

# An overview of the BIGTAIL data type

By Urpo Lankinen

## **Abstract**

This is a non-technical overview of the BIGTAIL data type which is implemented in the Nintendo® GameCube™ hardware.

## **Document history**

### **Revision Date**

v0.1 2004-03-30 First draft, release candidate

v1.0 2004-04-01 Release (same as 0.1 unless details need to be changed)

## **Introduction**

The BIGTAIL data type is one of the programmer-friendly features of the Nintendo GameCube hardware. The data type, implemented in the GameCube's custom Gekko GPU, is used for easy manipulation of a 3D objects and their properties.

This document does **not** contain proprietary information which would be protected by Nintendo as a trade secret. To the best of our knowledge, the data type is not covered by any of Nintendo's patents.

## **The reasons for a new data type**

The design of GameCube was a long and difficult process. The graphical capabilities of game consoles and PCs were getting better and better, soon reaching the level where graphics could be almost photorealistic. As we are writing this, it is widely considered that game graphics are beginning to be of secondary importance to game developers, as there is not that much more to strive for – the graphics are already pretty much as good as they can be.

The primary need for a new storage type was to contain certain level of cuteness. While Nintendo has tried to make their games less cute over past few years, they are still known best for unbelievably cute fun-for-the-whole-family characters. For cartoony style, characters such as Toad or Yoshi were already near perfect on Nintendo 64 platform. Yet, one character posed a difficult problem.

Fox McCloud, the lead character from games such as StarWing and Lylat Wars, was a difficult character to handle because the designers wanted to move away from cartoony graphics style of StarWing to more and more realism. Nintendo 64's graphical abilities had to be pushed. Lylat Wars was already a difficult case

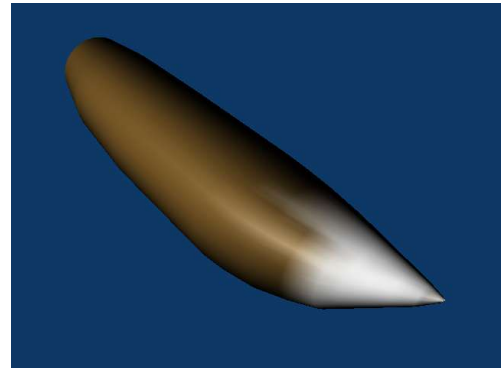


Illustration 1: A sample render of fox tail in smooth Gouraud mode without "FluffyBushy" fur shader

– the characters hadn't completely shaken their cartoony outlook. Most importantly, Fox McCloud's tail was too small in this game. For Super Smash Bros., the designers could push N64 hardware to give Fox McCloud almost a proper size of a tail, but the rest of the character still had rather cartoony look.

The BIGTAIL data type in GameCube was specifically designed to overcome this problem of maintaining realism in Fox McCloud.

## **Data type features**

The BIGTAIL data type is used to hold various kinds of 3D data, such as

- vertex data
- face data
- texture coordinate data
- procedural surface shader data
- animation data

In short, all sorts of data that is required to hold and manipulate the 3D models of any character in the game.

There are specific challenges:

- The tail needs large amounts of vertex data to produce the perfect shape.
- The animation data has to be likewise stored with great care and precision.
- The surface shaders need to be powerful enough to produce enough heart-melting Fluffiness.

The first game to feature Fox McCloud on GameCube, Super Smash Bros Melee, didn't actually fulfill the last requirement due to the lack of complete support in the GameCube development kit, but the other two requirements were there.

## **Advanced features**

What sets BIGTAIL apart from traditional structure types? There are two main things for this – the use of high-precision data types, and the use of shorthands. There's also built-in compression.

The **high-precision data types** store the data in compressed format. The compression saves storage space. There are two compression algorithms used – one is a lossy format that can be used for textures (the system rivals DDS and other texture compression algorithms, just that it's faster and uses less compression, results thus containing noticeably less artifacts), the other is a lossless format used to store arbitrary-precision numbers. The lossless format takes almost no time to decompress due to Gekko hardware support.

The high-precision data types are interesting in that Gekko does arbitrary precision floating point calculations almost as fast as traditional processors do IEEE float/double calculations. The processor uses arbitrary precision BigInts for

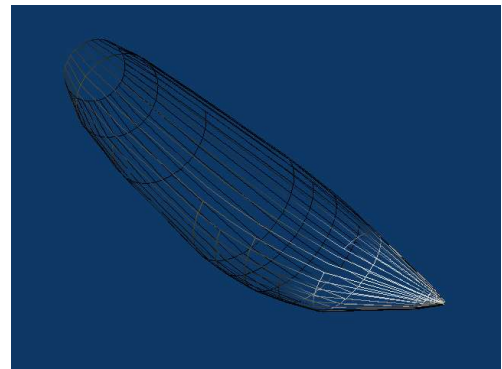


Illustration 2: Example of vertex and face data

almost all integer operations.

The **shorthands** are basically macros built into the hardware, pointers to data stored in GameCube ROM. For example, in Fox McCloud's case, "FluffyBushy" shader would be appropriate for making the fur look correct on tail, arms and head. There are shorthands for all things described in a BIGTAIL structure. For example, there's "Hypnotic" animation shorthand (vertex group-based animation for tail, a good example of which can be seen when Fox runs in Star Fox Adventures). Some of these shorthands are parametric or adaptive, latter meaning that it can use existing model data to its advantage. For example, there is a shorthand that almost automatically wraps the texture around model without even need for pre-specified texture coordinates. (The 3D artists working on the games still prefer to specify UV coordinates themselves, however.)

### ***Data type extensions***

It should be noted that there are extensions to this data types. They are ranked "extensions" because they don't actually work on the BIGTAIL data itself, but rather the data related to the character represented by the BIGTAIL structure. The most notable extensions to BIGTAIL are the links to character voice data. There are some audio filters, such as "Vulpinate", that makes any voice actor sound like a young fearless fox-man – these were used in Star Fox Adventures in the Rare cases when Steve Malpass was sick, they needed some sounds quickly and didn't have time to replace them when mr. Malpass was around again.

### ***Trademarks***

Nintendo and Nintendo GameCube are trademarks of Nintendo.