Obtaining Volume Data

Although Cubiquity for Unity3D comes with built in editor tools, they a currently quite limited (particularly for the colored cubes volume) and it is possible you may wish to obtain your volume data through other sources. Alternatively you may want your volume data to be generated procedurally at runtime rather than designed in advance. This section discusses both these possibilities.

Importing From External Sources

We have already seen how Cubiquity for Unity3D can import existing voxel databases (see **Creating From An Existing Voxel Database**) but where did these .vdb files come from? For colored cubes volumes there are a couple of options discussed below.

Importing From Magica Voxel

You may wish to model your voxel geometry in an external application such as Magica Voxel, Voxel Shop, or Qubicle Constructor. Such applications will typically allow more comprehensive editing options than we will ever provide in Cubiquity, and Cubiquity already provides the option to import Magica Voxel files (others will follow in the future). This is done by the use of the command-line `ConvertToVDB` tool.

To use this tool you should open a command prompt and change to the `StreamingAssets\Cubiquity\SDK` directory. From here you can run:

```
ConvertToVDB.exe -i input.vox -o output.vdb
```

You can then copy the resulting .vdb into the `StreamingAssets\Cubiquity\VoxelDatabases` folder before following the instruction in **Creating From An Existing Voxel Database** to import it into Unity.

Importing From Voxlap

You can also use the 'ConvertToVDB' tool to import maps in the Voxlap '.vxl' format as used by games such as Build and Shoot. This allows you to use editors designed for this game to build maps for **Cubiquity**.

```
ConvertToVDB.exe -i input.vxl -o output.vdb
```

Note that these maps are expected to have dimensions of 512x512x64 voxels.

Importing From Image Slices

The same tool can be used to import colored cubes volume from a series of color images representing slices through the volume (see `Assets\Cubiquity\Examples\VolumeData\VoxeliensLevel3` for an example of such a series of slices). This provides means to get data into Cubiquity from any other application provided you can write an exporter to output these slices. You can also use this approach if you want to write a standalone program which generates volume data off-line and which can then be imported into Cubiquity (this is faster than generating volumes though C# scripts).

You can call the converter in the same way as before, but providing a path to a folder containing image slices rather than to a Magica Voxel .vox file:

```
ConvertToVDB.exe -i input.vox -o output.vdb
```

Note that both Magica Voxel and image slices are only appropriate for importing colored cubes volumes. Currently there are no methods for creating terrain volumes *outside* of Cubiquity for Unity3D, but you can still create them procedurally as discussed later.

Generating Volume Data Through Scripts

Cubiquity for Unity3D provides a very simple but powerful API for generating volumes through code. Each volume is essentially just a 3D grid of voxel values, and the API gives you direct access to these through the VolumeData's GetVoxel(...) and SetVoxel(...) methods. You can then choose any method you wish to decide which values should be written to which voxels. Common approaches include:

Using a noise function: Evaluating a 3D noise function (such as Perlin noise or Simplex noise) at each point on the grid can generate both natural and surreal environments. Multiple octaves of noise can be combined to add additional detail. Please see the 'Procedural Generation' example in the examples folder.

Reading an input image: The 'Maze' example (see the examples folder) reads a 2D image of a maze and sets the height of voxel columns based of whether the corresponding pixel is black or white. The same principle can be applied to generating a terrain from a heightmap.

Building planets from spheres: You can create spheres by computing the distance function from a point, and a cube map can then be used to apply a texture. The 'Solar System' example shows how this can be used to create planets.

Overall there is a lot of potential for using your imagination here. If you can think of a way to generate a particular type of volume data then Cubiquity provides a way for you to bring it to life.