'Cubiquity' vs. 'Cubiquity for Unity3D'

One of the first points to understand is the difference between 'Cubiquity' and 'Cubiquity for Unity3D'. The first of these, 'Cubiquity', is a native code (i.e. C/C++) library for storing, editing, and rendering voxel worlds. It is not tied to any particular game engine or platform, and can be used from a variety of languages. On the other hand, 'Cubiquity for Unity3D' is a set of C# scripts which connect Cubiquity to the Unity3D game engine. These scripts allow Unity3D games to create, edit and display Cubiquity volumes. As a user of Cubiquity for Unity3D you do not (normally) have access to the underlying C/C++ code of Cubiquity, but you do have a compiled version of the Cubiquity library and the source code to the C# integration scripts.

Calling the Cubiquity Native-Code Library

Functions defined in the native-code library can be called from Unity3D scripts using some magic known as P/Invoke. The file 'CubiquityDLL.cs' uses this P/Invoke technology to provide thin .NET wrappers around each function which is available in the Cubiquity engine. The rest of the Cubiquity for Unity3D scripts are then implemented in terms of these wrapper functions. As a user of Cubiquity for Unity3D you are unlikely to come across these implementation details unless you deliberately go exploring the code (which of course you are welcome to do).

Using native-code allows for some significant performance and memory optimizations compared to systems implemented using just Unity3D scripts. However, it also imposes some limitations, in particular that it cannot be used with the Unity3D web-player because this does not support native code. The native-code library also has to be compiled separately for each platform on which it needs to run (Windows, OSX, Linux, etc), but this is an issue which we as developers have to deal with rather than you as a user.

The Cubiquity Voxel Database Format

Voxel environments can get very large (potentially containing billions of voxels) so efficient storage of such worlds is of utmost importance. The Cubiquity voxel engine stores a volume as a *Voxel Database*, which is a single file containing all the voxels in the volume. Internally it is actually an SQLite database and so can be opened with a tool such as SQLite Browser. Such as tool will let you view certain properties of the volume such as its dimensions, but you won't be able to gain any meaningful insight into the voxel data itself as it is stored in an efficient compressed format.