Thermite3D Manual

Resource System

The Ogre graphics engine (upon which Thermite is built) provides an advanced resource management system. Thermite builds upon this system and so many of the resource types recognised by Ogre are automatically supported by Thermite. In addition, Thermite extends some of these resource types and adds new ones.

Where are resources/how are they found?

Figure XXX illustrates some of the available resource types. In this figure, resource types which are part of Ogre have a red background, while those which have been added for Thermite have a green background. An arrow from one resource type to another indicates that the latter is in some way dependent upon the former. It is something of a simplification but is useful to aid understanding:



An explanation of each of the resource types is as follows:

**GPU Program Source:** These are files containing OpenGL GLSL, Direct3D HLSL, or NVidia Cg program source code. They typically intended to form part of a vertex, fragment (pixel), or geometry program. The same program code may be shared between multiple programs, for example, a Perlin noise function might be used to perform displacement in the vertex shader or procedural texturing in the fragment shader. These files will typically be written by a programmer or technical artist.

Note: The terms 'GPU program' and 'shader' are often used interchangably in graphics literature. A shader is a program which runs on the GPU, but it can also refer to the hardware unit which executes the program.

**GPU Program:** These connect shader source code with other information (such as the entry point, supported GPU profiles, default paramaters, etc) into a complete GPU program. Again, they are typically written by a programmer or technical artist.

**Texture Data:** This is typically images which are applied to the surface of objects, but it can also be used as a way of passing arbitrary data to GPU programs (e.g. normal maps, lookup tables, etc). Texture data is most commonly created by artists.

**Material Script:** These combine GPU programs with texture data and graphics card state (blending modes, etc) to create the final look of a surface. They are conceptually similar to Direct3D .fx files, or to NVidia's CgFX system. Ogre material scripts also have a lot of advanced properties including supporting inheritance and level of detail. More information about material scripts can be found in the Ogre manual:

<http://www.ogre3d.org/docs/manual/manual_14.html>

This also gives further details about the shaders and GPU programs mentioned previously.

**Compositor Script:** These are used to define operations which are applied to the final image as post-processing effects. For example, effects such as depth-of-field, colour correction, and high dynamic range rendering are likely to involve the use of compositors. They may also make use of Ogre material scripts to control their behaviour, and will usually be implemented by programmers or technical artists. More information about compositors can be found in the Ogre manual:

<http://www.ogre3d.org/docs/manual/manual_29.html>

**Mesh Data:** Most objects in the game world (such as characters, weapons, vehicles, etc) have their geometry described by meshes. The exception to this is the world geometry which is represented by voxels (the defining feature of the Thermite3D engine). Ogre supports both skeletal and vertex animation for meshes, and allows complex materials to be applied. Meshes are usually created by an artist using a 3D modelling package, before being converted to the Ogre format.

**Particle System Script:** In game effects such as fountains, sparks, and explosions are often modelled using particle systems. Ogre's particle system scripts allow you to define the behaviour of particle emitters as well as the way the particles themselves behave. These scripts are typically written by programmers or technical artists, and more information about them can be found in the Ogre manual:

<http://www.ogre3d.org/docs/manual/manual_34.html>

**Ogre Scene:** This is an XML file which describes the properties of all the elements which make up an Ogre scene. For example, it can reference a .mesh file and augment it with additional information such as it transformation and whether it should cast shadows. As another example, it can be used to describe the lights which exist within a scene.

More details about the .scene format can be found on the Ogre wiki:

<http://www.ogre3d.org/wiki/index.php/DotScene>

<http://www.ogre3d.org/wiki/index.php/DotSceneFormat>

Note that Thermite does not make use of Ogre scene files directly. Instead they are used as a basis for Thermite's map format, which extends the scene format with XML node types which are specific to Thermite.

**Volume Data:** Support for representing environments using volume data is, of course, the key defining feature of the Thermite 3D engine. This volume data is stored in a compressed form in .volume files. As has been mentioned (see XXX), each voxel in a Thermite volume is a single byte (a 'material index', in the range 0-255). A material index of 0 represents empty space, and any other value represents a solid material.

**Thermite Map:** A map in Thermite is an extension of an Ogre scene. It supports all the properties of a scene (meshes, lights, etc) but adds the ability to reference volume data for the purpose of representing a destructible environment. It does this using the following XML node:

<volume source="VolumeName.volume">

<voxel value="1" material="FirstMaterial"/>

<voxel value="2" material="SecondMaterial"/>

<voxel value="3" material="ThirdMaterial"/>

</volume>

In the current version of Thermite only one volume is supported per map. Hence this volume node should not be placed inside an Ogre scene <node ">...</node> pair, but should instead sit directly within <node ">...</node>.

TODO: Describe new physical properties for meshes.

Reference relevant Ogre wiki pages?