Subdivision modeling tips

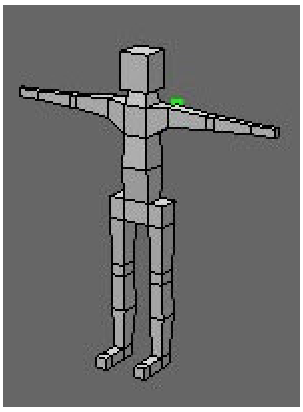
These methods and tips are based on WingedEdgeTech's Mirai, but will usually work with any modeling package that has a smoothing algorithm based on, or similar to Catmull-Clark subdivision surfaces. These softwares are 3D Studio Max 3.1 (NURMS meshsmooth) , Lightwave 5.5+ (MetaNURBS), Maya3 (smooth command; also supports real subdivision surfaces).

The subdivision modeling method:

In the beginning everything you build is a cube.

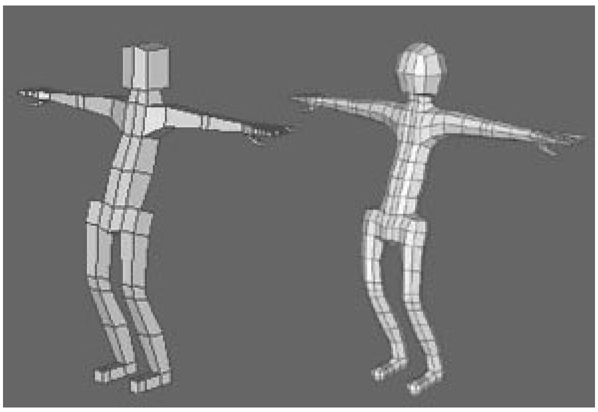
Do a combination of extruding faces, and scaling to block out the basic shape.

See box man:



After you have the basic "box volume" defined (head, arms, legs, feet and fingers) you should magnet move the points around to make a shape closer to something you like.

Also, it's good if you usually make one side a little more detailed than the other (only make a left hand).

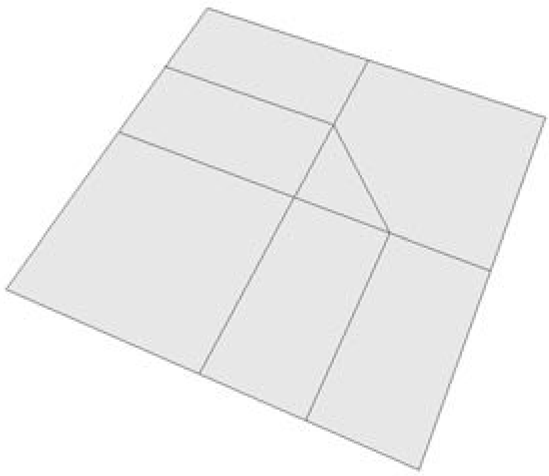


After you get the volume you want, do an interp smooth/meshsmooth/smooth to rapidly build up geometry from the box man into a more reasonable clay-like state, that you can then start cutting edge loops into.

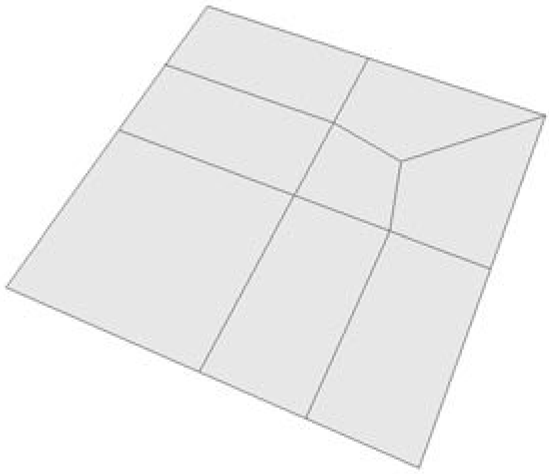
It is important to work with this box volume shape as much as possible because it has so few points. When you have a lot of points you can still work with it, but you will focus on the surface more than the volumes, and at the beginning the volumes are really what count.

N-sided faces and n-edged vertices:

While most modeling packages allow you to model with faces of virtually any size, if you intend to smooth your model, your results will be more consistent and predictable if you stick with quads. This is because bicubic smoothing works by averaging the tension between the corners of a face. In this case, a quad has a consistent “center” that you can clearly visualize by drawing an x connecting its opposite corners. Non-quads don’t have such an obvious, predictable and even distribution of tension. Unlike quads, there’s no guarantee a non-quad’s center of tension will coincide with its center of volume. And this, as you may have already experienced, will result in pinching when you smooth.
It's particularly important to watch out for triangles and have strategies for dealing with them. It’s very easy to introduce triangles when you’re building edge loops. A particularly common scenario involves crossing the corner of a quad:



This gives you a triangle and a pentagon. Bad. But easy to fix. Cut the inner face of the triangle, connect the new point to the far corner of the pentagon, then scale the point toward that corner:

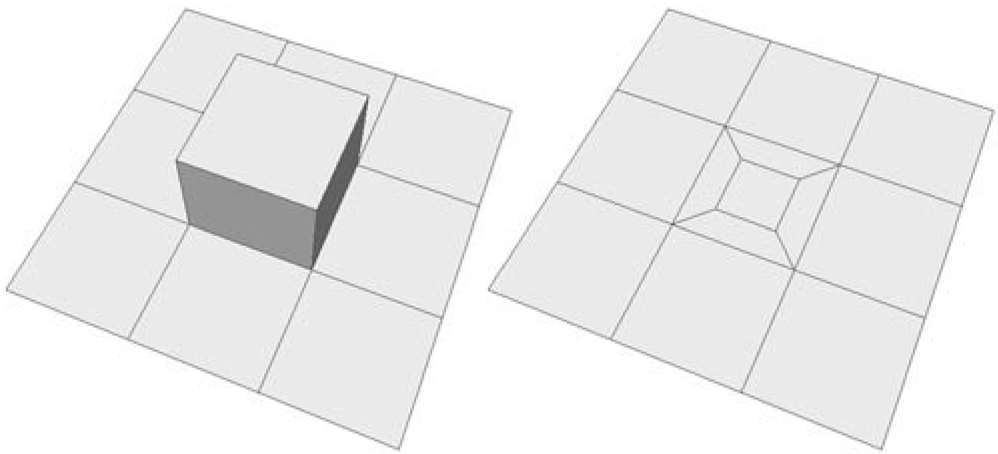


Now you have all quads.

You should make every effort to maintain an all-quad surface when you model. It is particularly important when modeling surfaces that have abrupt variations in detail or contain fine creases or bumps on an otherwise smooth surface. Of course, the human face is a surface that has these characteristics in spades.

The question arose as to how this technique relates to the theory that you should strive to only allow each vertex of
your model to have 4 edges. The answer is, it doesn’t relate at all. Modeling with all quads and modeling without “poles” (non-4-edged vertices) are very different practices. If you were to follow both philosophies simultaneously, your head would explode. Nevertheless, it is possible to try to use the best of both worlds.

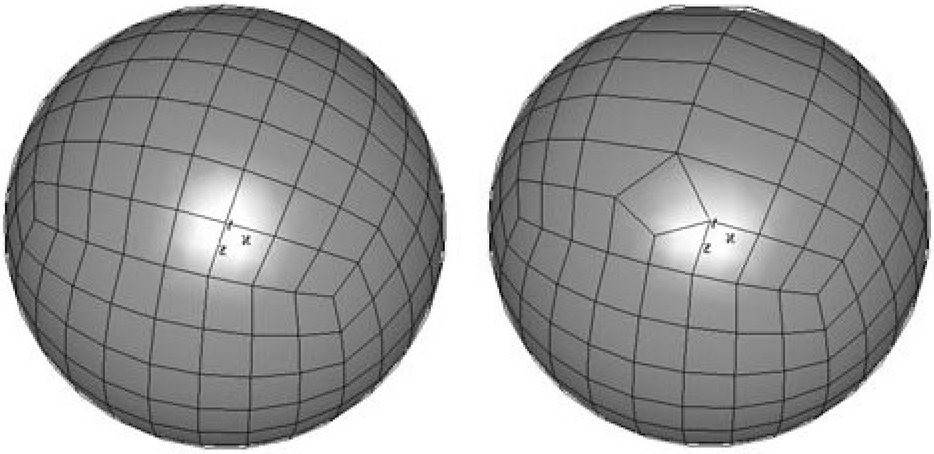
Modeling with all quads doesn’t guarantee an absence of poles and vice versa. In fact, quite the opposite is often
true. Perhaps the most basic example of this is the cube. It’s an all-quad surface, yet none of its vertices have 4 edges. Another example is what happens when you inset or extrude a quad face:



These actions maintain a quad surface, yet create poles.
On the other hand, the example given above of an edge loop crossing the corner of a quad avoids poles but creates non-quad faces. However, the experience has been that, when it comes to getting consistent, predictable smoothing results from organic models, poles are not an issue. Non-quad faces - triangles in particular -

are.

In fact, not only can you not avoid poles, there are techniques to use that rely on them. For example, here’s an edge
loop pattern you can use to terminate local detail, blending a high detail area with areas of lower detail:



As you can see, the “diamond face” used to merge three edge loops into one creates 3 and 5 edged vertices.
However, the derived surface remains completely uniform. This is because using poles allows you to terminate loops
abruptly while maintaining an all-quad surface.

It’s definitely worth mentioning that Bay Raitt, one of the most prodigiously gifted modelers in the world, is an advocate of the “no poles” modeling philosophy. As many of you may know, he’s been banging out world-class geometry since he was a fetus (or there abouts). In fact, it was watching him casually hammer out a fully articulated head in under an hour that got some people interested in 3d in the first place. So, it is to be emphasized that none of the above comments are meant to suggest that these methods constitute the best or only way to make good models. You can use only dodecahedrons in your models and if they end up looking like Bay’s work, noone is going to take issue with your methods.

That said, generating quad faced models is a very good approach to building the highest quality subdivided surface you can. You will need a denser mesh to do it, but the results will pay off if you can "afford" it at render point.

They are guidelines more than they are rules.

As a modeling technique you can still build with 4 sided vertices and N-sided faces, but then when adding detail the model becomes an "all quad" model. But only at that third 'detail' stage. When starting the volume or refining the form, its all 4 edged vertices. It keeps it pure. I can see the edges more clearly. It also makes you more aware of where the poles are going to need to be later.

When laying out the form initially its very good to focus on the "flow" of the edge loops that define your model. Its easiest to do this if you keep all of your points with only four edges coming out of them (this means that having N-sided faces is fine).

Once you've got a clean network of control loops that work well for your animation requirements, then go in and cut edges into your model so that you have as many quad shaped faces as needed. Three and five sided vertices are fine for this. More than that can get awkward in some systems, and it will also encourage bad habits.

Also bear in mind.
Six sided face ---> six sided vertex on subdivide.

So its a 2 step process:
Edge loop model first, then cut in to "quadrify" your model second.

Edge loops:

The short version:
Cut edges into your models as if you were drawing contour lines on your model.

The technique version:
Cut edges as you sculpt so that you can have the profile lines you want. Same goes for the surface shading. Keep the points so that they all have four edges coming out of them, except for places that don't move very much (pits of the eyes, nostrils, ear canals, etc).

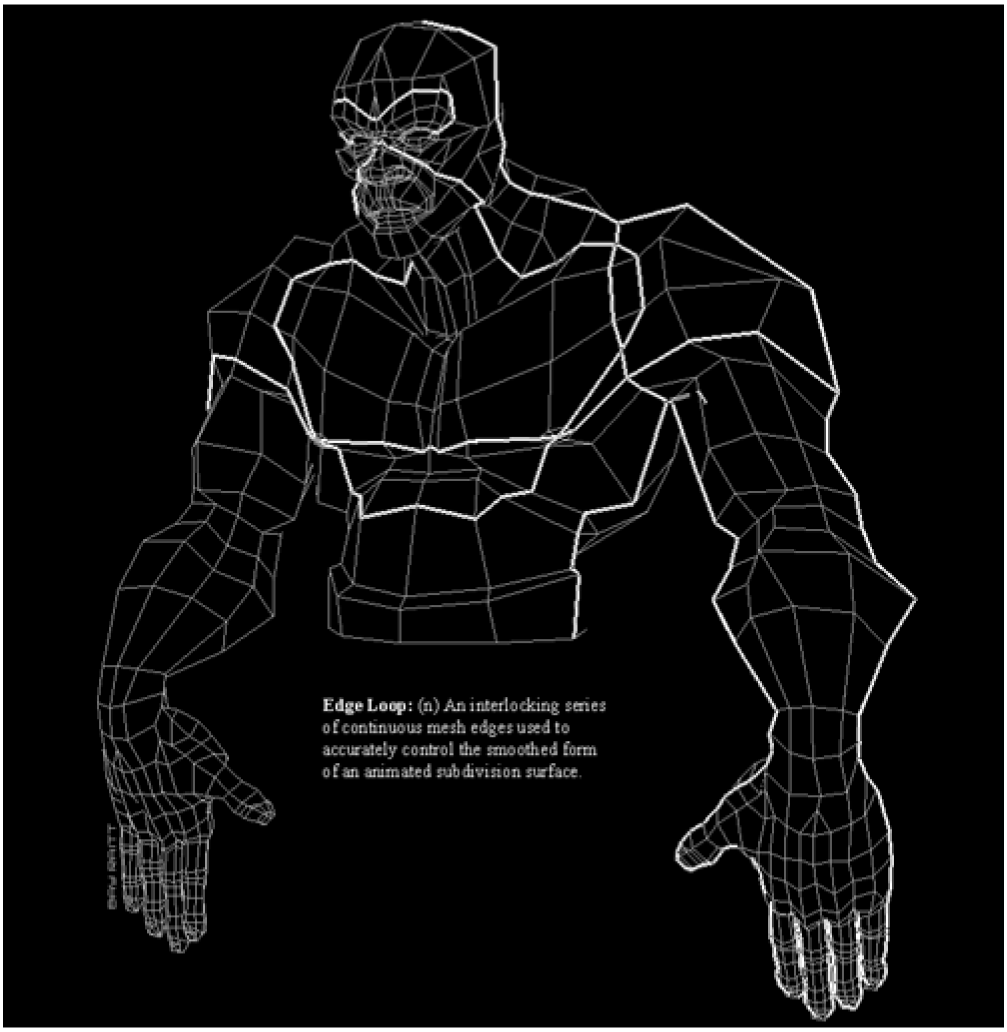
The kinetic version:
Rework the surface as you animate it so the the edges are arranged in a "ripple" like pattern arranged perpendicularly to the plane of movement (think eyebrow wrinkles, fat folds, major lines of action on a surface).

The subdiv version:
When you want a crease, lay two parallel edge strands so that they run very close to each other creating a crease. Then add two more edges on either sides to create the "pillowing" effect that will give your model substance. This way when it moves you will have accurate control of the surface form, control that you would have otherwise lost using an over simplified "non-uniform" weighting.

The texturing version:
Arrange the edges so that you have groups of faces that cleanly turn into separate parts so that you can cleanly texture the object.

The end version:
After you've done the above a few times, you'll start to notice a strange interconnected pattern emerges. I refer to this method as an 'edge loop'.

Enough talk. More pictures.


Edge Loop: (n) An interlocking series
of continuous mesh edges used to
accurately control the smoothed form
of an animated subdivision surface.