Entity

Entities are like GameObjects in Unity. They make up the contents of the scene. Attaching components to an entity defines their behaviour. The design of entities and their components follows the ECS architectural design pattern.

**Entity Description**

The entity class serves as an interface so you can define different types of entities by making a new **subclass** of the entity class. These subclasses entities must exist in the **sandbox** project and they allow you to create entities with certain properties and behaviours. However, for an instance of the subclass to be used, they (entity) must be added to the scene otherwise the engine will not process them. Adding an entity to the scene will set the entity’s **parent scene** variable and the entity’s **name** variable. These will be set by the scene’s **addEntity** function. The name of the entity in the scene list will be this name at this point in time, although the internal entity name can change, the name in the list will always be the original name.

The subclass has access to functions which it can override from the entity base class, these consist of the user input functions: **onWindowResize**, **onWindowFocus**, **onWindowLostFocus**, **onWindowMoved**, **onKeyPress**, **onKeyRelease**, **onMousePress**, **onMouseRelease**, **onMouseScrolled** and **onMouseMoved**. The two types of update functions (**preUpdate** and **postUpdate**) and finally a render function (**onRender**) so the subclass can define *how* to render this entity.

Entities inherit the destroyable class so they can be deleted by scheduling for their deletion through the **destroy** function which will delete the entity when the scene is next updated.

**Layers**

Additionally, an entity must belong to a layer otherwise it will not be rendered, each Scene has a layer manager, and the entity can set its layer through the entity’s **setLayer** function and the layer can be retrieved by accessing the Layer Manager through the scene class you are adding the entity to. Entity -> Parent Scene -> LayerManager. Layer will only be set if it exists in the entity parent scene. So set the layer **after** adding the entity to the scene.

**Defining the characteristics of an Entity**

You can attach and detach components to the entity to begin defining the characteristics of the entity. See the component documentation for the different types and descriptions.

**Scene Loader**

While entities can be created at runtime through classes, entities can also be loaded from file when the scene is loaded. The entities.json file in the scene folder allows you to define entities for that scene, the data being stored in JSON format. Any new entity subclasses must be defined in the scene loader class for the loader to load the new type by converting the string literal type to the runtime class type.

**Creating New Entity Type Instructions**

1. Create new subclass in sandbox project which inherits from Entity
2. Override (but not required) the following functions:

onPreUpdate

onPostUpdate

onWindowResize

onWindowFocus

onWindowLostFocus

onWindowMoved

onKeyPress

onKeyRelease

onMousePress

onMouseRelease

onMouseScrolled

onMouseMoved

onRender

1. Modify scene loader’s **createNewEntity** function so it can load the new entity subtype from file & add **include** to sceneLoader.cpp
2. Add entity to the scene (unless loading from file as loader will do that for you)
3. Add the layer in which the entity belongs on (unless loading from file as loader will do that for you)
4. Attach components to the entity