Chrome Engine

Documentation Technique

Propulsé par

Etudiants :

Guillaume Castellana (castel_g)
Guillaume Casalis (casali_g)
Laurent Catala (catala_l, Dublin)
Geoffroy Laptes (laptes_g, Dublin)
Dylan Oudin (oudin_d)

Table des Matières

Architecture et interactions

Diagramme de Classes

Vous trouverez ci-dessous un diagramme simplifié montrant l'héritage des classes à l'intérieur du Chrome Engine.

En bleues les classes dites « mères » et en rouge les classes dites « filles ».

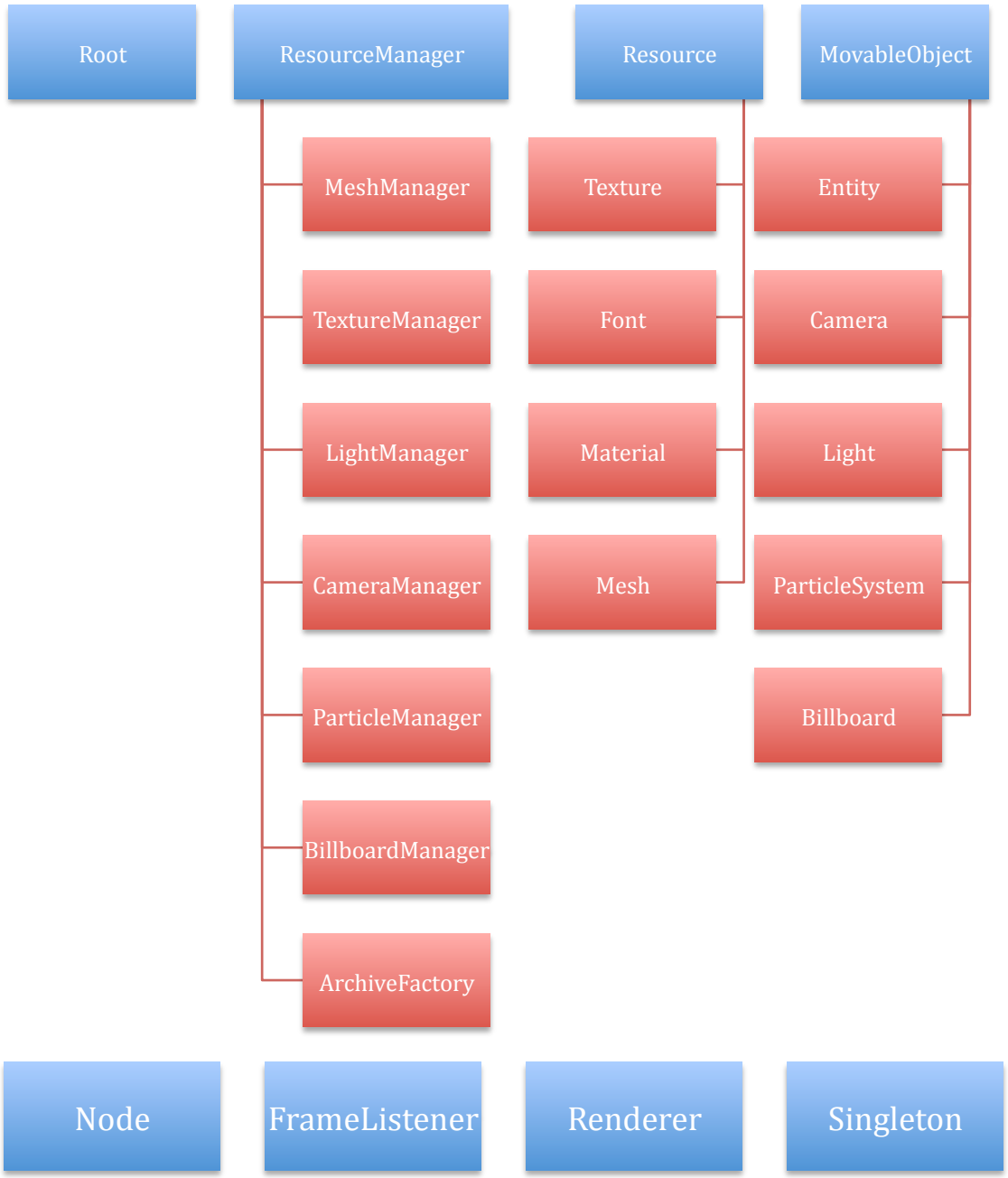
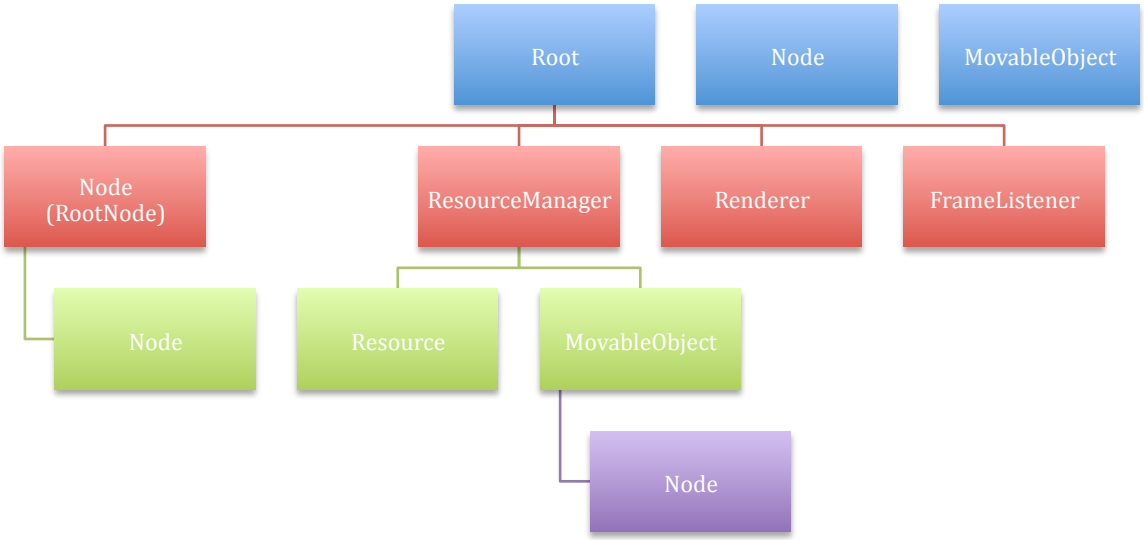


Diagramme de Communications



Le diagramme ci-dessus montre les différentes interactions entres les composants du Chrome Engine.

La classe Root représente bien entendu le noyau (ou Core) du moteur. Elle contient les pointeurs menant aux autres objets instanciés et régit la majorité des instanciations de l'application.

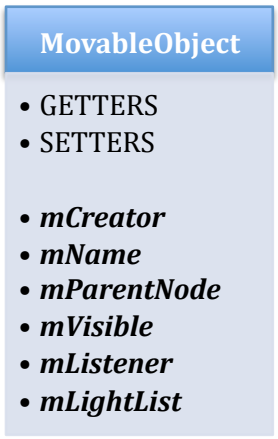
Un autre mécanisme important du Chrome Engine est la relation Manager - Ressource. En effet, chaque manager gère un type de ressource et se voit dédié un budget mémoire pour apporter plus de précision au traitement des données.

De la même manière, chaque MovableObject est lié à une Node qui lui permet de se représenter dans l'espace.
Cette Node étant elle même reliée à une autre Node et ainsi de suite jusqu'à la RootNode (origine) contenue dans la classe Root.

Classes

Root

La Root est par définition l'entité englobant le reste des autres classes. Elle sert entre autres aux déclarations d'objets principaux tels que la RootNode et les Resource Managers.
Du point de vue développeur, c'est le point de départ du projet ; un organiseur qui permet de créer les outils pour gérer le tout en détail par la suite.



MovableObject

Le MovableObject est l'entité basique du Chrome Engine. Il sert d'enveloppe à n'importe quel élément mobile d'un rendu ; lequel pourra par la suite être lié à une Node pour pouvoir être affiché.

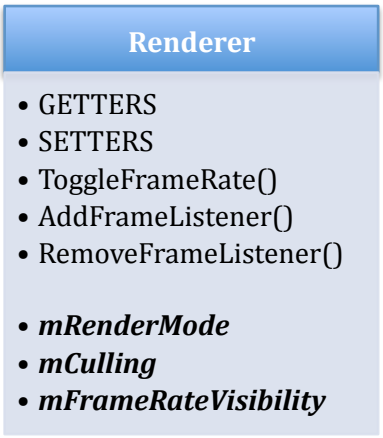
Une LighList contenant les lumières affectant l'objet est mise à jour à chaque requête du client et permet ainsi d'affecter le shader associé.

Node
• GETTERS • SETTERS • scale([x,y,z]) • translate([x,y,z]) • rotate([x,y,z]) • ***mName*** • ***mParent*** • ***mChildren*** • ***mPosition*** • ***mOrientation*** • ***mScale*** • ***mWorldMatrix*** • ***mLocalMatrix*** • ***mListener***

Node

La Node est en quelque sorte une ancre dans un espace 3D.
Elle permet - par le biais d'un parcours d'arbre ascendant – de remonter jusqu'à la Root Node et d'ainsi définir le positionnement de chaque object par rapport à l'origine.

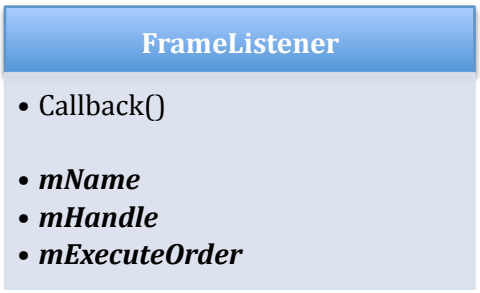
Chaque object de la scène est lié à une Node.



Renderer

Le Renderer s'occupe des fonctions bas niveau telles que le changement du mode de rendu (continu, on demand etc…), l'activation du Culling ou encore la visibilité du Framerate.

Un autre point important est qu'il peut ajouter ou supprimer un FrameListener (voir Doc).



FrameListener

Le FrameListener est une fonction appelée à chaque rendu du moteur qui sera exécutée en prenant un Evenement système comme argument.

Le FrameListener est utile pour mettre en place des traitement post process ou des synchronisations par rapport au framerate (ex: mouvement, horloge, etc…).

Resource

Une ressource est une donnée brute interprétée par le Chrome Engine.

La classe Resource est une classe virtuelle permettant l'interprétation et la gestion de tout type de données par le moteur et le gestionnaire associé (Manager).

Son utilisation va du Mesh à la Texture en passant par les Polices d'écritures ou les fichiers Shaders.

ResourceManager

Le ResourceManager est étroitement lié à la classe Resource.

En effet, chaque Resource doit recevoir un ResourceManager qui permettra au moteur et au développeur de gérer l'instanciation, la suppression ou l'utilisation de ce type de données.

Aussi, le développeur peut assigner un budget mémoire à un manager pour bénéficier de plus de précision dans la gestion des ressources associées.