Chrome Engine

Documentation Utilisateur

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

Description

Chrome Engine est un projet entrant dans le cadre des Epitech Innovative Projects (EIP) pour le cycle Master de l'Epitech Paris sont développement est étendu sur une période de 24 mois (Septembre 2009 – Aout 2011).
Il s'agit moteur de rendu 3D léger et flexible permettant le développement de contenus riches sur le Web.

Les utilisateurs ciblés sont des développeurs habitués aux standards de codage JavaScript et ne possédant pas de connaissances approfondies en matière de programmation graphique.

Cependant, le processus de création d'applications et l'architecture du moteur permettent - par le biais du polymorphisme ou de l'héritage - aux programmeurs avancés de bénéficier d'un large spectre de fonctionnalités.

Bien qu'innovant du point de vue technique, Chrome Engine se base en grande partie sur un autre moteur graphique open source : OGRE 3D.
Cette similarité s'explique par l'envie des développeurs de se baser sur un standard connu et d'offrir un mode de diffusion du contenu simple et efficace; l'univers de la programmation web se basant à 80% sur l'exploitation d'un code déjà existant.

Foire aux Questions (FAQ)

Questions Communes

Que puis-je développer en utilisant Chrome Engine ?
Tout d'abord, Chrome Engine n'est pas un moteur graphique uniquement orienté vers les jeux vidéo. Partant de cela, il peut être utilisé dans tout type de « web app ». Du visionneur de modèles 3D au simulateur, en passant par l'architecture d'une maison ou la configuration de ses meubles. Tout est possible… enfin presque !

Comment ça « enfin presque » ?
Chrome Engine n'est pas l'équivalant d'Unreal Engine ou du Cry Engine. Par définition et puisqu'il utilise votre navigateur ainsi que du JavaScript, il ne possède pas le niveau technique, la rapidité, ni le potentiel d'un moteur écrit entièrement en C++.
Cependant il est facile de reproduire le rendu visuel de Quake, Warcraft 3 ou encore TorchLight.

Je suis débutant dans la 3D, puis-je utiliser Chrome Engine ?
Bien qu'il nécessite quelques connaissance en matière de programmation 3D, Chrome Engine ne demande pas aux développeurs de tutoyer les arcanes des processeurs graphiques.
Nous avons opter pour une conception simple et intuitive du type JQuery ou Moo-Tools. Il vous suffit de consulter les tutoriaux « Premiers Pas » puis la documentions technique une fois que vous aurez terminé ces derniers.

Mais alors, pourquoi avoir choisi O3D/JS plutôt que Flash/Java/C++ ?
Le but principal de Chrome Engine est de s'ouvrir au plus grand nombre et de promouvoir les contenus riches sur Internet. Pour cela il est nécessaire de se baser sur une technologie respectant les standards du Web, en l'occurrence le HTML5.
Tout naturellement, une telle ambition ne peut être acceptée que si les développeurs se joignent au projet. Le support doit donc être Open Source.
JavaScript remplit tout à fait les critères du projet, la seule chose lui faisant défaut étant ses performances.
Fort heureusement, l'utilisation d'O3D (écrit en C++) permet de contourner cette limitation et de bénéficier ainsi d'un gain non négligeable de vitesse d'éxecution.

Puis-je vendre une application utilisant Chrome Engine ?
Oui, Chrome Engine est développé sous la License BSD. La seule contrainte est que vous devez mettre à disposition des développeurs toutes les modifications apportées au code source du projet.
Le JavaScript étant non privatif par définition, vous ne pouvez pas vous soustraire à cette règle (:

Questions techniques

Quel est le langage de Shaders utilisé par O3D ? Pourquoi ?

Chrome Engine s'est aligné sur le modèle de shaders utilisé par O3D, à savoir le standard HLSL2 utilisé par DirectX et développé par NVidia.
Nous aurions pu choisir le modèle GLSL mais la grande hétérogénéité du public de Chrome Engine ne permet pas un support correct du GLSL.
Nous vous renvoyons vers le site de Google O3D pour de plus amples informations sur les problèmes encontrés.

Comment puis-je gérer les lumières ?

Google O3D se base uniquement sur des shaders et non pas sur des lancés – mêmes sommaires – de rayons, il faut passer et gérer manuellement les lumières dans les fichiers shaders.
Chrome Engine offre néanmoins une surcouche à ce niveau dans le LightManager de manière à ce que l'accès aux sources de lumières soit simple et sécurisé.
Nous vous renvoyons vers la documentation technique pour de plsu amples informations.

Puis-je utiliser des contrôles complexes (CTRL-A, CTRL-C, SHIFT-1 etc.) ?

Oui mais pas nativement, les navigateurs ne permettent pas aux plugins d'accéder à ce type de modificateurs pour des raisons de sécurité.
Cependant, le JavaScript lui y a accès. Il existe donc un moyen d'outrepasser cette limite mais cela affectera les performances d'une manière importante car la synchronisation entre la fenêtre O3D et le JavaScript hors de la fenêtre n'est pas naturelle.
Si vous connaissez un moyen « sécurisé » de remédier à ce problème contactez nous et nous remonterons l'information vers Google qui se fera une joie de l'intégrer à O3D.

Chrome Engine fonctionne-t–il dans un environnement émulé ?

Cela dépend, nous avons récemment constaté que Chrome Engine fonctionnait sous VMware sur le Mac OS tant que l'accélération matérielle était activée.
Cependant quelques problèmes font qu'Ubuntu ne supporte actuellement pas O3D et par conséquent, Chrome Engine.
Dans tous les cas, la partie JavaScript du moteur est fonctionnelle, nous attendons donc la prochaine version d'O3D qui devrait corriger cela.

Utilisation

Développeur

Après avoir téléchargé le Framework et inclut la librairie a l'intérieur de la page HTML, le développeur utilise l'API de la manière suivante.

```
//Création d'une Root et initialisation
myRoot = new Root()
myRoot.initialize()

//Création des gestionnaires de ressources
MeshManager = myRoot.getResourceManager("Mesh")
TextureManager = myRoot.getResourceManager("Texture")

//Chargement d'un mesh et d'une texture
MeshManager.load("totomesh", "mesh.o3dtgz")
TextureManager.load("tototexture", "texture.jpg")

//Création de Nodes pour lier le joueur et la caméra
PlayerNode = new Node("playernode", myRoot.getRootNode(), [x,y,z])
CameraNode = new Node("cameranode", PlayerNode, [x+10, y+10, z+10])

//Création d'une entité, d'une lumière ambiante et d'une caméra
Player = new Entity(MeshManager.getResource("totomesh"),
TextureManager.getResource("tototexture"), myRoot.getRootNode())
Light = new Light([x,y,z], [R,G,B], "Ambiant")
Camera = new Camera (CameraNode)

//Lancement du rendu puis shutdown (:
myRoot.startRendering()
....
myRoot.shutdown()
```

Utilisateur (Comportement du plug-in O3D et du navigateur)

Etape 1 - Arrivée du client sur la page demandée
Etape 2 - Vérification de la disponibilité du plug-in

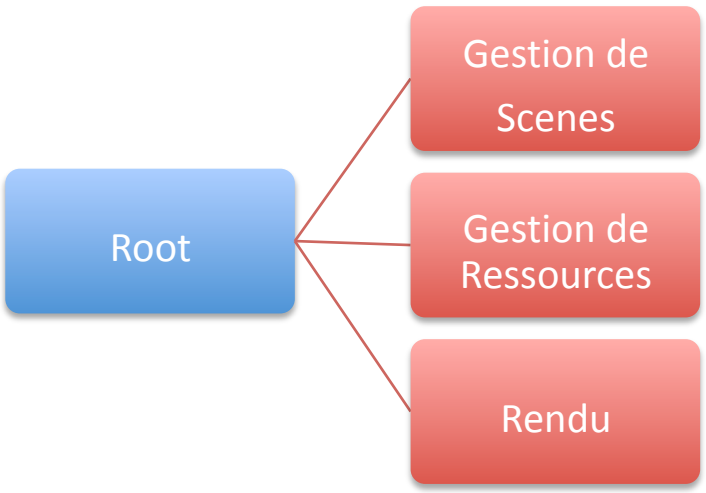
Etape 3A - Plug-in O3D installé
Etape 4A - Initialisation de la fenêtre

Etape 3B - Plug-in O3D indisponible
Etape 4B - Renvoi vers google.com/o3d

Etape 5 - Affichage

Modules

Chrome Engine est divisé en différents modules eux même composés de plusieurs classes et ayant chacun un rôle spécifique. Vous trouverez ci-dessous un diagramme simplifié montrant les parties principales du Chrome Engine.



Racine

La racine est par définir l'entité englobant le reste des autres classes. Elle sert entre autres aux déclarations d'objets principaux tels que les Scene et Resource Managers.
Du point de vue utilisateur, 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.

Gestion de scènes

Cette partie gère l'architecture et la gestion de la scène, la façon dont elle est structurée et la manière dont elle est perçue par la/les caméras.
Elle sert majoritairement d'espace de configuration et de déclaration pour les différentes entités du projet.

Gestion de ressources

Cette partie s'occupe des ressources : le chargement de textures ou d'archives, leur décompression, leur déchargement et leur gestion en mémoire.

Rendu

Le rendu affiche les entités du projet à l'écran, il s'agit de programmation bas niveau gérant le pipeline graphique, le buffer matériel et les états logiques de rendu.
Les composants du gestionnaire de scènes tirent leur fonctions haut niveau de ce composant.

Tutoriaux

Version 1 (o3djs+)

Voir Steering Comitee (.pptx)