Back to the Root Page of The No-Joke Topology Guide For Serious 3D Modelers

(Strict) Quad Topology

The emergence of tendencies or requirements towards the strictly quad topology have emerged from the increasing ease quad topology can be derived. If you model without any triangles or N-gons to begin with, follow a standard pipeline (low-res, then high-res, then bake, then animate, and, optionally, export for game engine) and make no stupid mistakes, then you don't even have to worry about those ever bugging you. So, why not do this and stick to strictly quads, in view of a smoother, easier, and less error-prone workflow later?

With the increasing polygon capacities by both rendering machines and game engines, the need to optimize models to quote/unquote *absolute minimum* of triangles was there and gone, past away into the history of long-gone graphics hardware support underdevelopment.

Kevin Lanning and his teammates have proven that the use of some triangles in game models is not yet completely outdated, but you have to remember that Kevin's pipeline is not exactly what you would call linear. In fact, I would call it a complete reversal of conventional methods. Kevin uses both 3Ds Max and ZBrush to come up with crazy-detail, really high-density models. Note, however, that Kevin always stresses working on these high-density models with perfect quad topology. Also, Kevin employs the work-ethic called kit-bashing in Epic's terms. This involves re-using (with possible adjustments) parts of previously-made models. Then, using 3Ds Max's modeling tools, he makes the low-res, going off his high-res. Then he uses a tool (in its 3Ds Max plugin form) that's called the Polygon Cruncher to optimize the ZBrush high-res model to the point where it contains polygon numbers manageable by Max, but doesn't loose any visual detail. Kevin adjusts his low-res in Max to minimize the differences between the high' and the low'.

Kevin doesn't mind some triangles in his Max models, and he has little trouble with this, since he knows instinctively not to disrupt the motion of the model with them. He defines the UVs on one half of the low-res *plus* a one-segment-wide strip of polygons all the way arond the model on the other side. He deletes the rest, which would be a little less then a half. This way, the additional strip of polygons prevents any baking problems in the middle seam area. Then he uses the Max's projection modifier to bake normal maps and other textures and shader maps from the high-res to the low-res. This gives him perfect maps for a set of stackable UVs, which he employs when making the model symmetric again.*

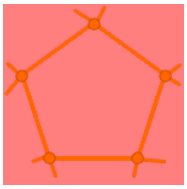
*Corrections to Kevin Lanning's workflow were referred by Stephen Minkin

However, had Kevin followed the standard pipeline, he would have had to create the low-res first-hand, UV it, export-import it from Max to ZBrush, then detail it, and then bake it. There are some advantages to this conventional process. Here they are:

1. If you bake in ZBrush, you don't have to worry about re-adjusting the low-res model or messing with the projection modifier: the high-res model fits over the low-res perfectly. The trick is that the low-res auto-adjusts in ZBrush as you're working in higher levels of subdivision, so the low-res is automatically an "optimized" version of the high-res, just no triangles in it. If the high-res doesn't fit over the originally intended low-res model that you took from 3ds Max or Maya, and you want that one specifically, there are simple ways to import the same old geometry again into the lowest subdivision level in ZBrush without losing any high-resolution detail in the higher subdivision levels.

2. If you use ZBrush normal mapping tools or the ZMapper, you do not have to wait for an hour for your 2048x2048 texture to render out. Pixologic (ZBrush author company) still shine with their highly optimized code.

The main catch here is that you cannot have any triangles and N-gons when you import to ZBrush, or you run the (about 95%) risk of getting the model, or at least part of it, totally messed up. Thus, you won't be able to save on the polycount in specific places, as you could if triangles were available.

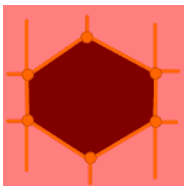


N-gons

This is a no-no, for many reasons, which include Catmul-Clark subdivision incompatibility (ZBrush) and unwanted stretching and/or bulging in animation.

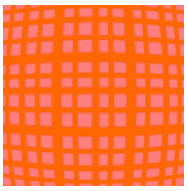
Triangles

NO. This is why this convention is called the (Strict) quad topology convention. ONLY QUADS, PEOPLE! That means, only 4-pointed, 4-edged faces! At least during the entire modeling, texturing, and animation process!



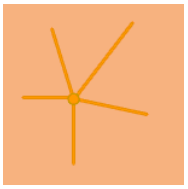
Holes

No. Quad-only also means no holes. A hole is not a quad, obviously.



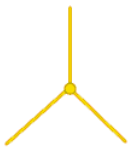
Dense / High Polycount Areas

I'm going to say no, once again. Same considerations as in the modern game modeling convention - there should not be a single polygon without a purpose. I will refrain from giving any specific numbers, since what is considered "dense" in a game model is not necessarily too dense for animation.



5-or-more-edged Vertexes

Vertexes that are surrounded by (define one end of) five edges are still fine, though this and the 3-edged vertexes are the only two things you have to watch your grace-lines* for this time. And again, 5 or more edges coming from one vertex pose, so to speak, a higher threat to your animation flow.



3-Edged Vertexes

These are also ok to use with this convention. As usual, watch your grace-lines.*

* Grace-lines are described in greater detail in the **Animation Topology** description, in the **Triangles** section.

Back to the Root Page of The No-Joke Topology Guide For Serious 3D Modelers