Entities

General

Generals all use .ent files that are created in the model editor tool. Exactly how an entity functions is largely up to the app using the engine. Entities can be background props, enemies, and pretty much whatever. For more specific information on the basic types that are provided in the default game setup, check the Game Scripting Guide.

On this page the more general structure of entities will be covered.

Lowlevel Structure

At the very basic level an entity comes as an .ent file that is loaded by the engine. The engine provides a basic loader for entities that handles loading of the basic types contained in it. The game can provide their own loader built from scratch, but one almost always wants to use the loader that the engine provides. The default game provides a few basic types like like Props (doors, buttons, dynamic crates, etc) and Agents (enemies, NPCs, etc) that are used as a foundation for the actual entity used in the game. The final entity type is defined by a script file, which takes care of all specific entity loading, setup and logic.

In order to specify a type it first needs to be added to "`config/EntityTypes.cfg`" under the basic game type is should used (Prop, Agent, etc). Here the name (used as an identfier), the script file, the class name (refering to the main class in the script file) and if it is forced to have full game save (if all data is always saved) is specified. Then the editor must also know about this type and do do this "`edtitor/EntityClasses.def`" needs to be updated with the type's variables. Once this is done, the type is ready to be used in game and editor.

Example:
The entity "Prop_Lamp" uses the basic type "Prop" and is meant to be used for any lamp-like objects in the game. The file "`config/EntityTypes.cfg`"has been updated with:

```
<PropType
  Name = "Prop_Lamp"
  ScriptFile = "props/Prop_Lamp.hps"
  ScriptClass = "cScrPropLamp"
  ForceFullGameSave = "false"
/>
```

This is added to the `PropTypes` element and contains the needed data for the game
To use it in the editor, "`edtitor/EntityClasses.def`" has also been updated with the following:

```
<Class Name="Prop_Lamp" InheritsFrom="Prop">
    <EditorSetupVars>
        <Var Name="AffectLightsVar" Value="Lit" />
        ...
    </EditorSetupVars>
    <TypeVars>
```

```
        <Var Name="CanBeLitByPlayer" Type="Bool"  DefaultValue="true"
Description="If the player can lit this lamp by direct interaction." />
        ...
    </TypeVars>
    <InstanceVars>
        <Var Name="Lit" Type="Bool" DefaultValue="true" Description="If the
lamp is lit or not." />
        ...
    </InstanceVars>
</Class>
```

This makes the editors (model and level) aware that the type exists and provides them with info on what variables it should have as well as any special behavior. More specifics are found below.

Data Structure

An entity is built up from several different elements, all contained or referenced to by the ent file. None of these are required, and an entity can contain all of these types, just a single one or any variation (except for animations which require a mesh).

Mesh

This is the core object for almost all entities. The data is a mesh file (".dae" or ".msh") that can that comes in the form of a separate file which the entity references to. An entity can only have a single mesh connected to it. Note that several entities can connect to the same mesh file.

Animations

An entity file can have one or more animations. In order to have animations there needs to be a mesh connected to the entity file as well. Animations are connected to the entity as separate files (".dae_anim" or ".anim"). It is okay if several entities share the same animations.

Bodies

This gives the entity its physics properties and allows it to collide and interact with the game's world and other entities. A body is made up from more or many shapes. If the mesh has several submeshes and there are several bodies, it must be specified which submesh belong to which body. This data is defined in the entity file.

Joints

These are used to joint up several bodies in various ways (e.g. create a door hinge between door and frame) or just attach a single body to the world in some way. This data is defined in the entity file.

Effects

This includes stuff like particles, billboards, flares, sounds, etc. Everything that can be added to spice the entity up. If there are multiple bodies then it needs to be specified where each effect is attached. Effects can also be attached to the bones of a skinned character. This data is defined in the entity file.

Creation Workflow

Level Editor variables