Static Objects

General

Static objects are placed in the editor in the static objects edit mode. Static objects are usually the performance wise the best objects since they are combined when the level is loaded. What happens is that the engine will look for nearby objects that share the same material and settings (shadows, collider, occluder, etc) and combine these into a single mesh. This heavily reduces the number of draw calls the renderer has to do and thereby speeds up the rendering.

Texturing

Because object using the same texture (.mat file really) it is important to have this in mind during texturing. Static objects that are often nearby one another (walls of certain type, pipes, etc) should all share the same material. So it is often good to have a really big texture that is shared among many objects, up to as large as 4096×4096.

It is important to not get carried away with this thinking though. First of all large textures may take longer to render and thus it can slow down if it does not reduce the number of draw calls enough. Also important to not have things that are almost always separate (say floor for a corridor and ceiling used in an office) on the same texture, because that will only bring up the memory requirements as part of the texture might not be used on certain levels.

Another aspect to this is that you can only set a single physics material per mat file. When generating physics colliders from mesh in the model file, the physics material set in the .mat is used. Because of this, it is important not to have too many different surface materials in the same texture. For instance instead of having steel beams and wooden wall in the same texture, make these two separate. However, do not take this thinking too far. This only applies to large surfaces. For instance, if you have many small metal pieces on a large wooden surface, there is no need to split this into several texture.

Collision

For most of the time, collision is automatically generated from the mesh of the object. This works fine for most of the time and when possible it is wise to spend a little less polygons a on an object with collision or to let more detailed (non-colliding) parts be in a separate object. The physics material of a collider generated this way is determined by the material (.mat) file that the mesh has, so make sure to set this up in the material file for all static object materials!

Sometimes though, the poly-count cannot be reduced enough (like in cylindrical shapes), or the collision needs to be different from the actual geometry (like a stairs, where you want a slope for characters). Then you need to add custom collides, this is done by adding a .ent file or using naming of the submeshes. As discussed more later on, ent files should be used for most objects and only when a mesh (not made out of primitive shapes) collider is needed, should the naming be used.

First of all, the collider must be child of one of the other (non-collider) submeshes in the dae file! This is because the material of the parent is used to get the physics material for the collider.

Then it has to be named according to a syntax, which is as follows:

collider[type]_[name]
charcollider[type]_[name]
If starting with "_charcollider", only characters will collide with it, otherwise any body will collide.

type
can be one of the following:
mesh This means that the submesh will only be used collider and not be visible. Works very will for making a low-poly version as a collider.
box (LEGACY)
cylinder Height in local coordinates along y-axis! (LEGACY)
sphere Must not be perfectly spherical! (LEGACY)

name
can be what ever you want!
Example:
_collider_mesh_shape01

Important note:
the types box, cylinder and sphere are legacy stuff! It is almost always better to use .ent file file!

Entity File

For more info om entities see here.

Static objects support having a .ent file, but this is not needed. The engine will check if there is an entity file with the same name as the model (eg, if "wall.dae" has "wall.ent") and if so load it. The only thing loaded from this file are:

Bodies & shapes
Only options that are viable are BlocksLight, BlocksSound and CollidersCharacter/CollideNonCharacter. Regarding the collision, only two setups are supported: Both on, or only charactercollider on (it is all saved internally as a single bool "CharacterCollider", which if on means it does not collide with normal bodies).

Submesh properties
This is bascially just the transforms for the submeshes, which allows a single dae file to have several configurations of the submeshes.

Also note that the material of the bodies are now determined by what is set in the ent file and NOT of the material (.mat) of the mesh.