by default look something like this:



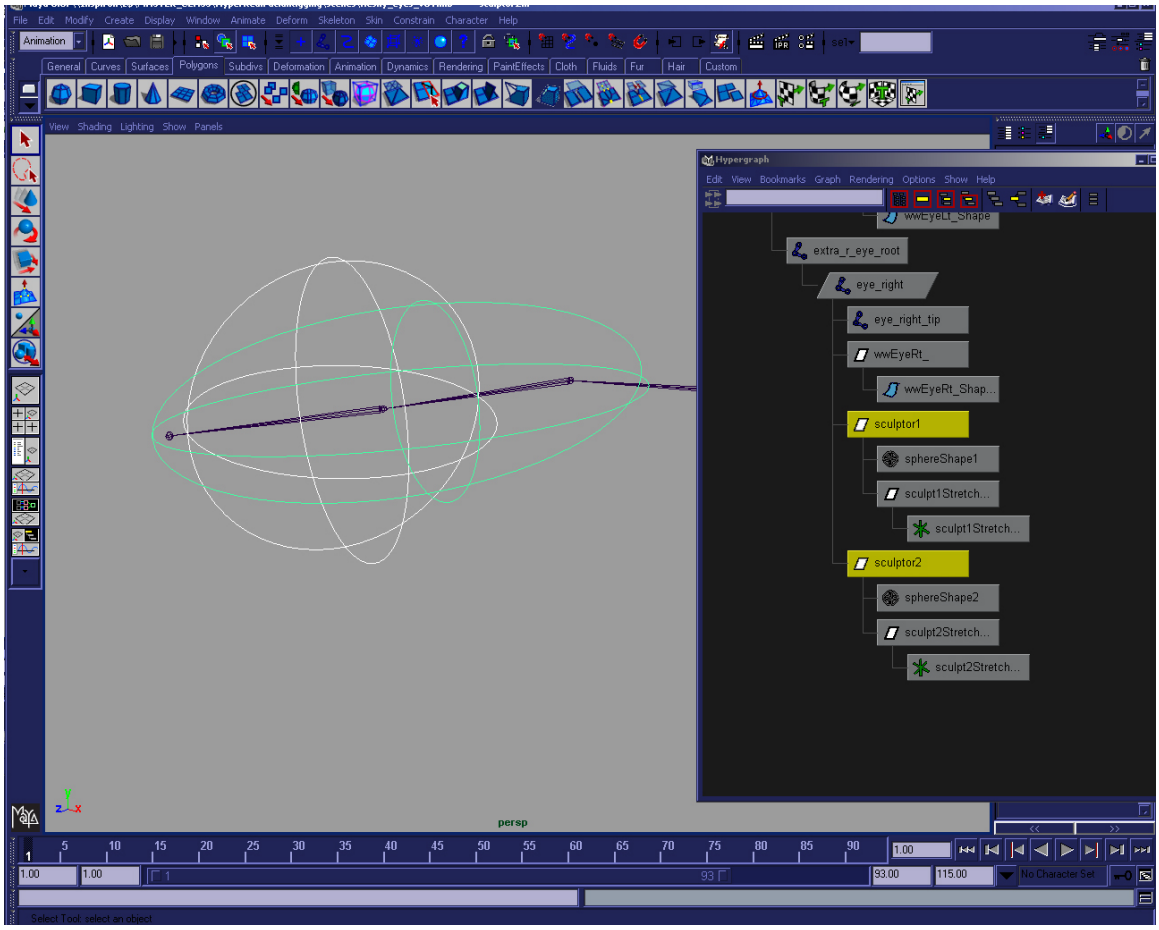
Next, all we need to do is insert some multipliers between the set driven key frames and the blendshape attributes based on another set driven key frame which uses the dot product between the eye vector and the original opposite eye vector as it's driver to determine the multiplier. The dot product automatically gives us a gradient between zero and one, but we will use a set driven key to

further tweak this gradient value so that it can give us exactly the fleshy eye look we want.

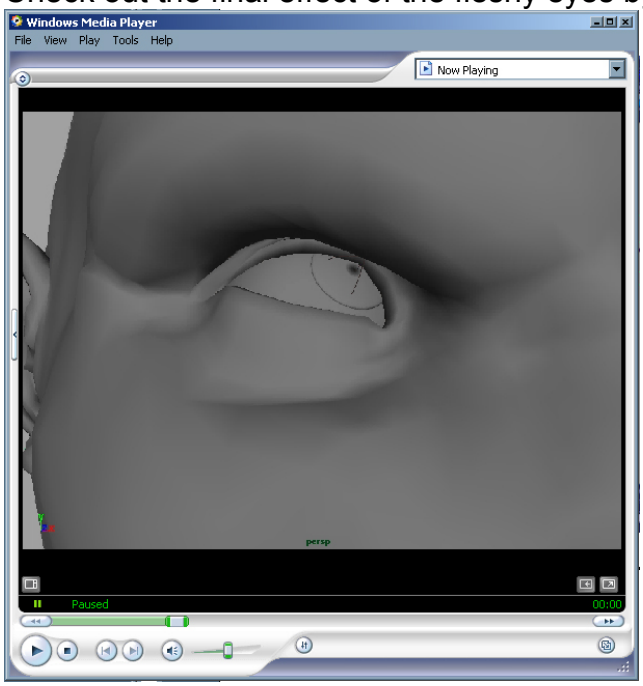
Take some time and insert multipliers between each set driven key frame, and then hook another single multiplier into those nodes. The highlighted multiply node in the following image is the additional multiplier. Next, simply do a final set driven key by moving the eye around w/ the eye joint, but using the dot product as the driver for the highlighted multiply divide node. The following image shows the final resultant dependency graph:



Finally, additional semi-dynamic looking sliding and bulging of the eye can be added with sculpt deformers – you want to have one sculpt deformer for the round section of the eyeball, and an additional sculpt deformer for the bulged portion that surrounds the iris of the outer eyeball as seen in the image below. Parenting the sculpt deformers to the eye ball joint will usually suffice. I also tweaked the max displacement and drop-off distance attributes to my visual liking, and then turned down the envelope values of the sculpt nodes, so that their overall effect was multiplied off just enough when combined with the rest of the effect.



Check out the final effect of the fleshy eyes by viewing the following movie file:



Take a look at "fleshy_eyes.avi" to see the final outcome of this rig

Pupil Dilation

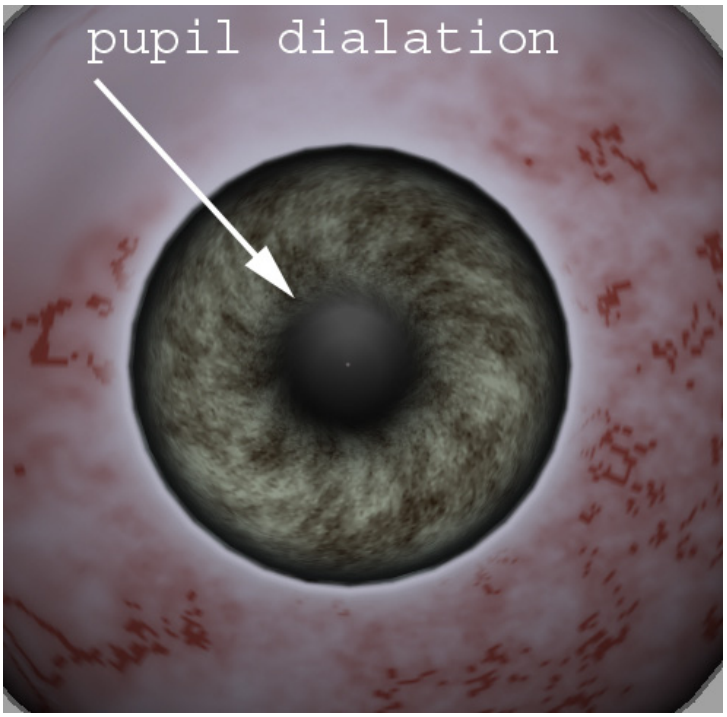
The pupil dilation can be done very simply with a couple of blend shapes on the iris geometry, it's so simple! ☺ Watch the movie file, and you will see it is a relatively acceptable looking effect that could be easily combined with some shading tweaks to really trick it out if you wanted to.

Basically all that is needed to achieve this look is to create 2 modeled blend shapes on the eye geometry. The modeling should occur with a hardware shader attached to the geometry in high quality display mode which allows the modeler to see the way changing the model is warping the texture mapped area which appears to cause pupil dilation:

1. A fully dilated eyeball model: A model that expands the center isoparms outward along the averaged plane of the iris, expanding and stretching out the uv coordinates on the area of the nurbs eyeball surface that the iris texture map will be attached to.

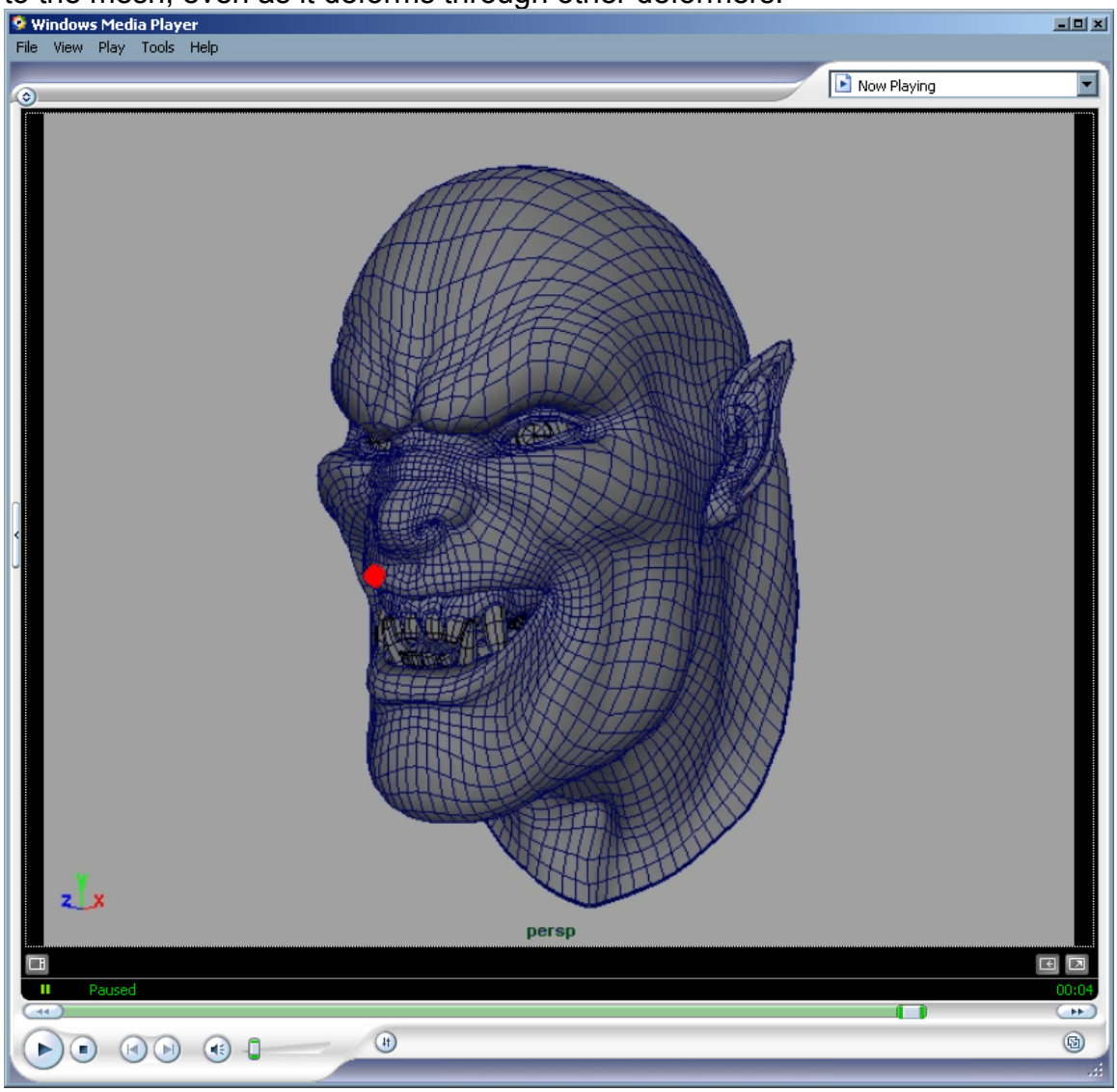
2. A fully non-dilated model: This would be a super beady-eyed model that is basically looking as though there is almost no pupil at all, and is meant to be pretty much opposite the model just listed in the last step.

   Check out the movie file iris_dialate.avi:



Rigging Cluster on Mesh "Stick-On" Face Controls

The cluster on mesh controls are really handy to use as additional translation controls to move the mesh around, and are really handy because they stay stuck to the mesh, even as it deforms through other deformers.

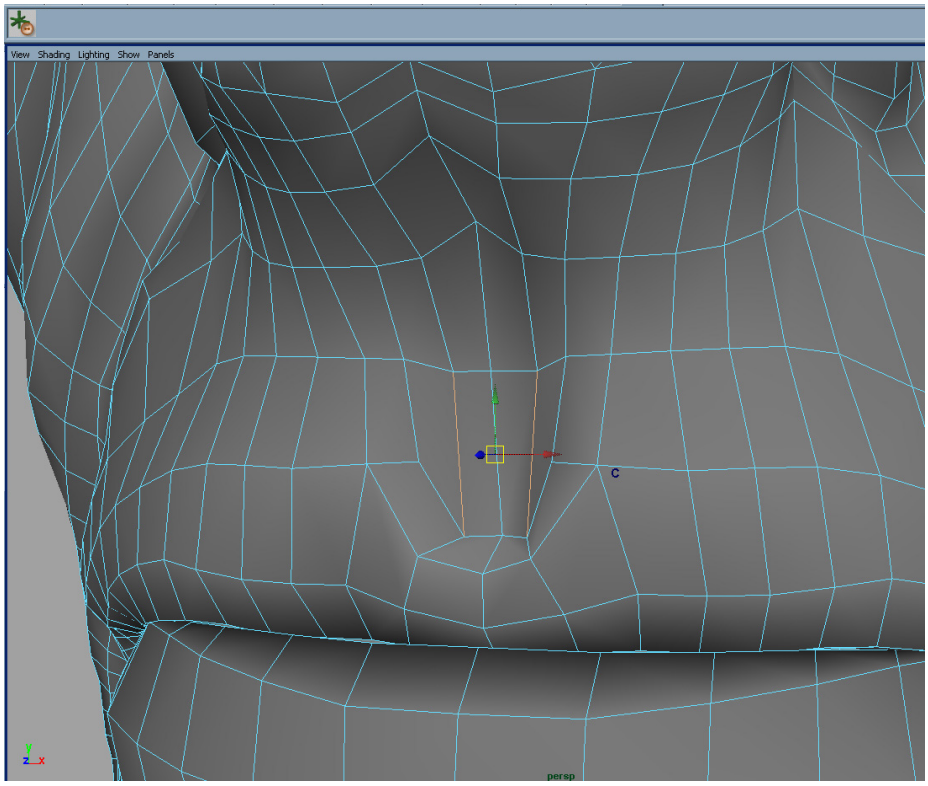


Check out the avi file in the movies directory called cluster_on_mesh_control.avi – the red dot is the controller, and if you notice, it moves with the joint deformations, and with the blend shapes and stays attached to the exact correct spot for tracking it's motion across the polygon face.

The cluster on mesh setup is made much simpler because of a MEL script written by Michael Bazhutkin called rivet.mel. You can find this script available for download from the Maya -> MEL section on http://www.highend3d.com/ as well as from Micheal's website, http://www.geocites.com/bazhutkin/ .

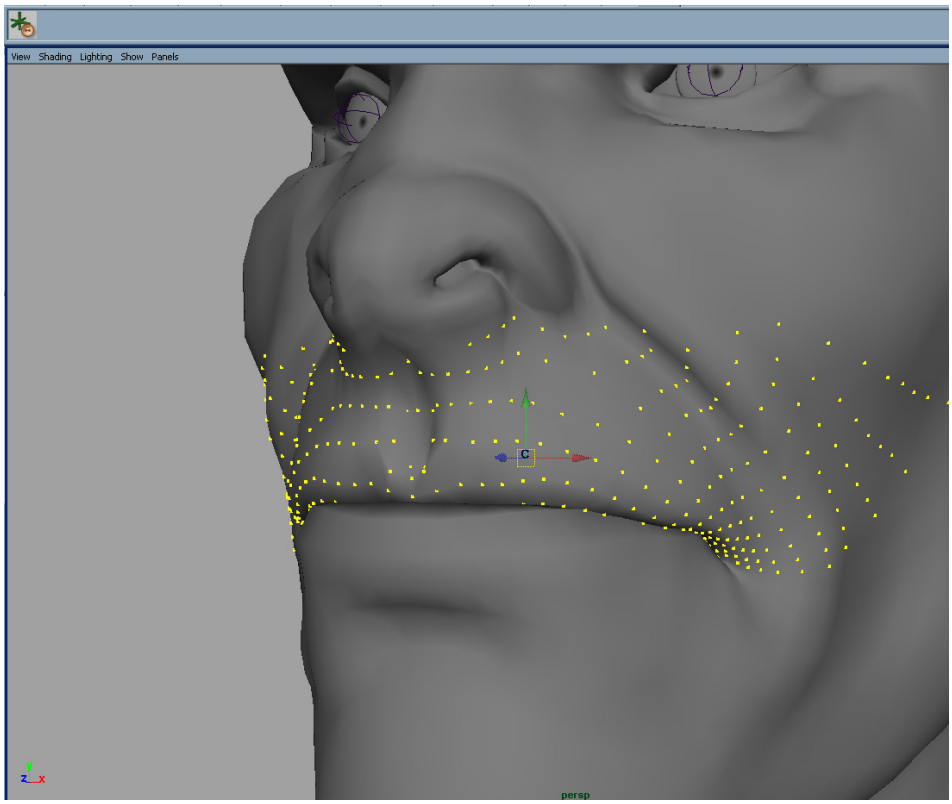
This script really rocks because it uses a tricky way to constrain transforms onto polygon faces using built in Maya nodes to internally build a small nurbs patch across the edges of face of a single polygon that you are constraining to, and then use the already existing nurbs surface info nodes to pull the transforms. Hat's off to Mr. Bazhutkin for using Maya's built in nodes, and not resorting to writing a plug-in to do something that is already available in the software. This is a super useful script because it allows you to really easily and interactively constrain objects to polygon surfaces, and I would highly suggest using it as frequently as possible to solve interesting problems that require you have something stuck onto a polygon object for calculation purposes.

1. So once you have downloaded and installed the rivet script, you will have a little button on your shelf like the one in the following diagram. All you have to do is select the two parallel edges that you want the script to do it's loft calculation across, and when you hit the button you will have a locator that will stay stuck to that point on the mesh!



Using the rivet script is fast, easy and incredibly powerful for rigging and deformation setups.

2. Next, go ahead and create a cluster on some vertices of the face, like the next image, and then paint some nice weights for it.



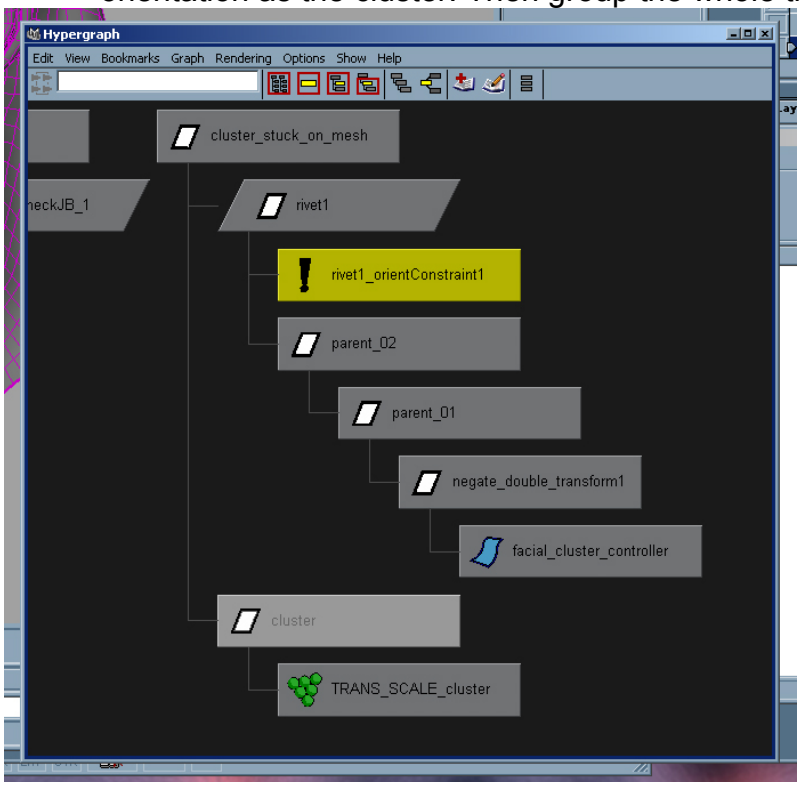
Creating the cluster portion of the cluster stuck on mesh control is super easy, just create a cluster, make sure your control has the same orientation and pivot /w the cluster, connect it's .translate and .scale attrs, and then hook the rivet script's output to control the controller's top parent group. After that it is a simple matter of following a few extra steps grouping and setting some attributes detailed here and on the DVD!

3. Next, delete the aim constraint that is attached to the rivet locator node, and make a weighted orient constraint, weighted between the two jaw joints. I chose a weight value of 0.90 for the upper jaw and 0.10 for the lower, since the control is on the upper lip.

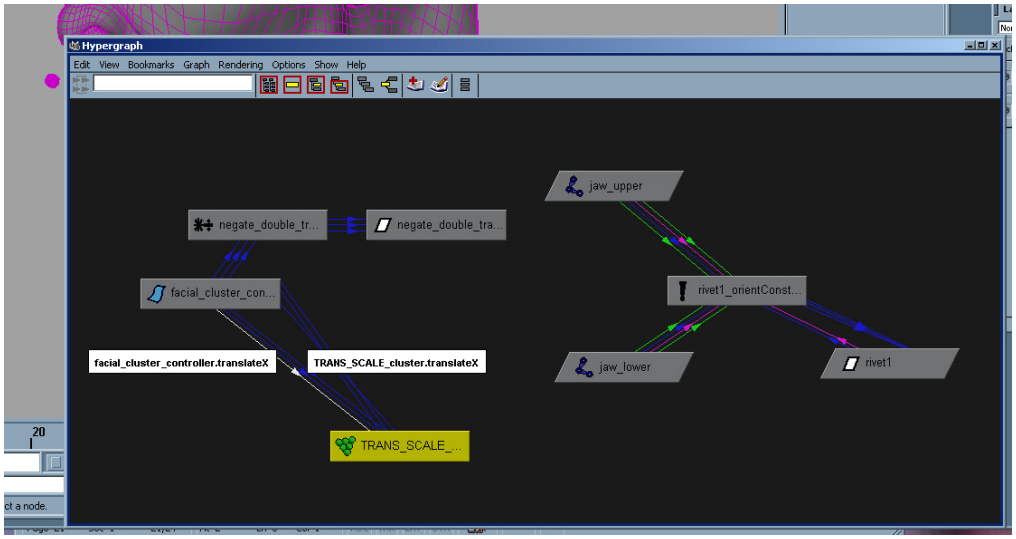
4. All you have to do now is make a couple of extra nodes – for the control object I like to use a nurbs surface that has it's override color set to something bright and has it's shader dis-connected. This creates this really cool looking circle in shaded mode that is completely opaque, like in the first diagram… but you can use anything as a transform: nurbs curves for example or even another locator if you want.

5. Then, just make sure that your control is under a couple group nodes which all have their transforms positioned and oriented exactly in the same direction as the cluster that was created, and their transforms are all zeroed out as far up the hierarchy as they can be. Also the cluster's .inhertsTransforms attribute should be switched off, and it should be set to relative mode, this is the most important part of this setup. Just parent the

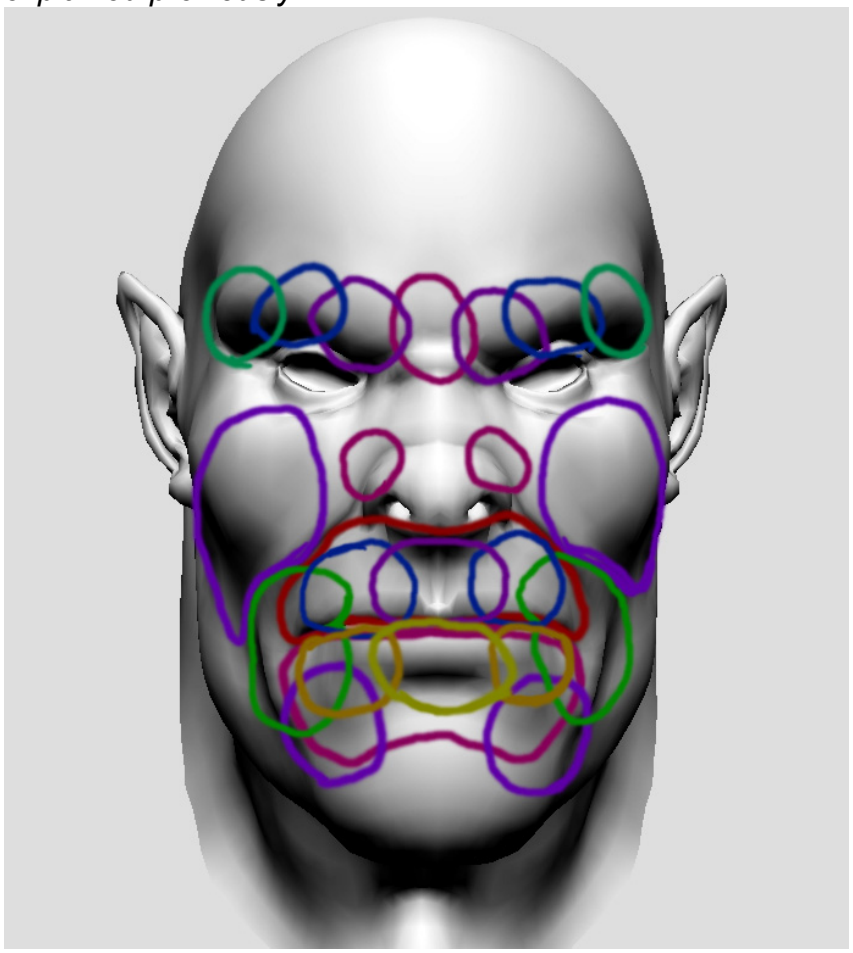
control's hierarchy under the rivet once it is in the same position and orientation as the cluster. Then group the whole thing as seen here:



6. Next, all you need to do is connect the translate and scale attributes of the control's transform to the cluster's transform. Lock the rotate values on both nodes because hooking up rotation is a big pain that causes dg cycles and matrix product nodes that we probably don't want to get into. There is a double transformation issue that occurs, and so it is band-aided up by multiplying the facial_cluster_controller node's translate values by -1.0 and then feeding those values back into the parent. This is a loop on an implicit connection, but it seems to do the trick, and fix the problem anyhow (surprise).



Here is the dependency graph view of the rigged up connections that were just explained previously.

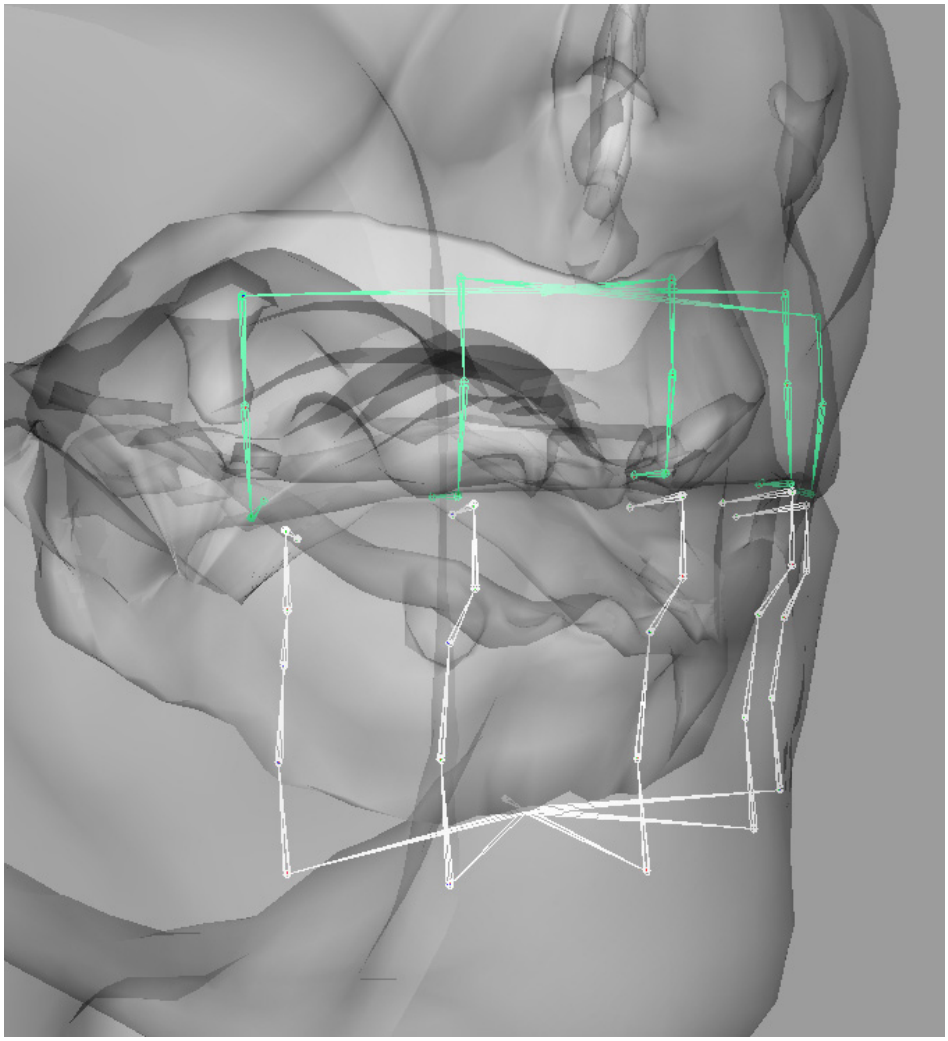


The example file has one single cluster rigged into our creature's face. In the DVD we show you how to make a second cluster. For a complete facial rig, using many with nice falloffs is the way to go. These regions shown in the diagram above work the best. And since they move with the deforming mesh, you can add as many as you want, and they will always stick right onto that spot.

Rigging of the Lips

Rolling Lips and Squashing Lips

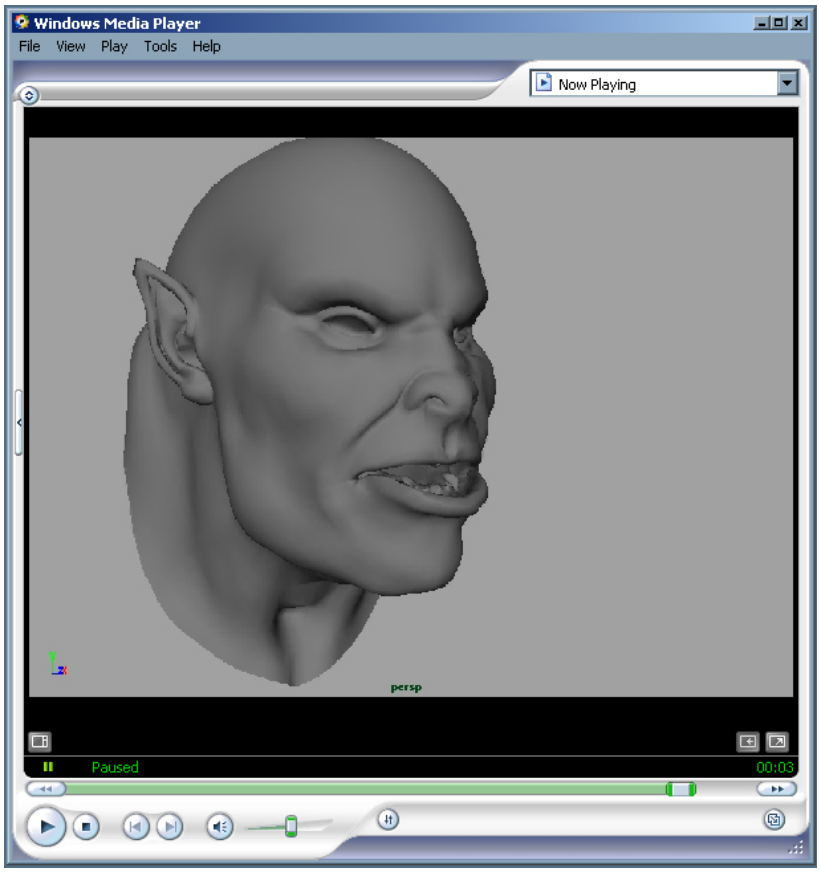
The rolling lips and squashing lips can be done using the exact same technique. Basically the idea is that you will rig up a joint hierarchy that can be very simply set-driven keyed to create the effect. This is done on the base mesh model, and then the file is saved as a totally separate component which then gets brought into the main rigging file and added as an additional blend shape to the character.



Joint hierarchy used to create the rolling lips

This gives you control on two different levels. First, you get the control of rotational joints to do things like rolling lips, you really do need to have rotations, or it just doesn't look real – you can have multiple in-betweens, but then the modeler is stuck with more work than is really necessary, since it is really a pretty

quick thing to weight some joints into the lips to get a rolling effect. It is highly recommended, though, that once the nice smoothly painted weights give you what you are looking for, you do a corrective blend shape to get the "exact" rolled open lip appearance that will keep the character "in-model".



Watch the movie file "rolling_lips.avi" to get an idea of the type of rotational control using set driven joints w/ corrective shapes for certain lip controls can get you.

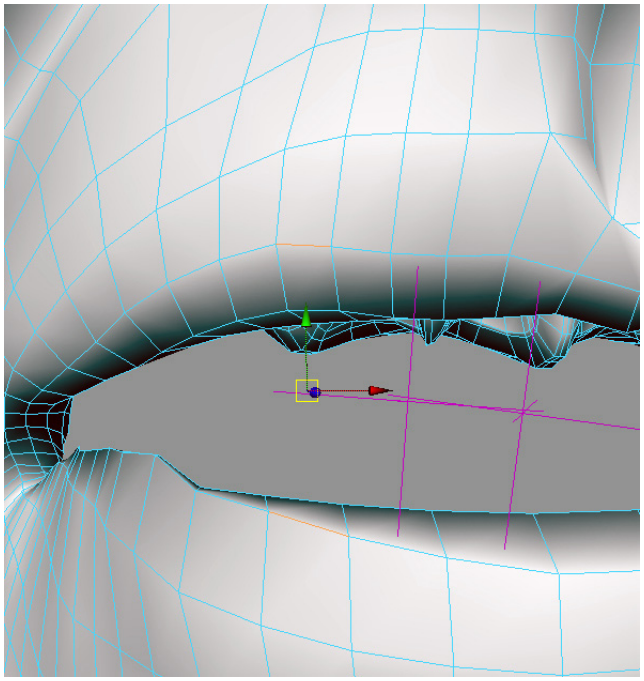
Sticky Lips

The sticky lips setup presented here is probably the final and most involved section of the geometric rigging portion of this course. The sticky lips are definitely meant to be the final touch on the facial rig – and due to the technique used, should also be applied at the very end of the deformation chain. Sticky lips should always be the last deformer so that they always have the last say about "where" the lips go in order to stay stuck together properly. Note that sticky lips are meant to be controlled by the animator. There is a common mis-conception that they should be "dynamic" or "automatically sticky" – unfortunately, as cool as it would be to rig something this way, it is not a very productive way to get good facial animation. Animators know exactly how and when they want the lips to be sticky, and therefore the means to make them sticky is what we will be rigging. A completely user controlled attribute will be the result which will allow the animator to stick and un-stick the inner regions of the lips together in a non-linear manner, at any time during his animation process.

The rig will take special care to try and get a nice appropriate centerline for the lips so that the fact that they have world space deformation precedence over the subtle inner skin lining the inside edge portion of the lips does not distract from the desired shape of the lips prior to the application of stickiness. Once again, sticky lips make strong use of the super cool rivet script (introduced in the previous cluster on mesh controls section) for attaching transforms directly to polygons.

1. First, start out with a facial rig that contains a jaw and perhaps some blend shapes (the shapes are not important, but having a nicely weighted jaw is very important). The idea here is that there is jaw rigging already in the file you will begin working on which will open the mouth (so that you have something there to rig the lips stuck together when the mouth opens).

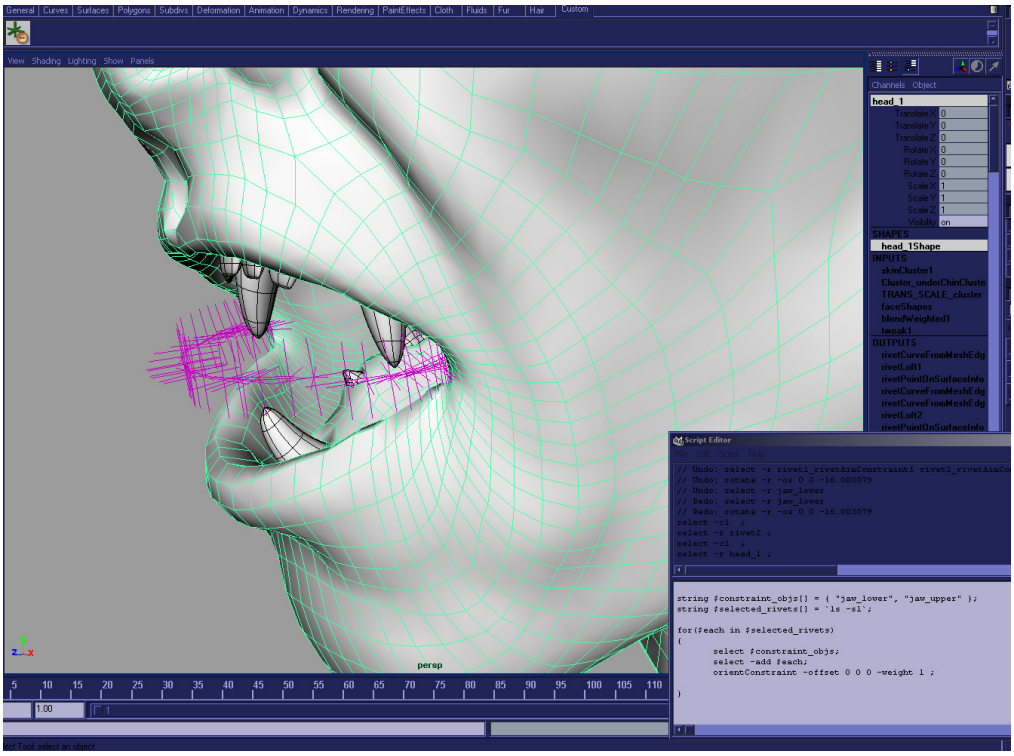
2. Begin with a partially open jaw and mouth, and begin applying rivets using the rivet script by selecting a top edge on the upper lip, and a bottom edge on the lower lip, and hitting the rivet button. Make sure you choose the outer edge of the lip, which will be the edge line loop of the lip's outline. Be sure not to select any edge that is directly connected to any vertices that you plan to be "sticky" later. It is ok if the edge is in the same face as a vertex you plan on making sticky, as long as the edge you are picking is not directly affected by one of the sticky lip vertices when the vertex it is moved.



Begin selecting edge pairs on the top and lower lips and make rivets to create the "center line" of the lips when it is in any open deformed position.

The goal is to attempt to form the mouth's centerline at any point and during any deformation (including "ooo" shapes and irregular bent or curled lip lines).

See following image:



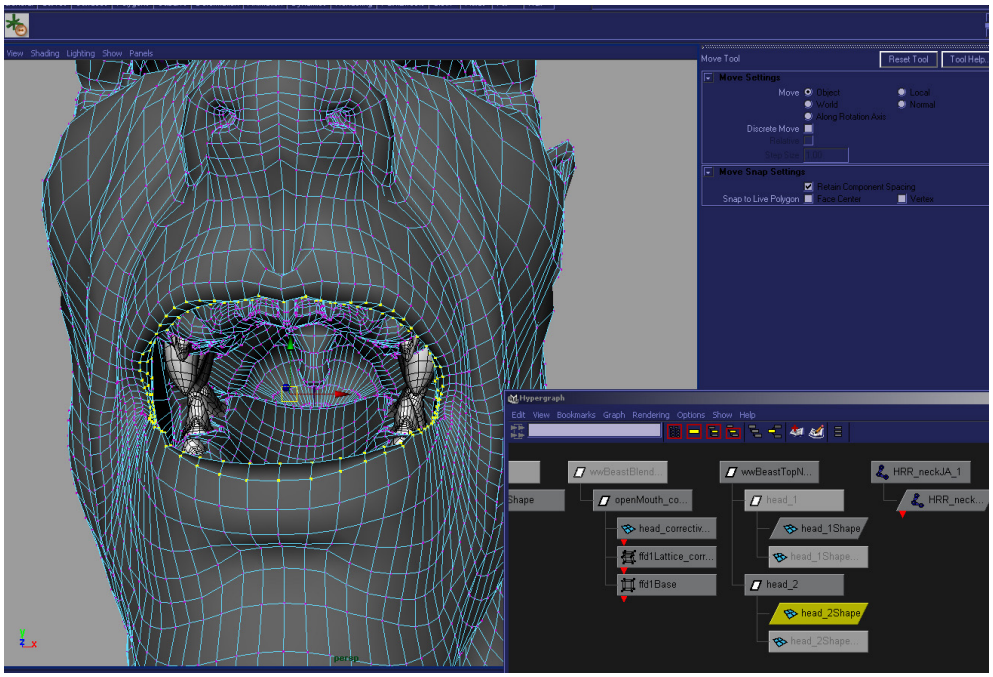
3. Delete the aim constraints that are on the rivets, and orient constrain each locator to the upper and lower jaw joints. You can use a really simple MEL loop to automate this; and a task, which could possibly take an hour, is reduced to mere seconds.

```
/////
// sticky lips -- after all the rivets have been applied to
// form the center line of the lips, use some simple
// MEL scripting to automate an otherwise tedious rigging task:
//
////
// select the all the rivet locators first, and then run this code:

string $constraint_objs[] = { "jaw_lower", "jaw_upper" };
string $selected_rivets[] = `ls -sl`;

for($each in $selected_rivets)
{
    delete `listRelatives -type aimConstraint $each`;
    select -r $constraint_objs;
    select -add $each;
    orientConstraint -offset 0 0 0 -weight 1 ;
    scale -r 0.2 0.2 0.2 $each;
}
```

4. Next, starting from a brand new base shape model (ie: the original un-deformed model), re-name it to something like “sticky_lips_head”, and select the vertices on your model that you will want to be “sticky”, or that you want to stay stuck together when the mouth opens (they will be controlled with a user controlled attribute slider that we will add in the upcoming steps).

Important tip (but be cautious): I used a blend shape from the model with the jaw in it, onto the new sticky lips base shape model to temporarily open the mouth while selecting the vertices and performing the next few steps (in order to simplify the vert selection process), but was absolutely sure to delete the blend shape node and completely close the mouth on the jaw model (BOTH VERY IMPORTANT) before moving on to step 9.

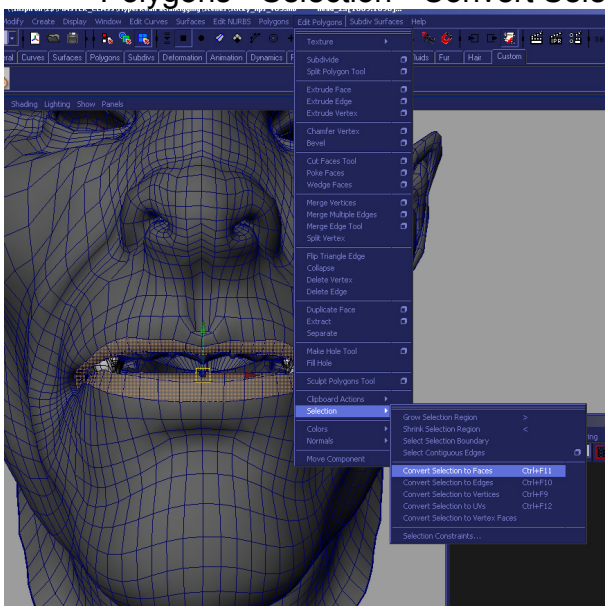
5. Make a quick select set so that it is quick and easy to select them again later.



Choose the vertices that you want to be "sticky" and make them into a quick selection set.

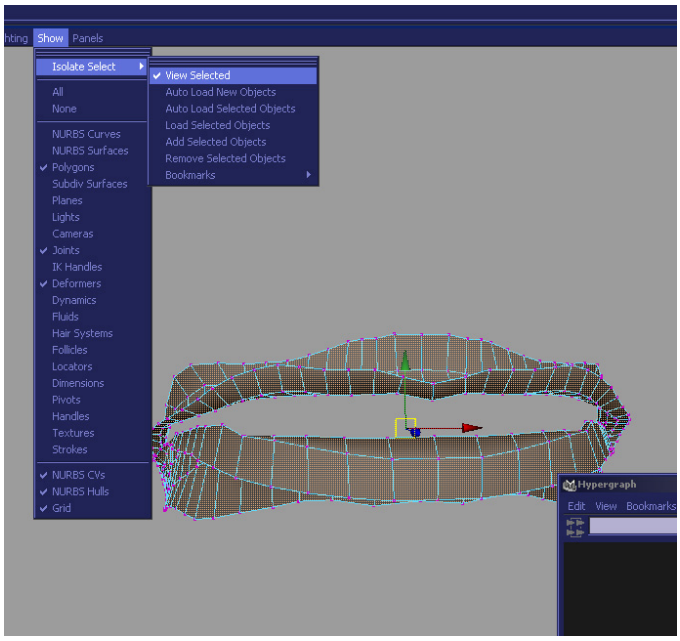
Next, we will want to try and clean up the view for the following steps so that it is easier to work on the lips, i.e.: select and operate only on this certain set of vertices.

6. Convert the selection of vertices into a selection of faces using the "Edit Polygons->Selection->Convert Selection To Face" menu command.



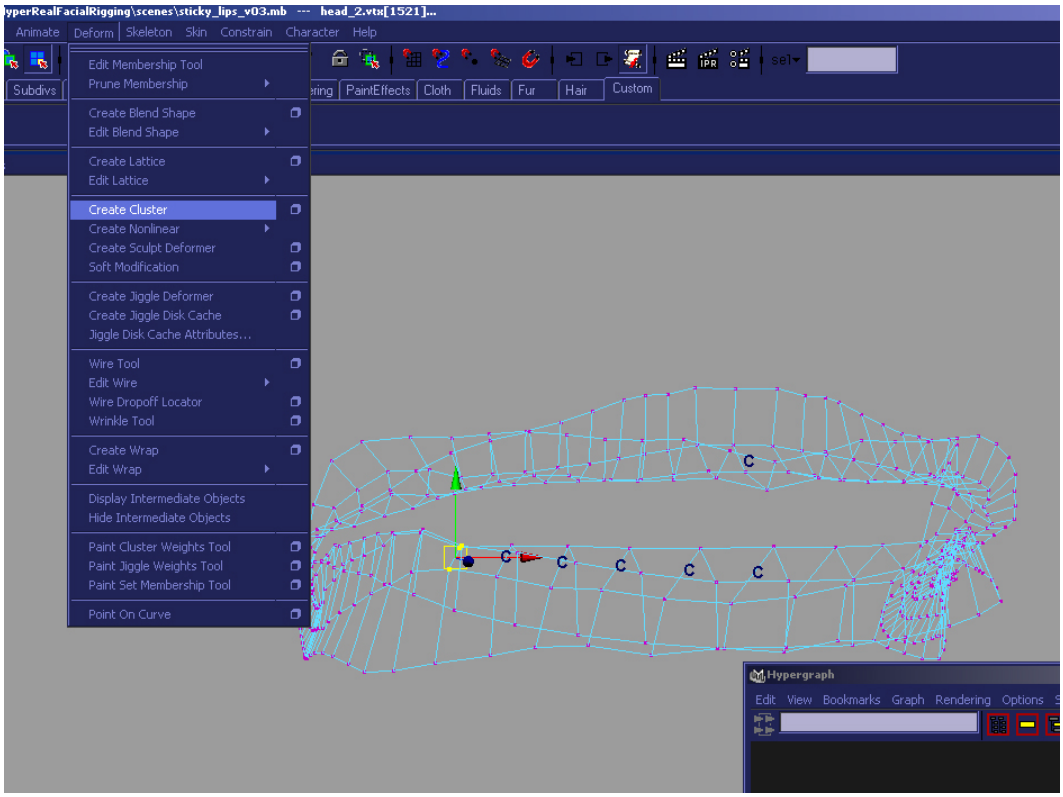
The selection of sticky lips vertices has been converted to faces for simple visual organization while applying the clusters in the next few steps.

7. With the sticky lip vertices converted to a face selection, go to the Show menu under the window view port and activate the following two options:
   Show->Isolate Select->View Selection
   and
   Show->Isolate Select->Auto Load New Objects



With the lips area isolated, it is much easier to work on this single set of vertices in the mouth, without actually deleting faces or changing the vertex order of the model. When you are done working you can simply deactivate this view mode.

8. Now, make clusters on each row of vertices of the model that you want to stay in-line with the mouth as shown and explained in the following image and caption.



Make a cluster on each set of top and bottom vertices, be sure to include the center three vertices of the lip, but not the outer vertices – since those are the edges that the rivets are attached to and you don't want to be deforming those verts (otherwise a dg cycle could occur).

*Remember, if you made a blend shape from the model with the jaw in it, in order to temporarily open the mouth for selection purposes, be sure to completely close the jaw on the deforming model, and then open up the blend shape editor and delete that temporary blend shape prior to the next set of steps.

9. Next parent each cluster to it's closest rivet locator transform (it's ok to do this by eye, as long as you are close). The reason for parenting the clusters is so that they will now move around with the center lip line of the model. This means that you will now have a deforming portion of the lips which stays nice and centered on the lips no matter what the lip shape may be.

Note: You should get the following warning when you parent the clusters, which is a desired safe guard that Maya takes for the user (if you don't get this warning in your script editor when you parent the lip cluster, then something definitely went wrong, go check your preferences and make sure the clusters don't actually move the vertices when you group them):

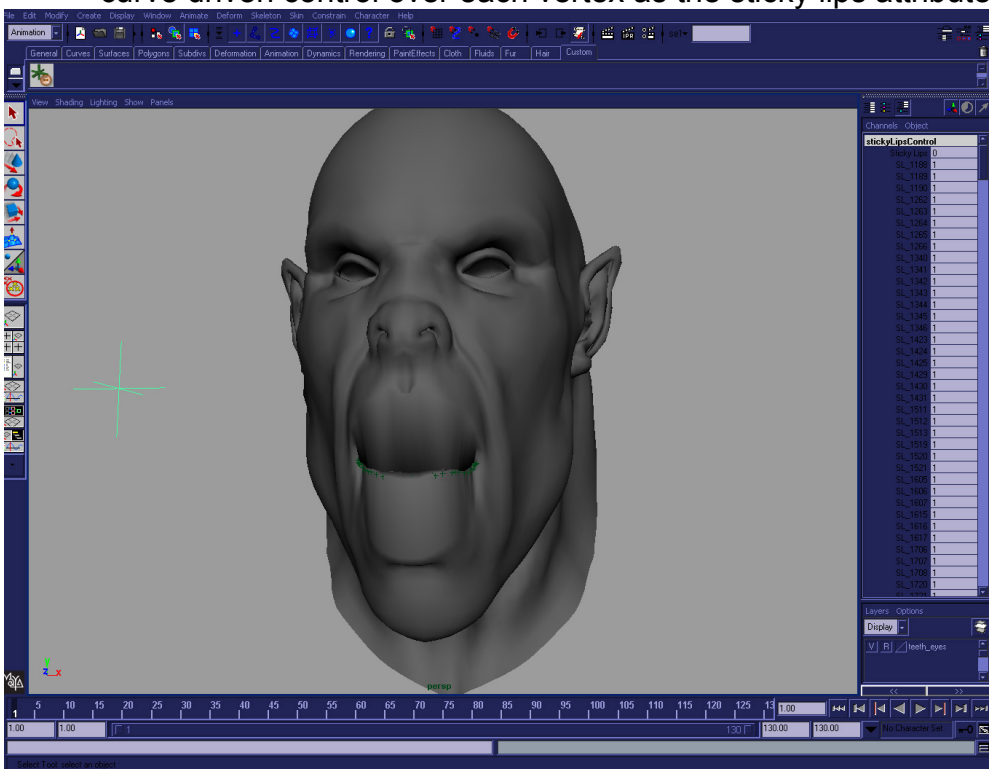
// Warning: clusters were grouped to preserve position //

10. Next, make a locator called "stickyLipsControl" and add a float attribute to it called “sticky_lips”. This eventually will be your final control attr.

11. Now, get a third base shape head (ie: another original un-deformed model), name it “output_head”, and blend shape the character’s deforming jaw head model into it. Set this new blend shape attribute to 1 so that as you rotate the jaw, this new model deforms as though it is bound to the jaw.

The reason for applying the joint deformed model as a blend shape onto a new base shape model is solely to avoid a bad dependency graph cycle and nasty cycle check warnings.

12. Next, we are going to do something that may seem a bit rash and unconventional – we are going to create a single world space mode blend shape node (and NOT front-of-chain) from our sticky lip model onto our new output head model, for each of the sticky lip vertices that we had in our original selection set. We are doing this because we want explicit curve driven control over each vertex as the sticky lips attribute changes.



When the per vertex world space blendshapes are all turned on, and the jaw is opened up, then this is what you get.. the beginnings of some pretty obviously sticky lips.

13. Then, we will create a list of consolidated attributes on a single locator controller node that is directly connected to each one of the blend shape's weight attributes in order to do a per-vertex set driven key using the "sticky_lips" attribute as the driver.

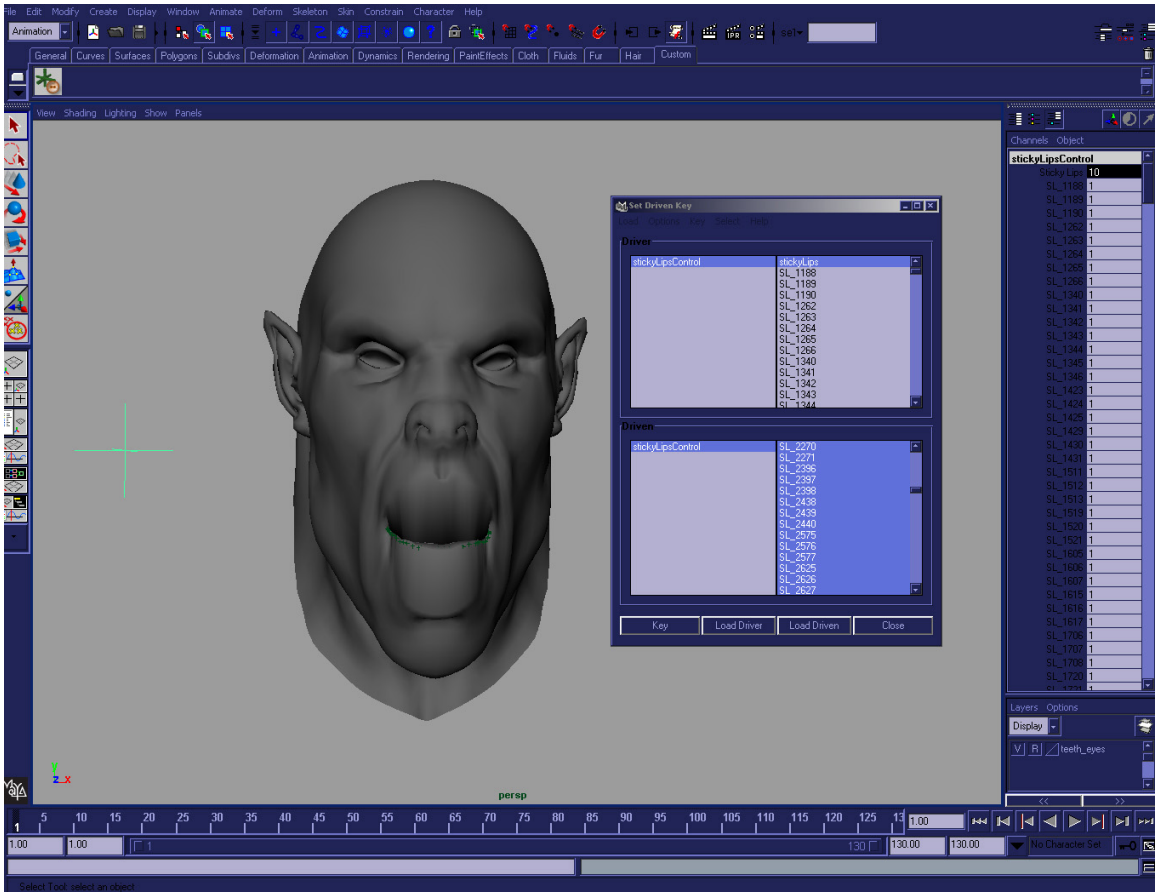
"Whoa?", are you thinking? Doesn't this many blend shapes and this many attributes controlling them all seem a bit insane? Well, that's because it is!!! But, seriously, don't' you worry! Doing this may seem overly complicated at first glance, but it is actually very simply when you automate the process with a really quick and easy few lines of MEL scripting; and luckily enough, this will all be automated with MEL, so it will only take a couple seconds (as opposed to maybe spending all day trying to get it right).

14. Select the quick select set of verts to be sticky, and run this code (be sure your node names match the ones declared here in the first three variables):

```
string $sourceModel = "sticky_lips_head";
string $targetModel = "output_head";
string $controller = "stickyLipsControl";

string $selected[] = `ls -sl -fl`;
$selected = sort( $selected );
for ( $vert in $selected ){
        string $targetVert = `substitute $sourceModel $vert $targetModel`;
        string $attrName = `substitute "^.*\\[" $vert ""`;
        $attrName = "SL_"+`substitute  "\\]$" $attrName ""` ;
        addAttr -ln $attrName -at double  -min 0 -max 1 -dv 0 $controller;
        setAttr -e -keyable true ($controller+"."+$attrName);
        string $blendShapeNode[];
        select -r $vert $targetVert ;
        $blendShapeNode =
                `blendShape -tc 0 -o world -name ($attrName +"_BS")`;
        connectAttr ($controller+"."+$attrName) ($blendShapeNode[0]+".w[0]") ;
}
```

**Explanation of the last few steps: Its actually quite necessary to do it this way, because doing this gives us exactly the type of driven per vertex shape control that we will need to shape the sticky lips exactly as we want when the sticky_lip attribute driver changes... the only other way this would even be possible is if Maya's blendshape deformer supported paintable and keyframable weights (which unfortunately it currently doesn't [Maya version 6.0]).*
Also note that doing things this way will allow us to have sticky lips at all times, and in all cases, since we are operating on the sticky lips at the end of the deformation chain, and on the points in world space - so no matter what random

deformer may be operating on the face as an input (ie: joints, clusters, blendshapes, sculpts, etc) the sticky lip controller will always stick them right together properly - which is a main goal of properly rigged and controllable sticky lips.



15. Load up the set driven key window, and load the “sticky_lips” attribute that you created as the driver, and load all of the per-vertex “SL_###” attributes that the script put on your locator controller as the driven. The numbers directly correspond to the mesh’s vertex number (you will need to use that knowledge later, for sure).

16. Now, set the sticky lips attribute to 10 (note the sticky_lips attribute should go between 0 and 10), set each one of the “SL_###” attributes to 1, and set a driven key frame.

17. Next, do the opposite, where you set the sticky lips to 0, and set the SL attributes to 0, and do another set driven key frame.

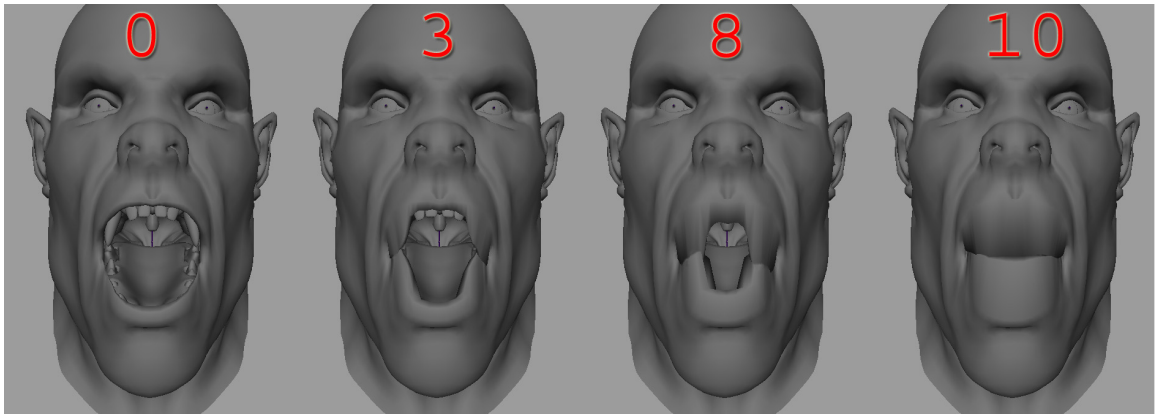
18. Next, it starts to get pretty cool (but also somewhat unfortunately tedious). Now you will begin to actually shape the sticky lips as they begin to “peel” apart as the mouth opens. They should peel apart in a very “oh” like

shape, starting at the center, and gradually un-sticking following through to the edges at the very end.

19. So, this is the very reason that we used set driven key – so that we could very particularly model the look and feel of the sticky lips on a per-vertex level as each one falls in and out of stickiness.

20. Set the sticky_lips attribute to 3, and now begin to tweak each one of the attribute values on the locator, which will in turn move the vertex in and out of being sticky. Hint: Change multiple attributes at once, and then go back and tweak individual vertices as needed.

21. At a value of 3, you should have most of the center vertices barely sticky at all, and have a lot of the vertices near the corner of the lips still stuck together. Perform a set driven key when you are happy with the lip shape at a value of 3.

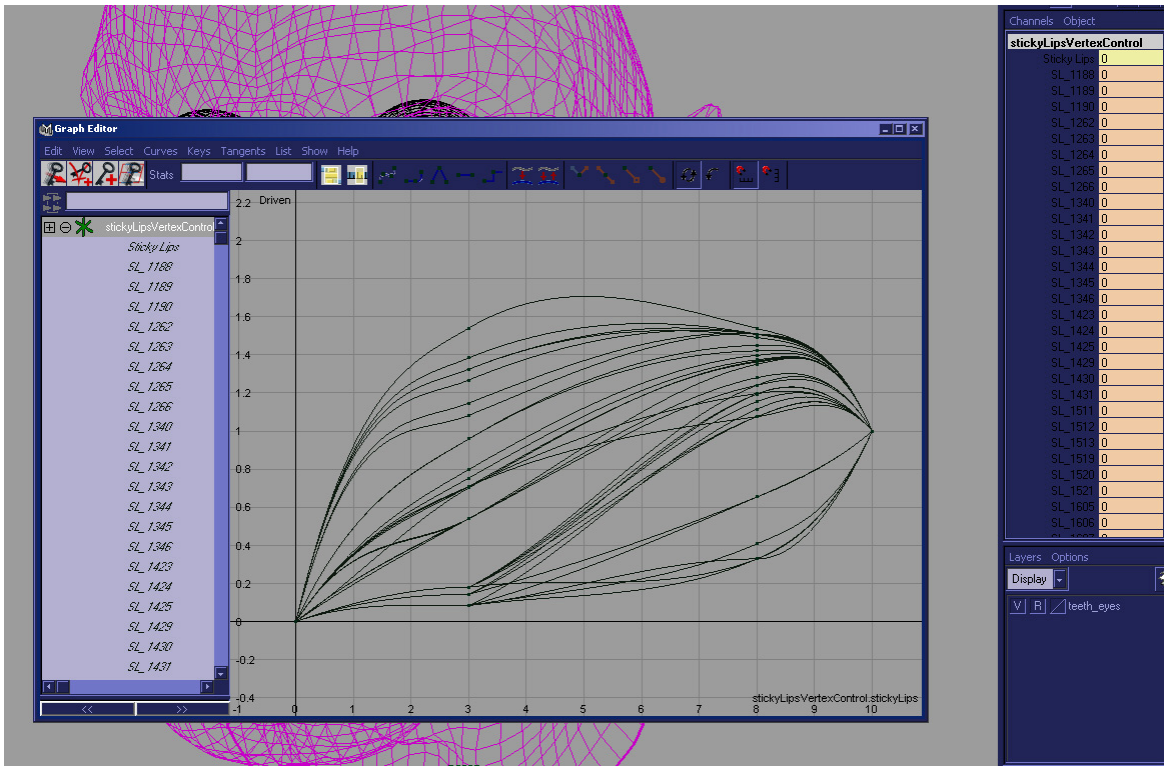


.

22. Next set the sticky_lips attribute to a value of 8, and again, tweak each one of the attributes until you have a shape that is mostly stuck together at the edges all the way until the center, where there should be a subtle “oh” shaped area in the middle that is still not really very stuck together.

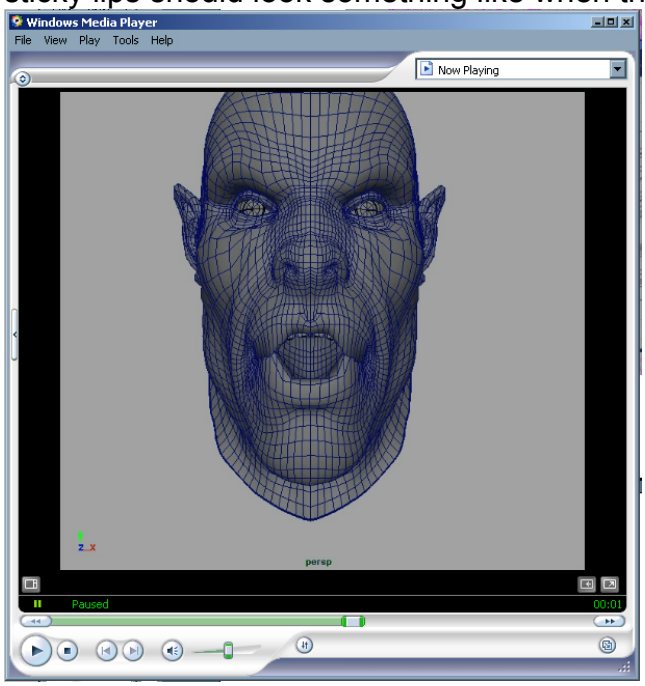
23. Now, feel free to add as many set driven key frames as you would like – I suggest starting out with as few as possible, and then adding more to really define the mouth shape as the slider is being manipulated.

24. Finally, tweak the actual animation curves and key frame locations and tangent values in the graph editor to get the sort of look that you are happy with as you change the sticky_lips attribute slider to drive this effect. The following image and caption shows what my tweaked graph editor curves looked like – although they could look much better for sure, if more time was just spent tweaking.



Here is what the final per-vertex set driven key frames looked like for this example. Note that more time can be taken to really tweak the look of the sticky lips, and do more in-between key frames to really get the look perfect. What is shown here suffices, though, as a good example of heading in the right direction.

Be sure to check out the following sample file for a simple example of what the sticky lips should look something like when they are animated, in –action.



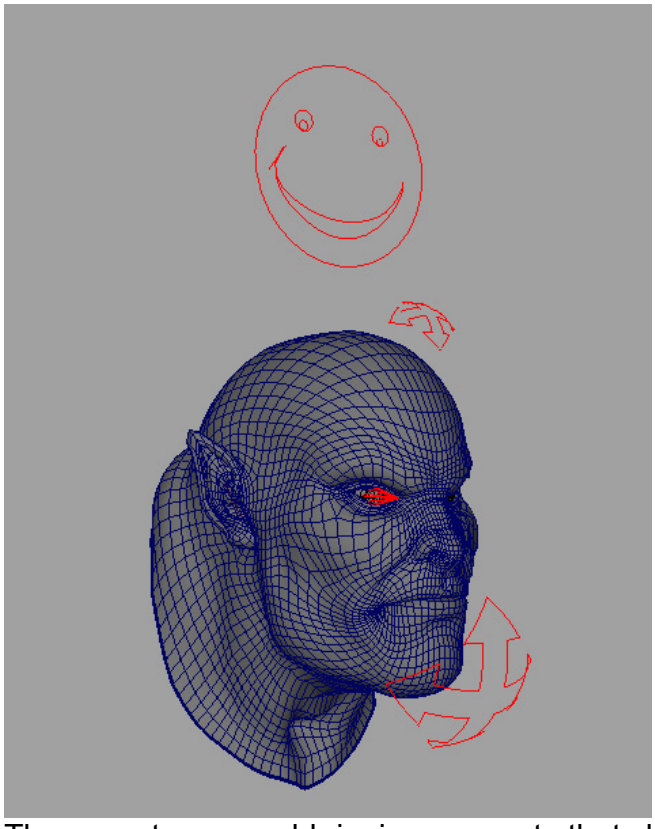
Check out the movie "sticky_lips.avi" to see a simple example of sticky lips…

Combining the separate rigs into a facial setup file

Once all the separate rigs are built, the process of combining them into the character setup file is quite simple.

All that needs to be done is to import each one of the separate rig files into the rig with the jaw joint in it. Most likely this file will already have the cluster on meshes rigged into it as well as the sticky lips. So, we will need to import the blend shape models, the blinks, the fleshy eyes, the rolling lips, and any other facial rigging component that you may have completed in a separate file.

Next, select all of the models that are the outputs of the deformations of each one of the separate rigging components, and do a blend shape onto the facial model which has the skinCluster node which deforms the jaw to open the mouth. Be sure to do a front of chain blend shape. Next, create and connect any controller nodes, and hook up your attributes so they are easily exposed to the user.



There are two age-old rigging concepts that should be used and remembered when doing this:

1. Iconic representation for character controls – no one should need to dig for a control, they should be obvious and stick out to the user.
2. Attribute consolidation & locking and hiding non-keyable attributes so that nothing can be animated that isn't meant to be.

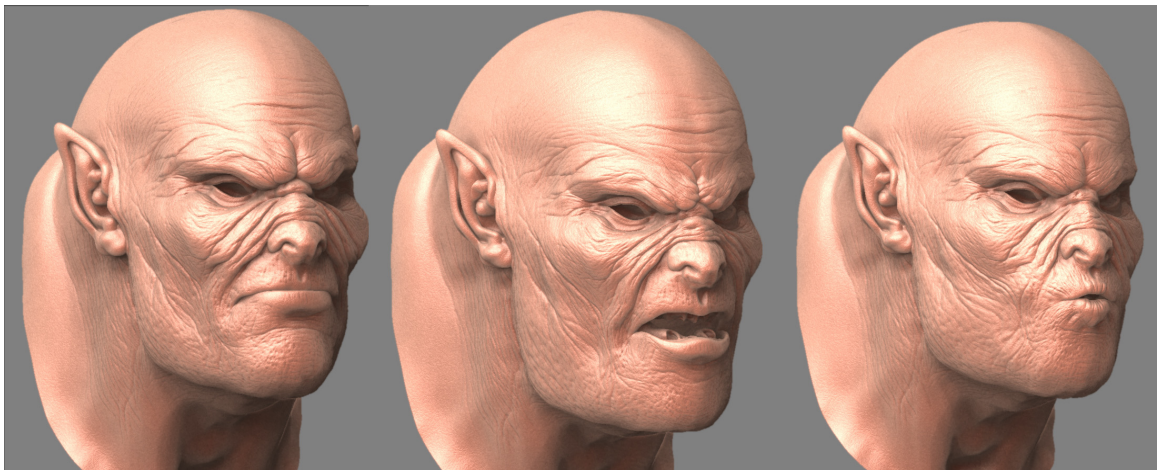
There are a few special connections that I ended up making when connecting things up – one was the fleshy eyes, obviously I had to orient constrain the real eyes to control the eye joints in the fleshy eye rig. Next, I did a set driven key on the .envelope value of the fleshy eyes blendshape so that when the blink attribute is equal to one, the envelope value is equal to zero, and when the blink is zero, the envelope value is one. The envelope value of the blend shape basically turns the entire blendshape off, so that as the eye blinks it doesn't get double transforms applied to it from both the blink and the fleshy eye setup. This technique can be used in many places with multiply divide nodes or expressions to achieve this sort of behavior in several parts of a facial rig, but this is an easy and obvious one.

The controls are made for both the upper and lower jaws, as well as the right eye (since I haven't rigged up the left eye as it wasn't necessary for the communication of techniques in this course). The rest of the attribute connections were pretty straight forward, and don't really need a detailed explanation – they can probably be achieved either with a direct connection or a set driven key if necessary.

Driven Displacement Maps (w/ mentalray) for Fine Skin Wrinkles

This next topic is one of considerable interest due to the super high level of realism that it can allow you to attain by tying your facial deformation rig intertwined with and actually driving large parts of the shading network. This is also (relatively speaking) a fairly new topic (or at least partially un-discovered) which by all means deserves great attention due to the fact that in reality our skin's inherent shading parameters (such as displacement and bump; even color and secularity) do actually change their appearance as our skin deforms around the bone causing the stretching and wrinkling of itself over blood filled vessels and striated muscle tissue – it is high time to admit that this phenomenon is not something that is a disconnected or unrelated issue, and is something that indeed should be synthesized in a character as much as possible to achieve the desired look.

It is finally possible to realize this level of powerful behavior in Maya, with built in utility nodes, (even custom mentalray shaders linked into custom Maya plug-in nodes if you are so inclined) and output a multi pass render to an advanced photo real film quality ray traced rendering engine such as mentalray.



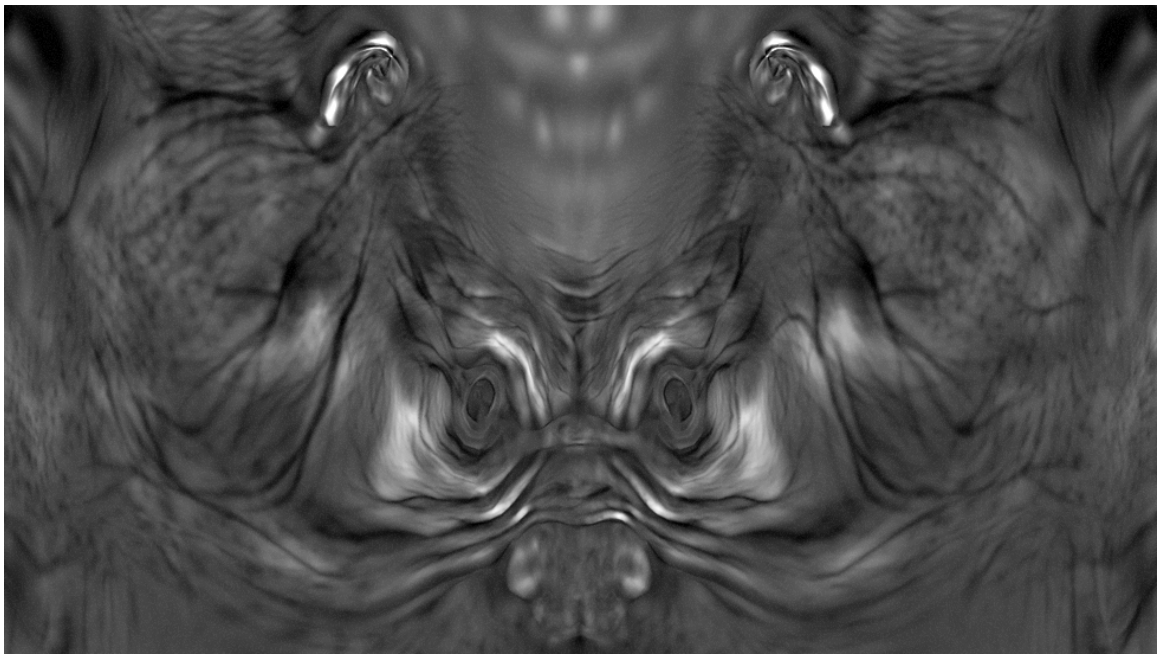
The heightened level of realism possible with custom displacement maps for each blend shape can easily be seen in this image. These fine and detailed forms shown here were sculpted like clay from the original models using displacement maps by a colleague, and extremely multi-talented artist, ***Nick Lloyd*** *(which he supplied to us exclusively for this class and DVD).*

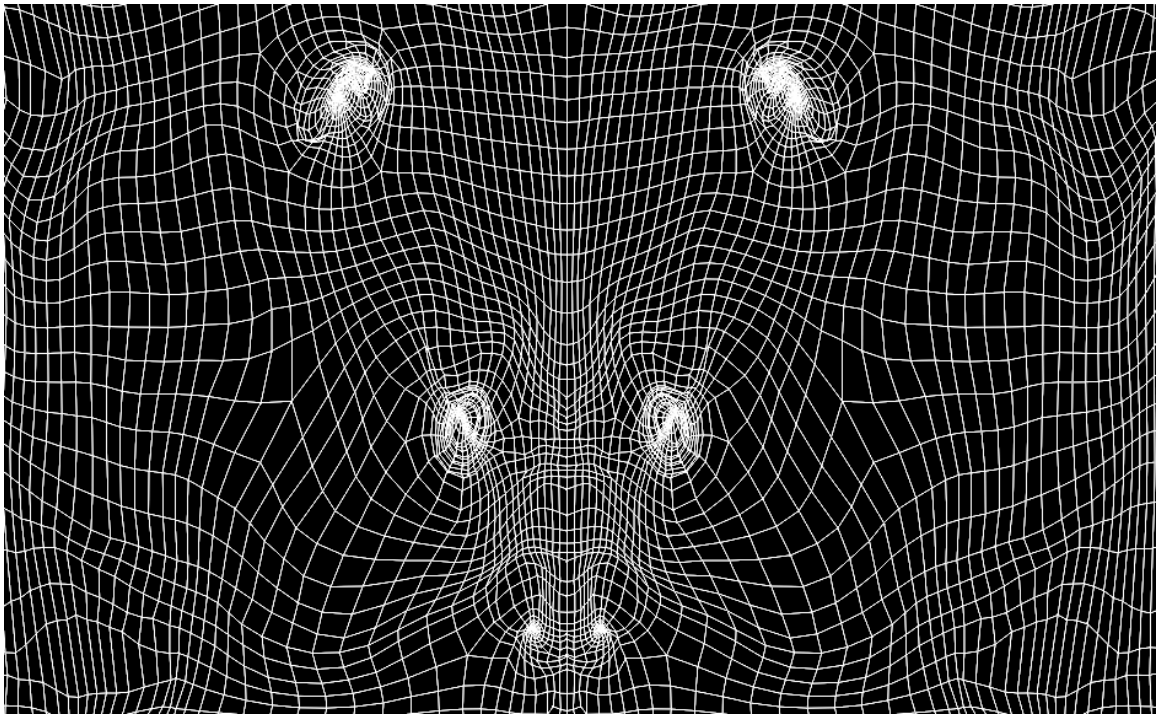
The idea here is that the model is never meat to go to such high resolutions in it's base mesh that could reach the level of detail needed for fine skin wrinkles as seen in the examples above. The pock marks in the face, skin pores and wrinkled age lines in the skin that radiate down are absolutely beautiful examples of the types of additional elements that will guarantee the appearance of added realism to your rigged up model when it renders.

The problem is that these kinds of details really do change considerably as the face hits different expressions. It is quite a catch 22 often times when you are attempting to make something look real as possible, and as it is animating the different elements begin to fight each other. We will see in a moment a way that as each shape is animated, the same model can have an associated displacement map which we blend in using a simple shading network in Maya which emulates the blend shape algorithm, the simple blend shape expression which we discussed earlier in this course will be built in the hyper shade using Maya's shading nodes.

I will briefly mention that the UVs of the model are of course important, and that they need to be evenly laid out on the geometry so that the texture space is grid-like for painting on. Even if your pipeline is using an advanced projection paint texture system such as Deep Paint or Z Brush, you should still be sure that the model has really nice UVs. The resulting maps will *always* look better if the UVs start out perfect, and the fact is that the maps can only be as good as the 2D UV texture space that they have been painted into.

Here is what the displacement maps (cropped region) for one of the blend shapes looked like, along with a UV snapshot of relatively the same area. Take important note that even this far way down the pipeline that a properly modeled edge flow that now can be cleanly painted with appropriate maps for clean displacement tessellation at render time is of significant importance.





Now that we understand the necessity good UVs and the importance of having a skin shader that reacts appropriately and can give the appearance of fine skin wrinkles when it deforms, we are faced with the challenge of rigging that up. Luckily the process is made extremely easy using Maya's built in shading nodes. Similar to how deformers work automatically on every vertex in a 3d model in order to deform it, a shading node works on every sample in an image in order to process it –much like a compositing operation during render time.
Basically in order to make this whole process work exactly the way it should, your shading algorithm should echo exactly your deformation algorithm.

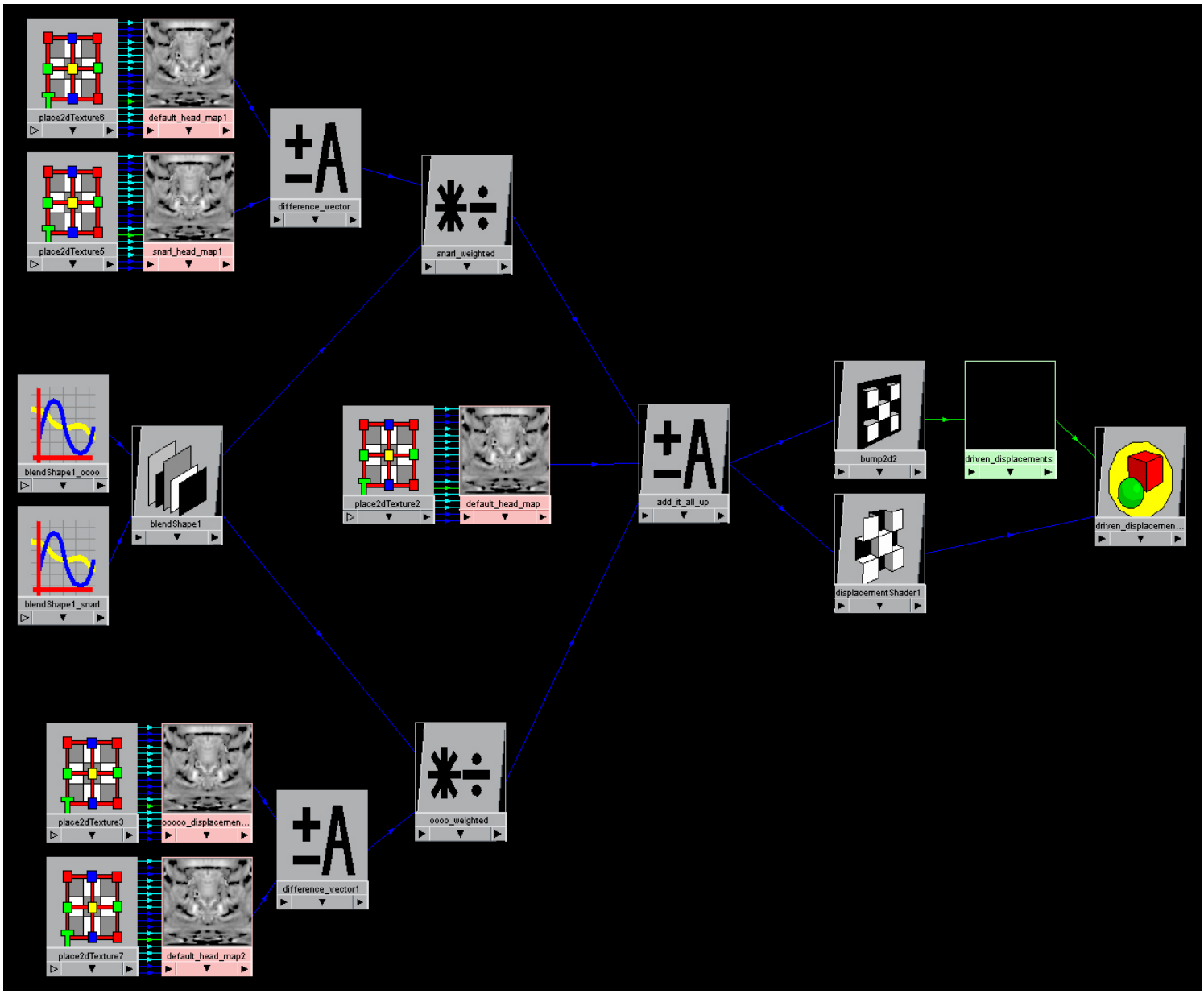
With this knowledge, we can go ahead and create a shading network that emulates exactly the behavior of blend shapes, and then use the multiplier that controls the blend shape attribute for a particular modeled shape to drive the multiplier that controls the calculated difference of a displacement map that was painted particularly for that very same-modeled shape.

So, knowing how blend shapes work *was* important after all. Did you skip over that section with the equation? I probably would have – but anyhow, if you did, go back and take a quick look over it one more time again.

The simplicity of the blend shape algorithm can be built directly into the shading tree as long as the original map is always started from as the base map for modifying the new maps that get created. This is a perfectly reasonable requirement, and is creatively straightforward since blend shapes already work that way. So, then all you have to do is subtract the original map from the newly painted map (making sure you subtract in the right order so you aren't getting incorrectly negative values), now you are left with the difference vector, just like a

blend shape has. Then you take the output of that subtract node, and multiply it by the actual blend shape value (the actual attribute on the deformer node). Next, all you need to do is take this output, and add it in with all the other difference outputs from any of the other blend shapes, as well as the base shape. The outputs of this single add node should then be connected to the displacement or bump map's input channels!

This node network is directly portrayed in the following hyper shade network. It is using a simple example with two blend shapes for the sake of clarity, but using this same technique, you can have an unlimited number of inputs coming in to that final add node that plugs into the displacement shader, and it would all work exactly as it should.



Be sure to check out the movie file “driven_displacements.mov” to see an animated example of this technique! Bear in mind that this looks pretty cool, but is a test with low quality render settings. If certain approximation editor settings in mentalray’s displacement subdivisions and ray tracing settings in render globals had been made higher there would be even more richly shaded detail in the geometry (since the maps were so absolutely well done [by Nick Lloyd]).

IV. EXPANDED TOPICS

Realistic *Skin* Shading Tricks (w/ mentalray)

So, all the buzz words are going to be used here (you are now either saying “hurray”, or “oh-jeez”).

At any rate, we’re using *mentalray*, we have *ambient occlusion*, we have *final gathering*, we have *global illumination* and in the end, we indeed have faked a relatively decent *sub-surface scattering* look (although there is no mathematical calculation that is explicitly calculating a subsurface algorithm, and there is no special subsurface shader being used…).

The trick being used to create the following image is extremely simple, and took very little time or effort using *mentalray* for Maya. I will attempt to explain how this look was achieved.

All of the techniques and shading models used to create the above image were standard, out-of-box mental ray rendering techniques, translated directly from Maya 6 into mentalray 3.3.1 -- *except for one shader*:

The ***ambient occlusion*** shader that I used does not come by default with mentalray, but was incredibly helpful, and also really easy to download and install for use in a Maya shading network.
It was written by **Daniel Rind**, a very talented Computer Graphics Software Engineer from Vienna/Austria. His mentalray shaders are available for download from http://www.animusartis.com/ and I would highly recommend checking them out! I really want to commend Daniel for such great work on writing really fast and stable open-source shader code, and distributing it to the cg community without charge. This is the sort of thing that is becoming less and less common these days, and I really want to give props to Daniel for amazing work, very well done indeed.

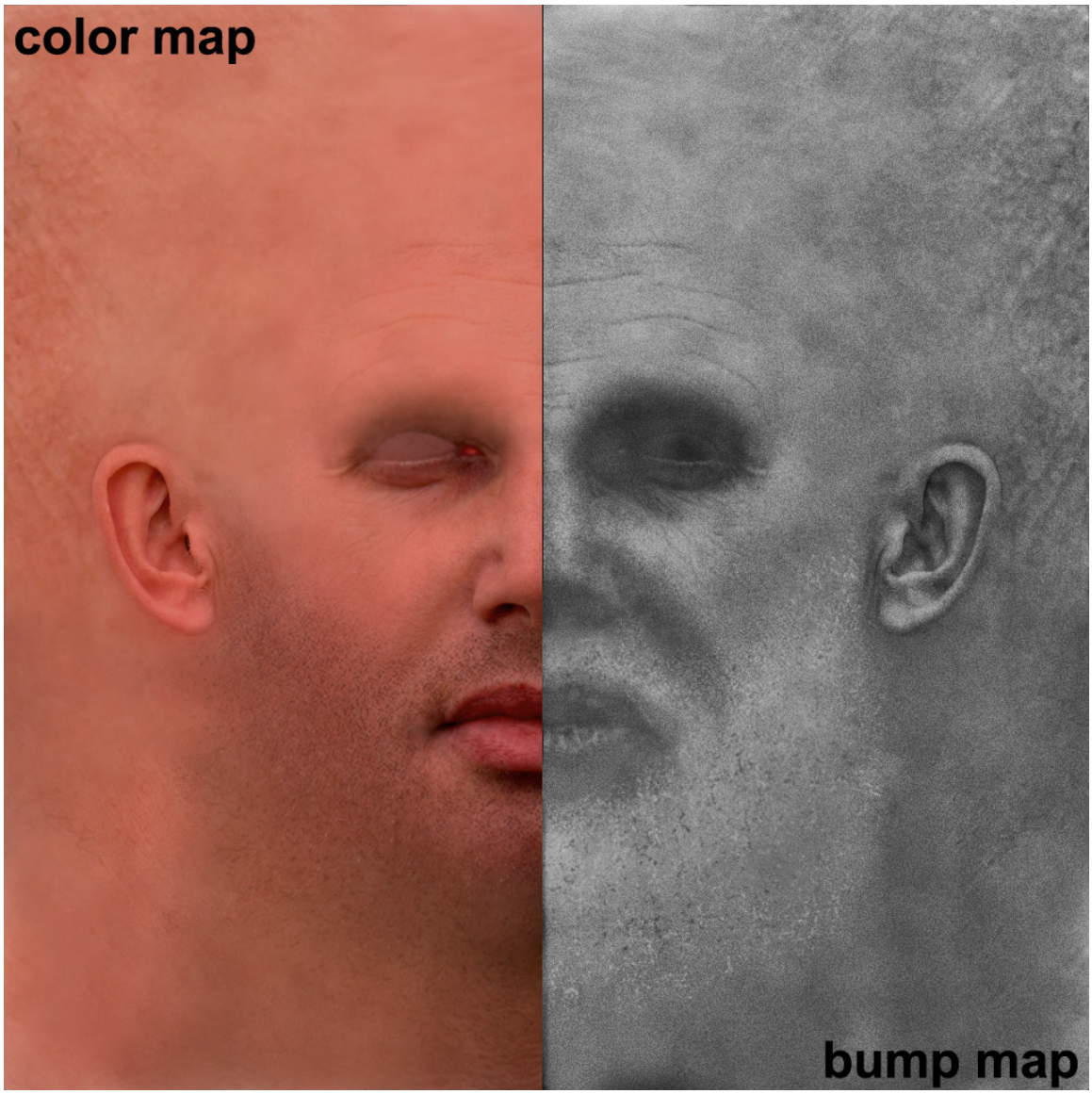
Sub-Surface Scattering Updates And News (Summer 2004):

- *Recently, Daniel Rind from http://www.animusartis.com/ released a really nice looking sub-surface scatter shader, which is also distributed with the source code. Unfortunately during the making of this class and the imagery included here, this sub-surface shader was not yet released, otherwise I definitely would have tried using it (to compare and contrast with the methods shown here), but as of now, I will leave that as an exercise for the viewer. I look forward to seeing the amazing images that I am sure people will begin to make with it.*
- *There is also a really nice looking subsurface scatter shader by Mark Davies, a really great TD and engineer that runs the popular website http://www.lighengine3d.com/ - unfortunately at the time of writing this course, I discovered the shader late in the game, and it was not available for mentalray 3.3.1. Chances are it is available now, and I would definitely check the website for details because it looked like a really nice shader.*
- *Subsurface algorithms are actually built in to versions of mental ray 3.3+ and there is currently a real, physically correct subsurface scattering shader that is being developed and beta tested by mental images. This is hot stuff, and most likely by the time you are reading it, this shader will be available from mental images.*

Textures…

Ok, so lets take a quick look at the textures first, so we can see two things –
1. They didn't have very much spec left in them, but still a bit more shadow.
2. Really they aren't that great of texture maps, because I painted them pretty much from scratch (w/ some good ref photos) so, I didn't have that much time to spend on them because of the class. ***But*** *the point is that they seem to be good enough to exemplify things, even though they really aren't that amazing.*

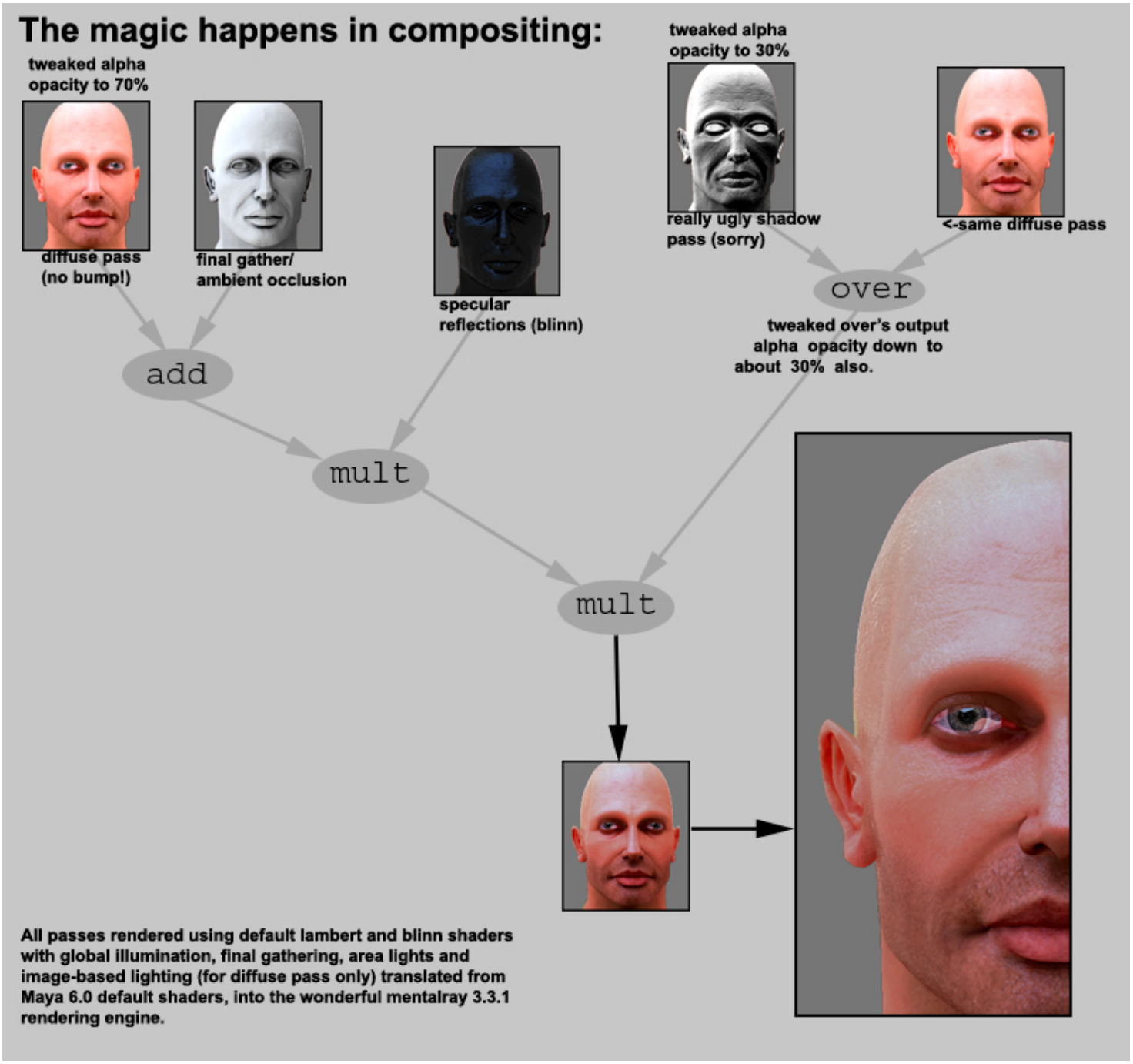
Careful consideration is taken to get a half way decent color map for the skin that has some nice natural hues in it. The maps themselves could definitely use more work, especially the bump map, but for the most part, they do their jobs for this example. The diffuse pass is then rendered using all the bells and whistles in mentalray – using texture mapped cards, spheres with images mapped, 4 area lights, and really high diffuse values on all the light bouncing object's shaders (the diffuse pass was a lambert with it's diffuse set to 1.0). The render took a little while, but for a 1024 x 768 render it probably took no longer than 15 minutes.



Here are the only 2 texture maps I painted, color and bump (and bump is really kind of shabby – I mean both maps could use tons of more work in paint if this were a real film).

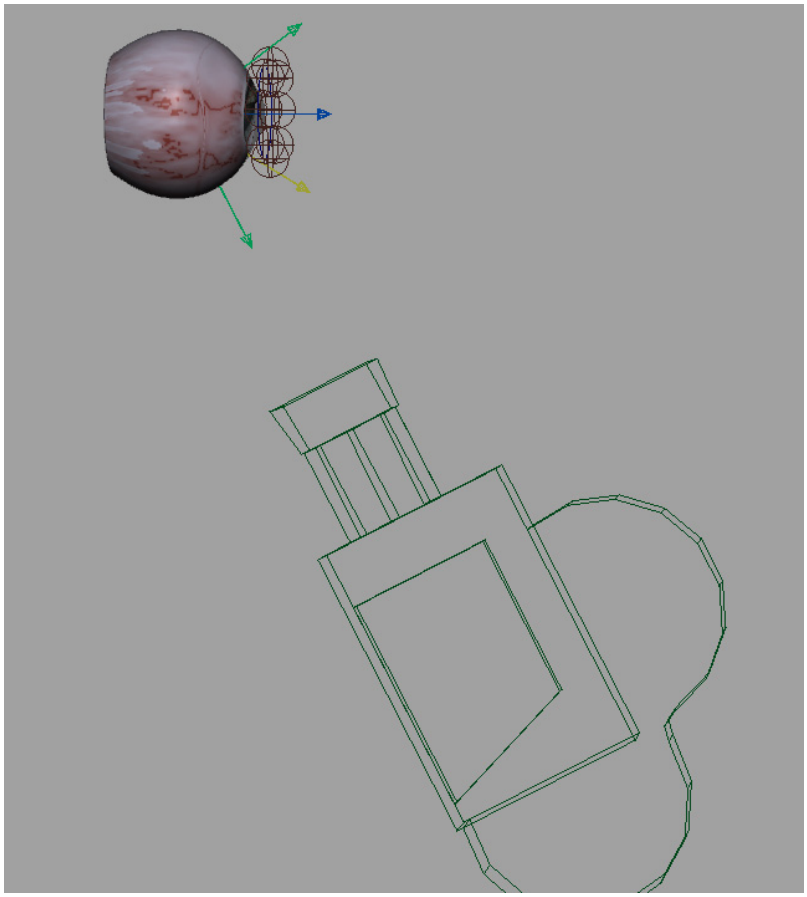
Composite…
There was no spec map, but light blue-ish green hues were used in the spec color channel. The specular channel was also "occluded" in comp using the ambient occlusion/final gathering pass to multiply it down. The key to this look really happens when you layer the diffuse color "bleed-ish" pass, which purposefully is rendered without any bump map, with the specular pass which obviously is rendered with high enough bump values as well as a nice looking sheen from image based specular reflections. The final gathering / AO pass is layered in such a way that it sort of works more like specular occlusion than shadow.

Here a mach-up of the comp tree. The actual shadow pass could use some real work, if there were some nice cast shadows across this stage left chin and upper lip the image would probably look a lot nicer… Also I would have to mention that the over-all placement of the lights as well as the bump and some of the color maps really could have been a higher quality had there been more time or people working on this section than just me under an after-hours style tight deadline.



Challenges of Realistic Eye/Iris Refraction

The difficulty of realistic eye refraction done using real bent rays and ray tracing in a shading model is not it's ability to look real- instead the problem is that it is near impossible to preview by the animator, who is the one setting the exact position of where the eyes will be looking. Instead, the possibility exists to pretty realistically fake eye refraction in the Maya scenes, using simple deformers, and some vector product nodes. This is possible since the eye is basically a spherical surface, and it is pretty easy to convincingly fake refraction through the tip of a round object such as the eye. Since the eyeball geometry is two separate surfaces, it works by simply using driven bland shapes and sculpt deformers, which are driven from the dot product and angle between the eye to camera vector and some simple weighted vectors that fall in-between. The mirror across the eye vector is also useful for deforming the geometry on the other side of the eye where it may have some minor intersection problems with the outer eye surface.



Here is a view of a faked refraction eyeball setup. Check out the movies ***fake_eye_refraction.avi*** *and* ***fake_eye_refraction_altView.avi*** *to understand how this effect is achieved.*

The idea behind doing this was based on a Siggraph presentation by the ever so clever *Kevin Smith*, from *Weta Digital* (at the time) which was about the tricks used for faking refraction on Gollum's eyes (yep, that's right), presented at the Pixar/Renderman User Group, as part of the Stupid RAT Tricks presentation. In the technique shown by Smith, this trick was done in the shading model (and, btw, it was much less of a hack than mine is). In some ways, the way this is done here is much simpler and less "reality based" – but if done using deformers in the geometry well, and there is enough surface resolution in the eye itself to support a nice curve, then it looks pretty darn believable, and the major important part is the fact that the animator can immediately see it in real-time, without any additional rendering or problems when it comes to posing the eyes and discrepancies between where the black part of the pupil will be once it is passed through a refraction shader.

V. CONCLUSION

Absolutely nothing in this class is a hard rule, or meant to be taken as mantra. In fact, not even everything in this class is necessarily the best way to do something, especially if the design of the character involved has radically different design challenges than the near-human creature that was presented here.

Fact is, though, this class is simply anatomically based 3D artistic theory and entry-to-mid-level software usage technique – none of which is proprietary or un-accessible to the common user of Maya as an off-the-shelf advanced software modeling, animation and rendering application. It just so happens that these techniques do work quite successfully, and also so happen to be based on many years of combined experience, and are time proven in major feature films across various digital production facilities. The reason is because it attempts to use generic, extendable, and technically conceptual ways to approach facial setup and deformation rigging to achieve a relatively modest, but usually fantastic result.

This being said, it should also be mentioned that there are literally *constant* advances and revolutions in technology, both on hardware and software fronts. This presentation makes a relatively modest attempt to approach it's methods with this in mind. Theory should always be more of a first priority, and initially is much more important than technical know how (especially the details of a software package… even Maya) when attempting to achieve a very particularly focused behaviorally based result. What's here today could be gone tomorrow, so don't bank too much on only detailed specificity. I believe that the genius Albert Einstein stated "Imagination is more important than knowledge", this is still one of my favorite concepts of all time, due to how true it really is; not only in

regards to CG, but also just to problem solving and conceptual absorbance in general.

As my final denouement and in true conclusion form, I would sincerely like to hope that this course was rewarding and opened some doors for the participants involved. Professional knowledge (w/ opinionated ideologies) is such an interesting commodity to disperse, and it is such a rewarding experience when you yourself have devoted so much personal energy and motivation into attaining this very same type of information. I really enjoyed creating these materials and continuing to work with such talented people such as my good friend Paul Thuriot, Jeff Unay, Nick Lloyd, and others that contributed their ideas or suggestions for improvement as well. It was once again an honor, gentlemen. I believe that it is their work, and not mine, which has made this class look visually pretty damn good. It is always rewarding to work with such inspirational individuals. I would especially like to mention, and thank Daniel Lamothe, Carmela Bourassa and Roark Andrade, as well as the entire crew from Alias's Maya team for being so cool, smart, innovative, helpful and especially kick-ass. ☺ You guys rock (and so does your software), so keep up the good work, and continue to let us re-define our reality as we know it today, as we discover it tomorrow!

Condensed Reference Video Appendix:
*The Real Human Face (w/ video footage on DVD)***:**

Note that this appendix has a corresponding video footage section contained as bonus footage with the DVD version of this Master Class. Good reference is a truly an in-valuable resource. I would highly suggest subscribing to http://www.3d.sk/ for really nice high quality and high resolution human facial and body photography, with truly wide ranges and hundreds of diverse angles and people photographed, organized by separate photography sessions.

Each of the following sections and items in each list has a corresponding high quality close-up video reference which attempts to explain in real detail the meaning and type of motion derived from each item listed. Enjoy.

Condensed List of Muscle Based Face Shapes:

1. Brow Down/Up (Furrow & Raise)
2. Forehead Brow Sideways (Inward Pull / Outward Stretch)
3. Eye Blinks (covered more in depth later)
4. Wide-Open Eyes
5. Tightly Closed/Squinted Eyes
6. Nose Inward, Outward and Downward Flares
7. Nose Sneer/Snarl
8. Squinted Upper Cheeks/Lower Eyes (Including Crows Feet)

9. Puffed Out and Sucked In Cheeks
10. Lips Up/Down (upward and downward facial shrugs)
11. Lip Corners Up/Down (pull of smile, push of frown)
12. Lip Corners Sideways (Inward Pull / Outward Stretch)
13. Vertically Flattened/Tightened Lips and Puffed out lips
14. Open Mouthed Smile
15. Oh-Ooo Mouth Shapes (several in-between targets required)
16. Squeezed CloseTightened lips (closed lip purse/pucker)
17. Upward Lip Sneer/Snarl & Downward Lip Sneer
18. Jaw Clentch / Forehead Clentch / Neck Clentch

Phonetic Ranges of the English Language:

Sample Sentence:
"The quick brown fox jumped over the lazy dog"

1. A =alpha
2. B =bravo
3. C =Charlie
4. D =delta
5. E =echo
6. F =foxtrot
7. G =golf
8. H =hotel
9. I =India
10. J =Juliet
11. K =kilo
12. L =lima
13. M =mike
14. N =November
15. O =Oscar
16. P =papa
17. Q =Quebec
18. R =Romeo
19. S =sierra
20. T =tango
21. U =uniform
22. V =victor
23. W =whiskey
24. X =x-ray
25. Y =Yankee
26. Z =Zulu

Condensed Emotional States:

Happy: Smile, Laughter, Unbelievable Joy
Sad: Crying, Grief, Pain, Frowning
Angry: Screaming, Snarling, Sneering, Annoyed
Scared: Petrified, Terror, Afraid, Worried
Surprised: shock and awe
Confused: unsure, misunderstood, un-intelligent
Disgusted: Repulsion, wretching, sneering, loathing

Included Data Files (on CdRom):

Maya Scenes:
- blendshaping.mb
- driven_displacement_maps.mb
- rolling_lips.mb
- cluster_on_mesh.mb
- sticky_lips.mb
- fleshy_eyes.mb
- eye_blink.mb
- jaw_weighting.mb
- facial_rig_combined.mb

Texture Files:
- displacement_map_defalut.tga
- displacement_map_oooo.tga
- displacement_map_snarl.tga

Image Files:
- ambient_occlusion_pass.tga
- diffuse_color_pass.tga
- photoshop_comp.psd
- shadow_pass.tga
- specular_pass.tga
- final_comped_w_blur.tga
- face_texture.tga
- face_texture_bump.tga

Movie Files:
- cluster_on_mesh_control.avi
- fake_eye_refraction.avi
- fake_eye_refraction_altView.avi
- fake_iris_dialate.avi
- jaw_range_of_motion.avi
- sticky_lips.avi
- eye_blink.avi

driven_displacements_RENDER.mov
driven_displacements_PLAYBLAST.mov
rolling_lips.avi
fleshy_eyes.avi
mixing_some_blendshapes.avi

Credits

Model & Sculpture:	**Jeff Unay**
Displacement Map / 3d Sculptural Detailing:	**Nick Lloyd**
Image Reference / Texture Resource:	**http://www.3d.sk**
Facial Rigging:	**Erick Miller**