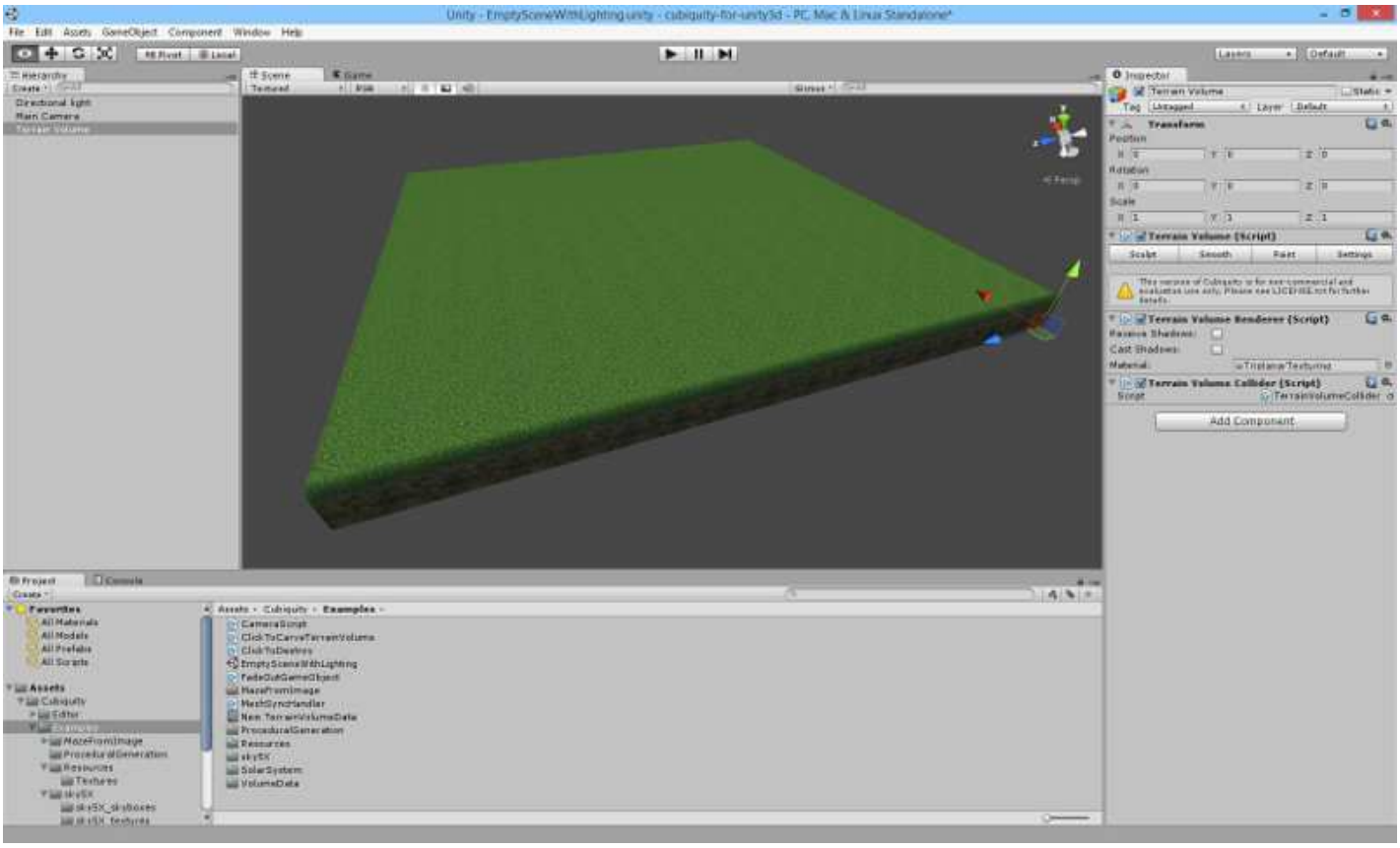
Quick Start

Cubiquity for Unity3D is designed to be as easy and intuitive as possible, and so this quick start guide should be all you need to begin creating voxel worlds for your games. If you have any difficulties or wish to follow up some topics in more depth then you can see the appropriate sections of this user manual for more details.

Creating your first voxel terrain

Cubiquity for Unity3D supports two types of voxel environments. We will begin by looking at the *Terrain Volume* which, as the name suggests, is intended for representing natural terrains. From a user point of view it is similar to Unity3D's built-in terrain, but additionally supports caves, overhangs, and flexible run-time editing.

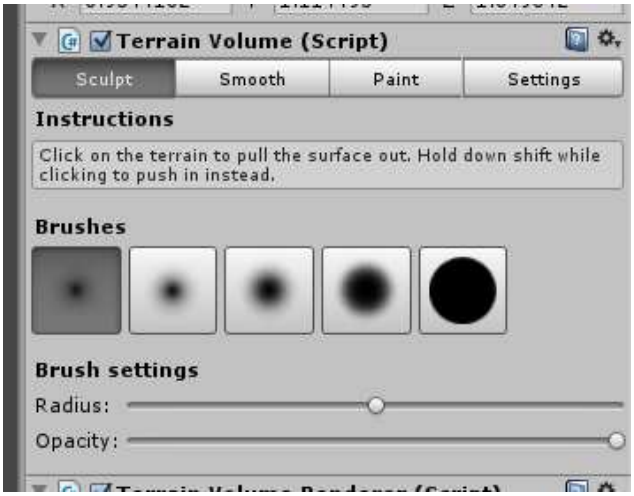
Begin by opening the scene 'EmptySceneWithLighting' in the `Assets/Cubiquity/Examples` folder. Now create a Terrain Volume from within the Unity3D editor by going to the main menu and selecting `GameObject -> Create Other -> Terrain Volume`. The terrain volume will be created at the origin of your scene and should appear as shown in the image below. If you do not see the volume then your camera position may be wrong, you can fix this by double-clicking the new 'Terrain Volume' in the scene hierarchy.



A newly created terrain volume

Editing the voxel terrain

Your new terrain should be automatically selected, and as with any other Unity object you can scale, rotate, and translate it by using the the usual gizmos. To actually edit the terrain you need to select one of the tools from the 'Terrain Volume (Script)' component in the inspector. For example, you could select the 'Sculpt' tool, and then when you move your mouse cursor over the terrain you should see a light-blue *brush marker* following your cursor position. This indicates where any terrain modification operations will be performed. You can click with the left mouse button to begin adding material into the terrain, thereby creating hills and ridges.



Terrain volume sculpting tools in the Unity3D editor

Take some time to experiment with the editing tools which are available. You can choose a tool to apply (sculpt, smooth, etc) by selecting one of the buttons at the top of the inspector, and then choose your desired brush options and/or materials. Left-clicking on the terrain will then apply the tool.

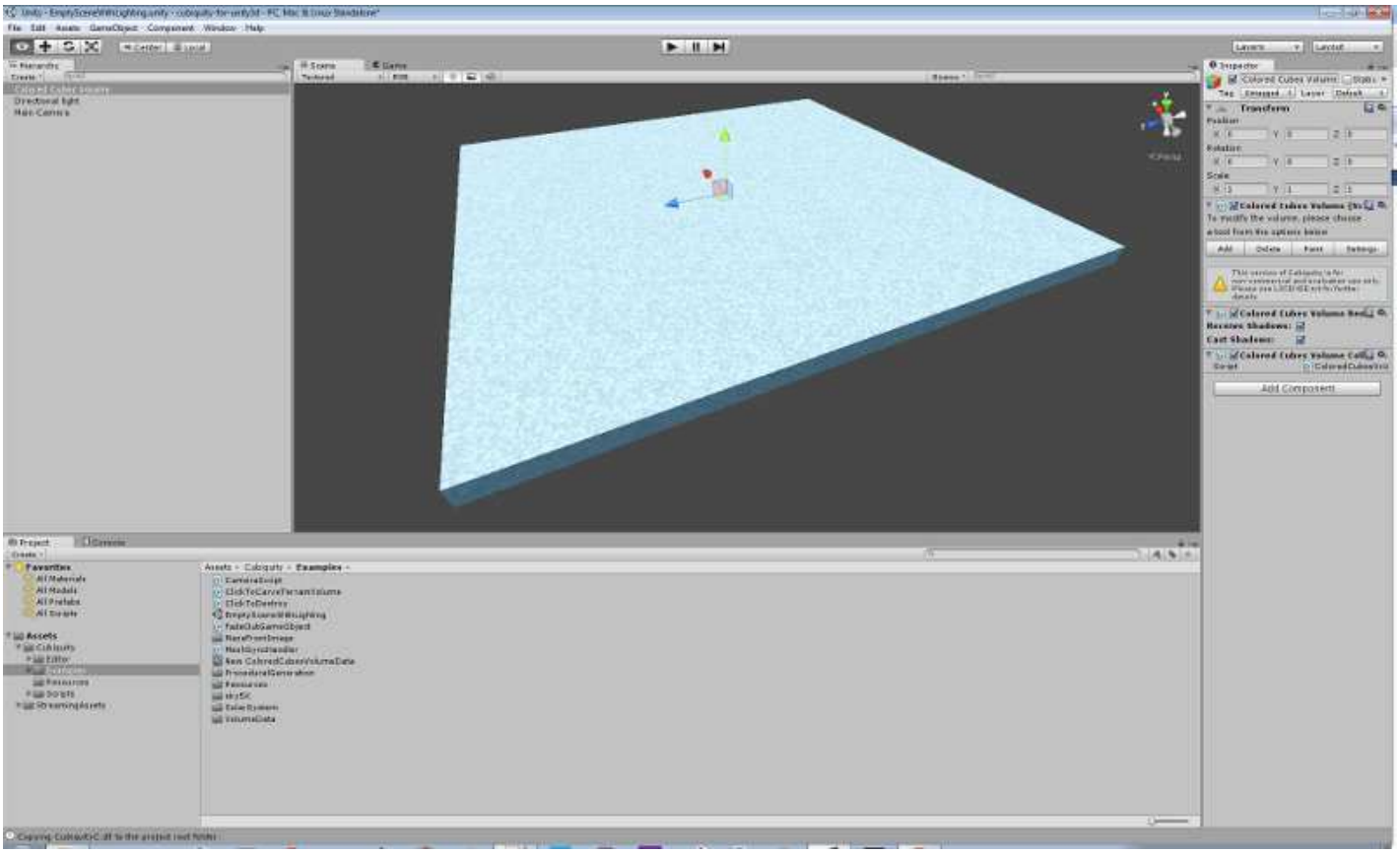
Now lets try adding a collider so that other objects can interact with the terrain. To do this we add a 'Terrain Volume Collider' component through `Add Component -> Scripts -> Cubiquity -> Terrain Volume Collider`. Note that this is different from the 'MeshCollider' which is often added to other Unity objects.

To test the collisions we can now import one of the standard Unity character controllers and walk around the terrain in play mode. Be aware that it can take a few seconds for the terrain to generate after you press the play button, so for this reason you should set your character to start a hundred units or so above the terrain. This way the terrain will have time to load before the character reaches ground level (there are better approaches, but this is fine for quick-start purposes).

Creating your first colored cubes volume

The TerrainVolume presented in the previous section is intended for creating real-world terrain with realistic textures and materials applied. But Cubiquity for Unity3D also supports a second type of voxel environment in which the world is built out of millions of colored cubes. This is not so realistic, but is an increasingly popular approach which has a strong stylistic appeal. Almost any kind of world can be built in this way, and it again provides opportunities for editing the world in response to player actions or gameplay events.

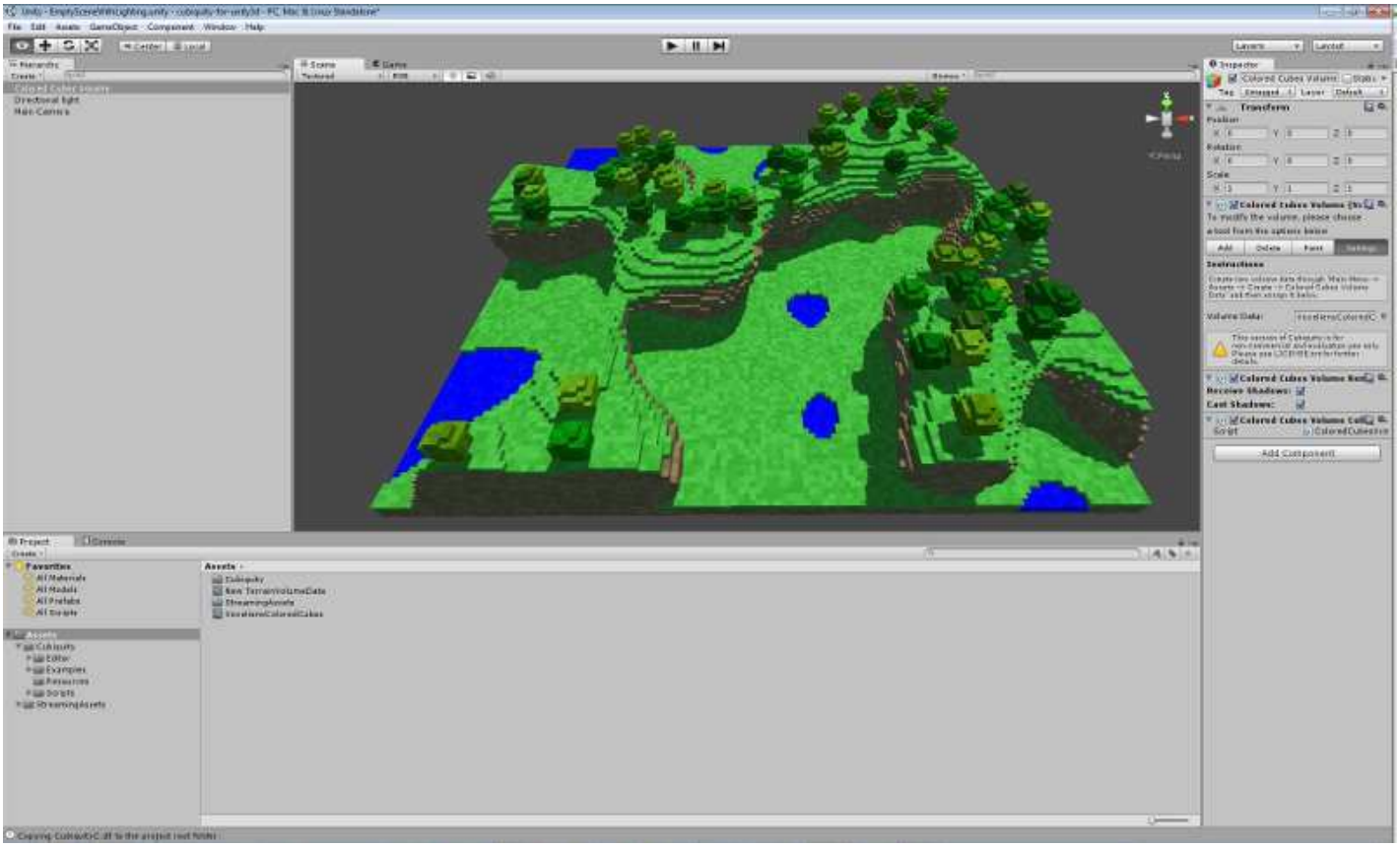
To see this in action you should return to edit mode and delete the terrain volume which you currently have in the scene from the previous section. You should also delete the character controller which you added. You can then create a *Colored Cubes Volume* by going to the main menu and selecting `GameObject -> Create Other -> Colored Cubes Volume`. The initial volume should look like that shown below:



A newly created colored cubes volume

As with the Terrain Volume, the Colored Cubes Volume comes with a custom inspector which is shown whenever it is selected. However, the editing facilities of this are currently very limited, and only allow you to create single cubes at a time by left-clicking on the volume. We will probably add more advanced editing in the future, but you should also consider creating your Colored Cubes Volumes in an external application and or generating them procedurally. This is described further in later sections of this user manual.

Let's try replacing the volume data with one of the example volumes which comes with Cubiquity. To do this, select the colored cubes volume, go to 'Settings' in the inspector, and click the small circle next to the 'Volume Data' field. From here you can select the 'VoxeliensColoredCubes' asset which will cause it be be used as the source data for the volume (this is a map from our previous game Voxeliens).



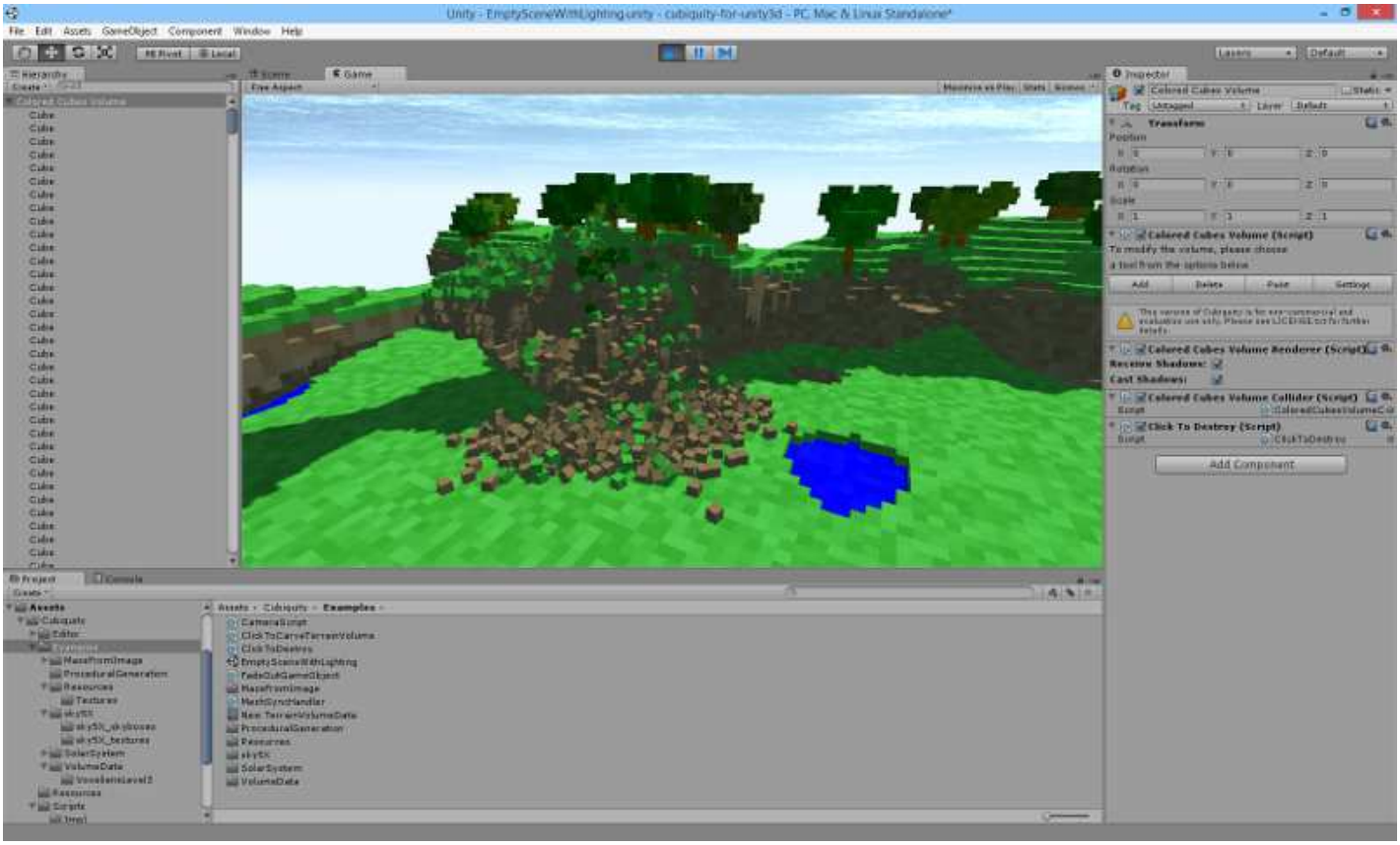
The colored cubes volume after importing our map from Voxeliens

The asset you have selected is actually just a very thin wrapper around our Cubiquity *voxel database* file format. If you click on the 'Volume Data' field (rather than clicking on the circle, as you did previously) then Unity will show you that the asset actually exists in 'Assets/Cubiquity/Examples/Basic'. If you now select the asset through the file browser you will see some information about it in the inspector, including the path to the database which actually holds the voxels. Later sections of this manual give further details about the relationship between volume assets and voxel databases.

Modifying the volume at run-time

We will now give a quick demonstration of how the volume can be modified during gameplay. Go to the 'Assets/Cubiquity/Examples/SharedAssets/Scripts' folder and find the 'ClickToDestroy' script. Drag this on to the Colored Cubes Volume in the scene hierarchy to add it as a component. We will also want a collider for this to work correctly so add one via `Add Component -> Scripts -> Cubiquity -> Colored Cubes Volume Collider` (note that a Colored Cubes Volume Collider is different to the Terrain Volume Collider you used in the earlier example)

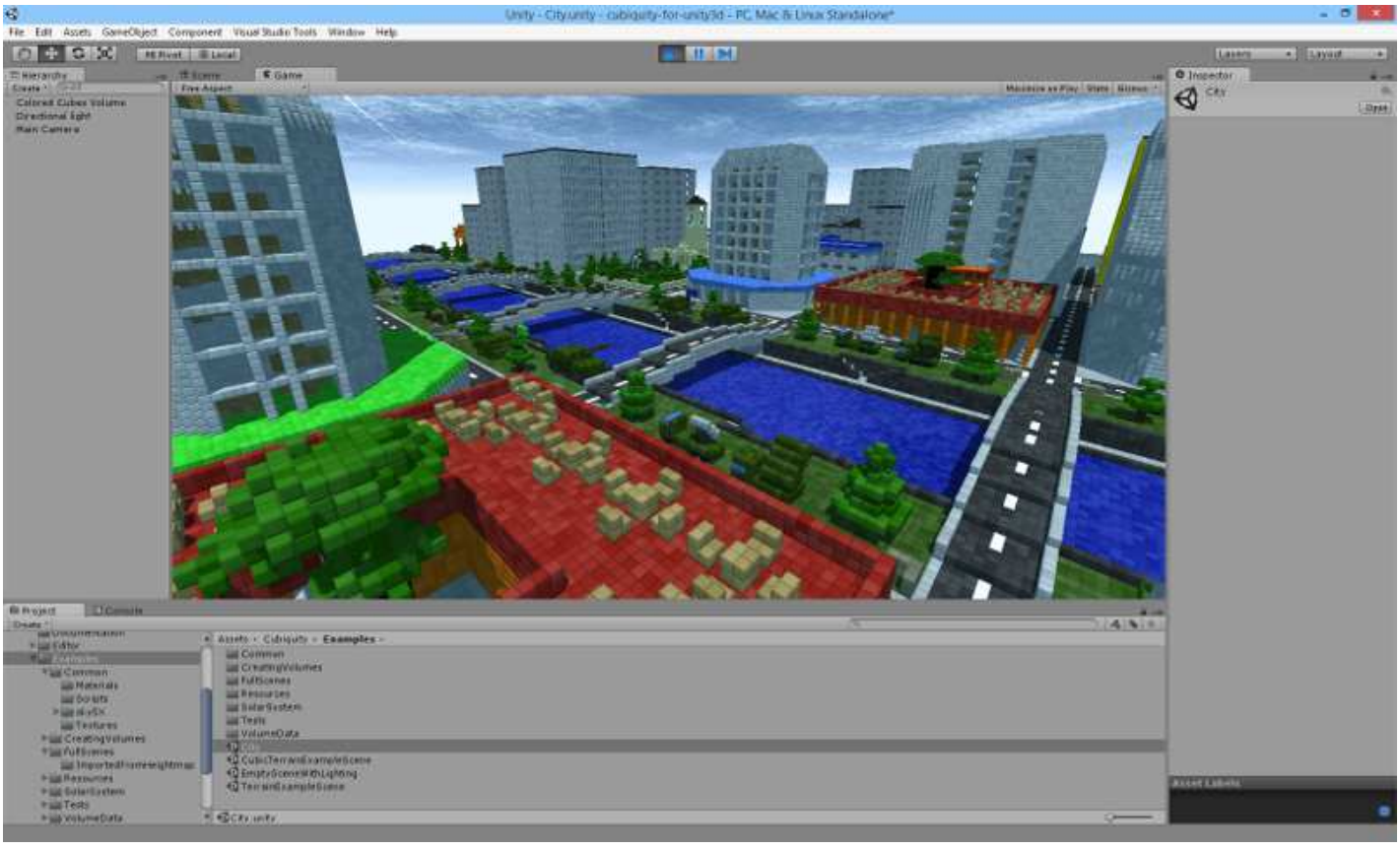
When you press play you should find you are able to fly around the scene, and that if you left-click on the volume it will create a small explosion which breaks off the cubes in the surrounding area. The separated cubes then fall under gravity and can bounce around the scene. This is simply an example of the kind of functionality we can achieve, and you can learn more by reading the **Class List** later in this user manual, or by looking at the code in ClickToDestroy.cs.



Destroying a colored cubes volume at runtime

Other volume data

If you want to see something a bit more impressive than this small terrain then you can also look at our city map. Return to edit mode, and again select the colored cubes volume, go to 'Settings' in the inspector, and click the small circle next to the 'Volume Data' field. From here you can select the 'ImportedCity' asset. This volume was imported from the game Build and Shoot using our 'ProcessVDB' tool. For more information on importing volume data see the **Obtaining Volume Data** section of this user manual.



A city map imported into Cubiquity from the game 'Build and Shoot'

That wraps up the quick start, so feel free to play around with the system some more and move on to the other chapters when you feel ready.