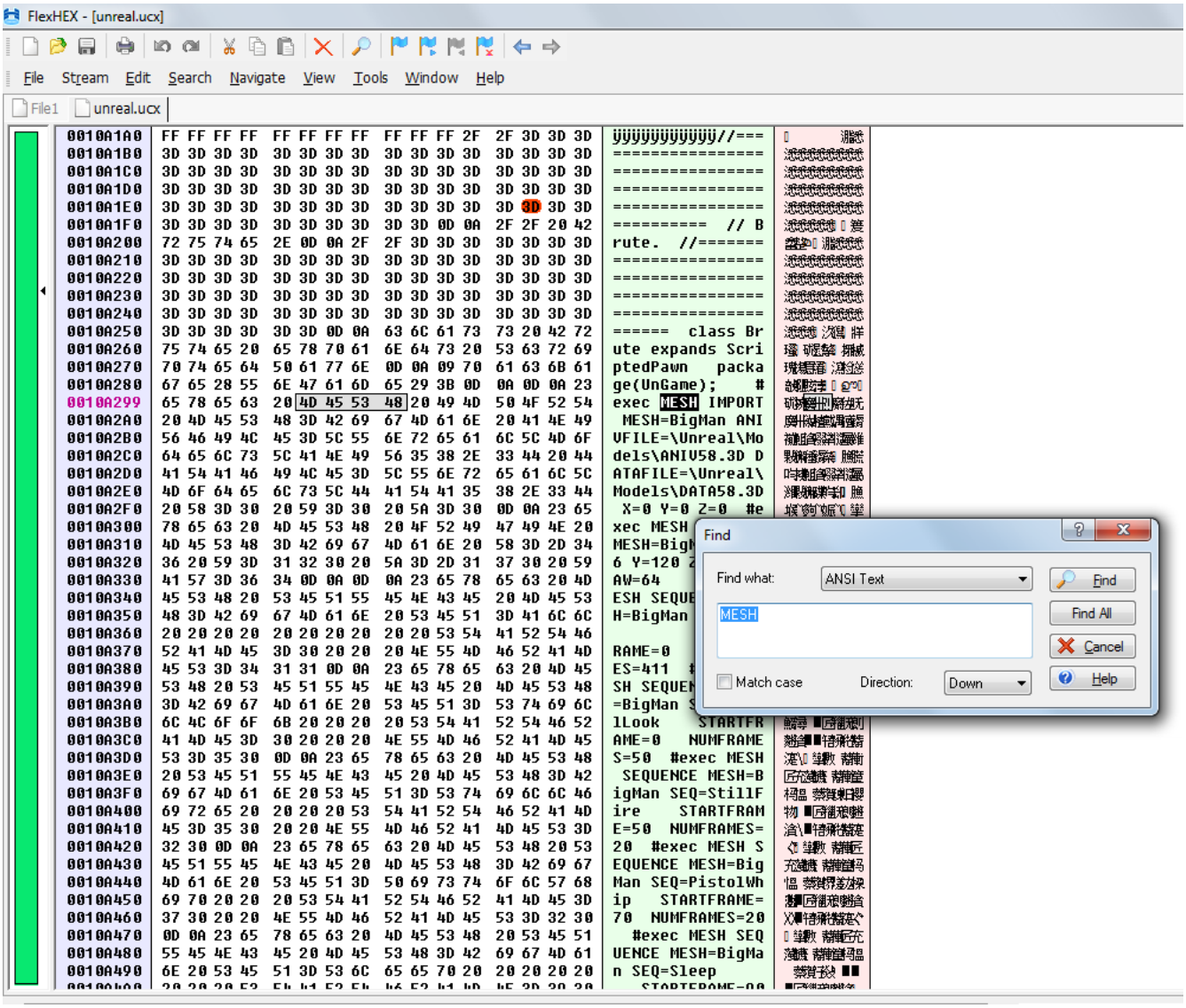
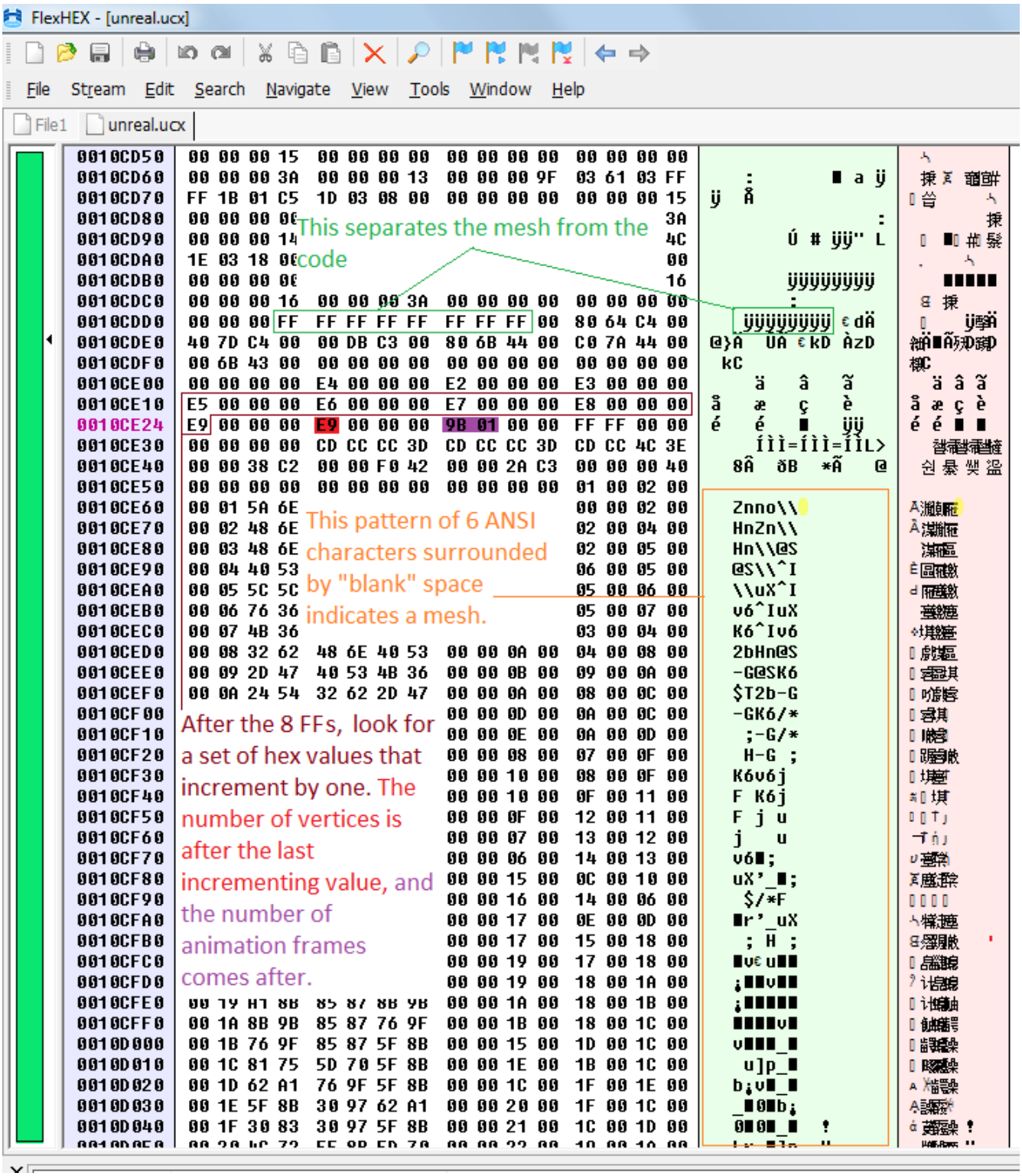
File exporters like UEViewer don't work with some versions of Unreal, including the beta and tech demos, which is why it might be useful to know how to extract vertex meshes through hex editing. The only tools needed are a hex editor and a calculator that support hexadecimal. Something like UnrealFX or UnrealNST will let you know if your extracted mesh will work in-game, but they aren't necessary. I'll try to keep this as non-technical as possible.

I use FlexHEX in this tutorial, but almost any free hex editor will do the job. First, open an Unreal archive; I decided to open Unreal.ucx from the '97 tech demo. This archive has the import sequences for all of its meshes, so try to find those first.

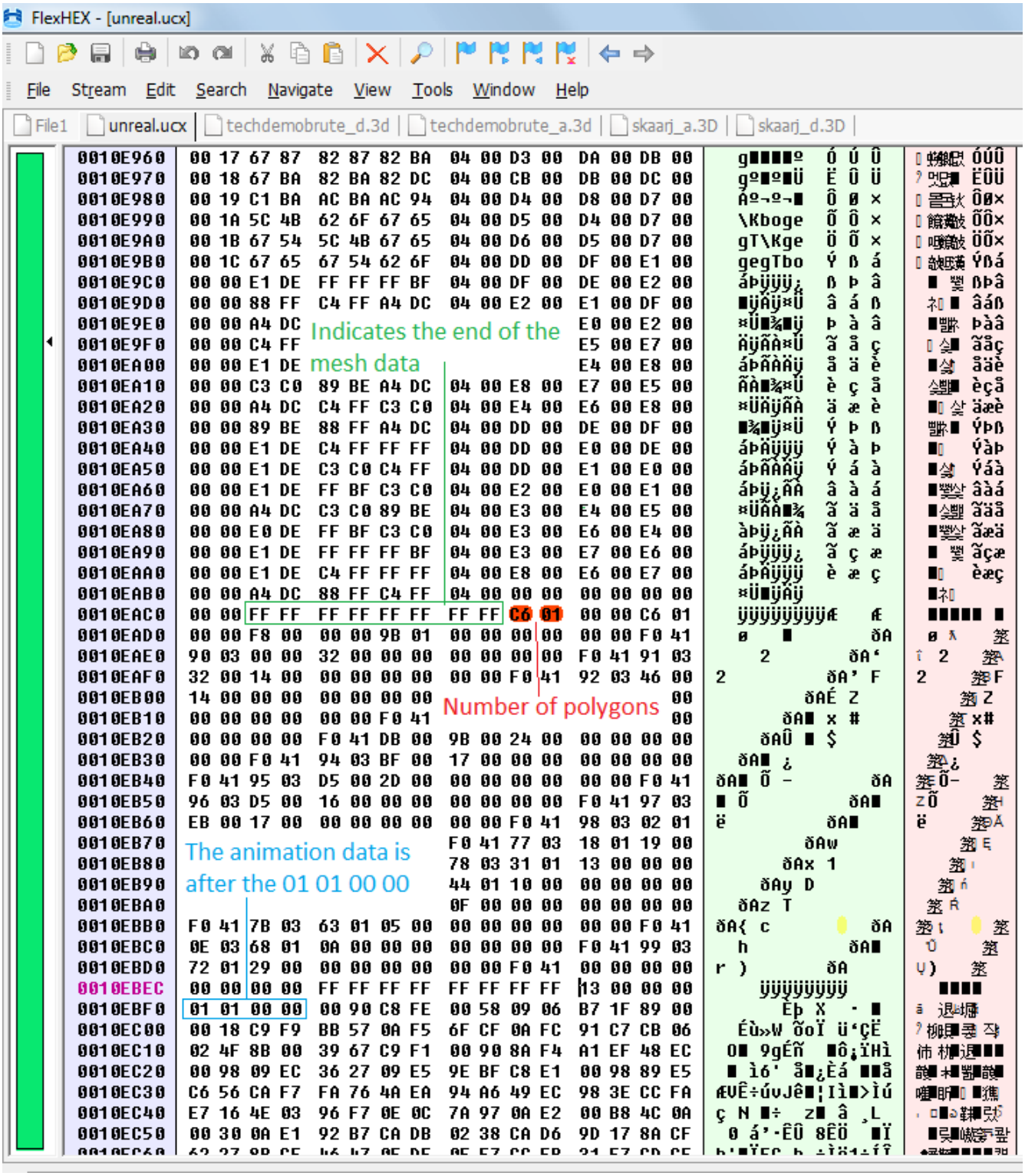


This is the import sequence for the BigMan. Skip it and any code that comes after, and you'll eventually find the mesh. What gives the location of the mesh away is a pattern of ANSI characters that are surrounded by blank ANSI text (there will be hex data there, but it doesn't appear in ANSI). Once you find that, look upward until you find a set of eight FFs, since this shows where the mesh data begins. There will be a set of hex values that increment by 1, followed by three 00s (ie. E5 00 00 00 E6 00 00 00). Follow those values until they stop incrementing. The value that follows will by the number of

vertices the mesh has; in this case, the BigMan has E9 vertices, which is 233 in decimal. After the number of vertices comes the number of animation frames. Keep in mind that Unreal archives are in Little Endian format, so instead of the BigMan having 9B 01 (39,681) frames, it has 01 9B (411) frames. Save both of these values (and their locations) somewhere, you'll need them later. See http://en.wikipedia.org/wiki/Endianness for more information on little endian.

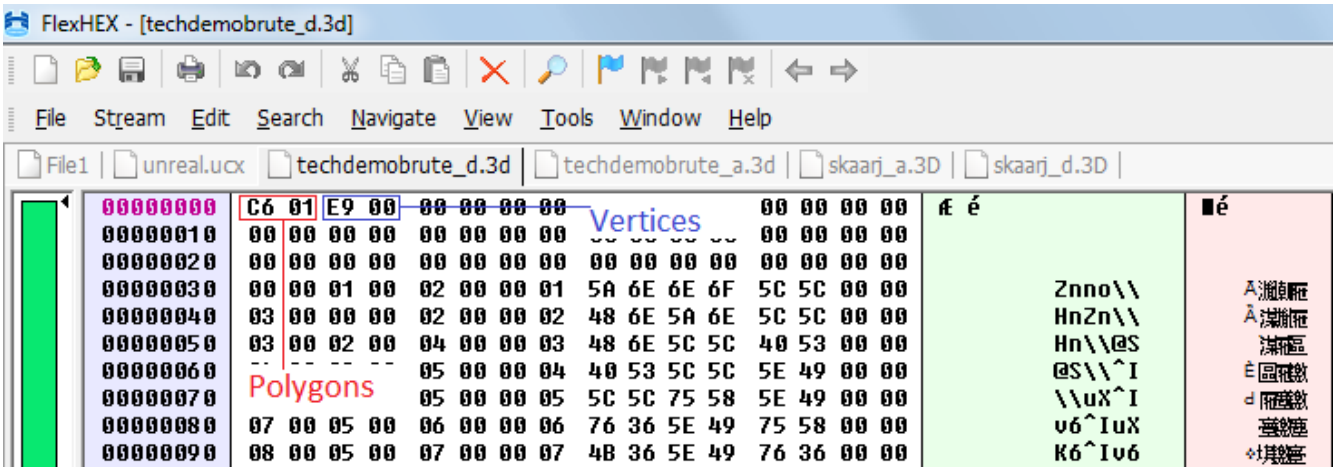


Follow the pattern downward until you find another set of eight FFs. The value after that pattern is the number of polygons in the mesh. In this case, the model has 454 polygons. The should be another set of eight FFs. Look for 01 01 00 00; the animation data begins after this.



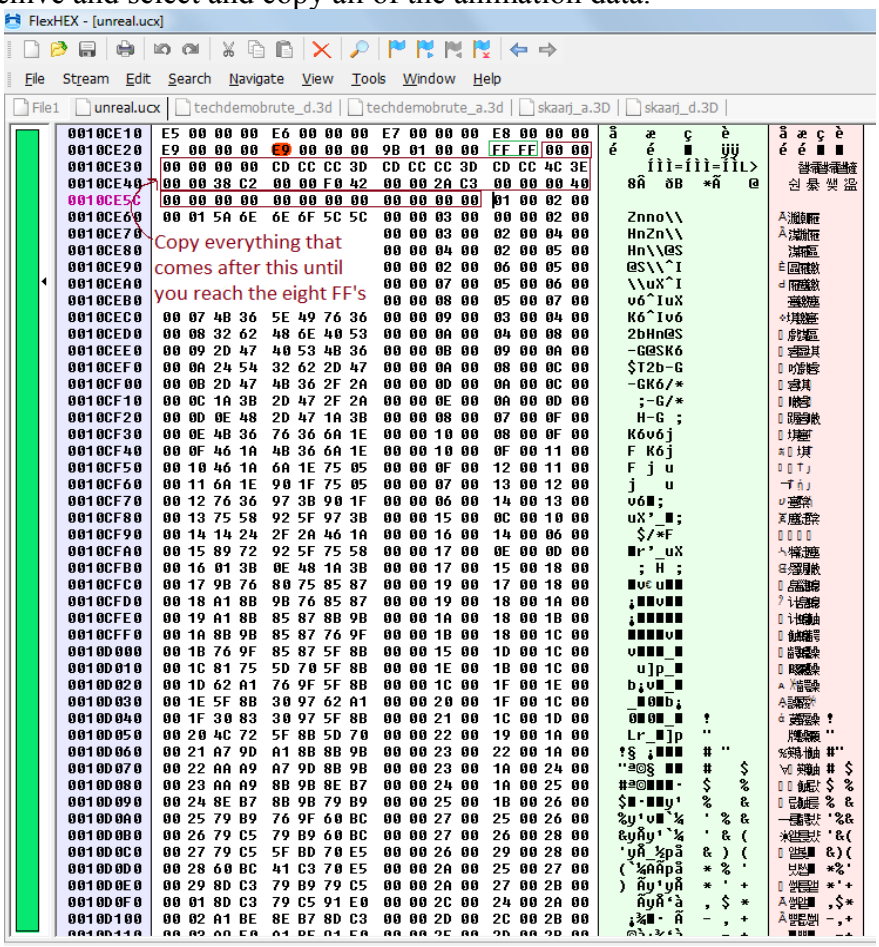
Now that we know where the data is located, it's time to make the data (_d.3d) and aniv (_a.3d) files that can be imported into Unreal. I like to start with the model, so go back to the location in the file that has the number of animation frames. You should see two FFs; you'll need to copy everything after the next 46 bytes until you reach the end of the model data.

Make a new file and insert 50 bytes (all 0). Paste what you copied earlier, and then go to the very beginning of the file and enter the number of polygons and vertices. The result should be like this:

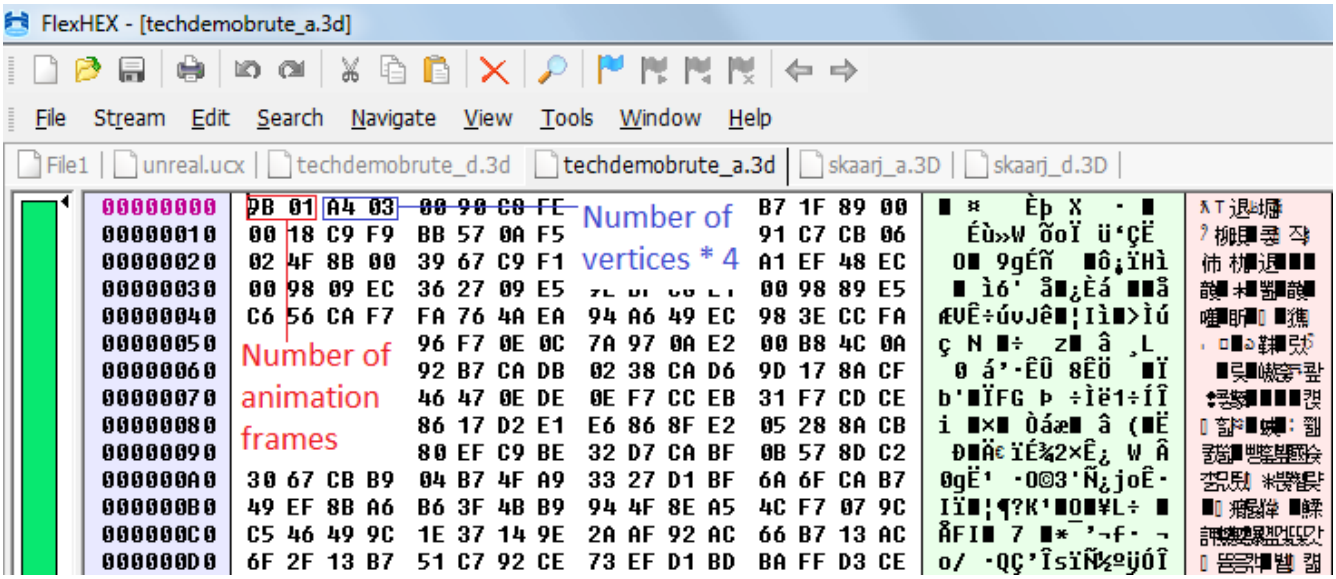


Save the file and give it a name that ends with _d.3d.

Go back to the archive and select and copy all of the animation data.

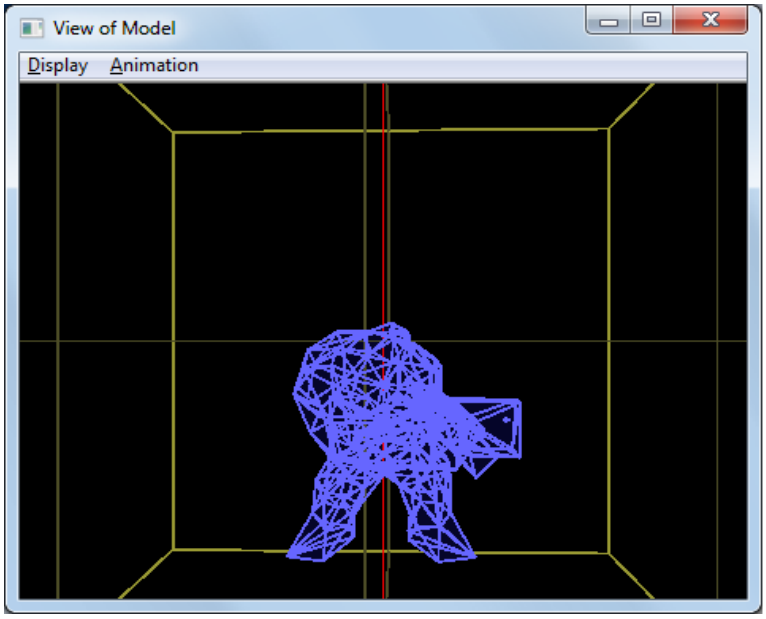


Make a new file and enter the number of frames that you found earlier. Using the calculator, multiply the number of vertices (in hexadecimal) by four. Put that value in little endian right after the number of animation frames. The BigMan has E9 vertices, which is A4 03 in little endian (03 A4 in big endian) when multiplied by four. Paste the animation data. The result should look like this:

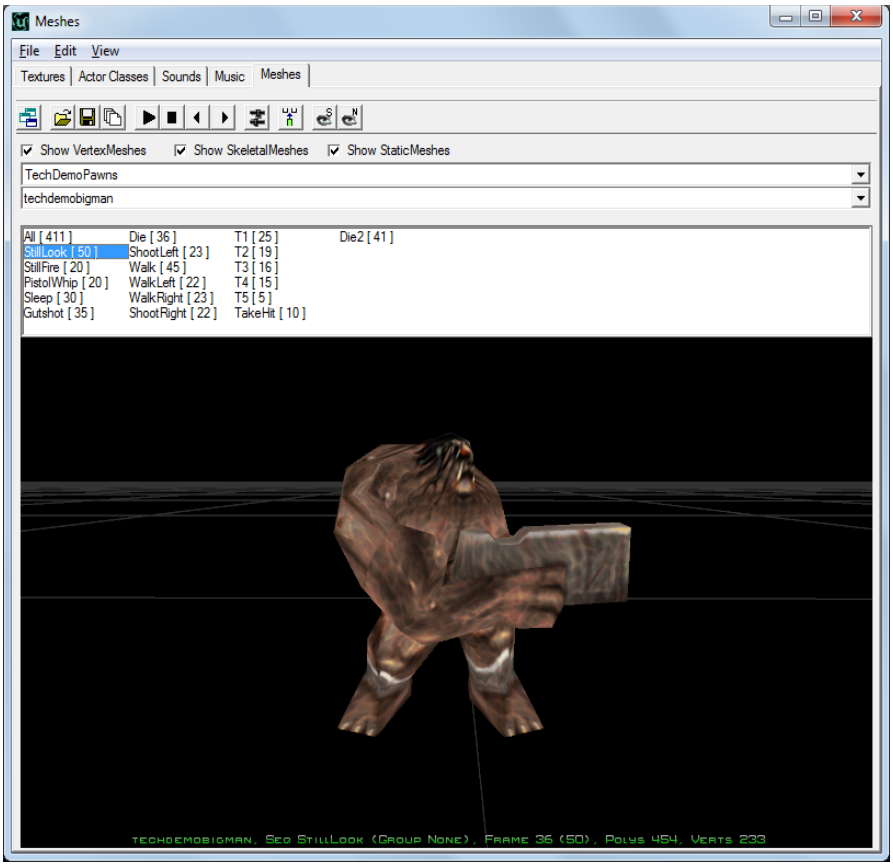


Save the file with the same name as the data file, but make it end in _a.3d instead of _d.3d.

UnrealFX shows that the animations play without any problems...



...and so does the Unreal Editor!



That's all there is to extracting vertex meshes. I haven't tried this with anything aside from the tech demo, beta, and unpatched Unreal, but it should (generally) work for other versions and UT.