Forum: Moteur de jeu GameBlender et alternatives
Topic: hARMful engine
Subject: Re: hARMful engine
PostÃ© par: Bibi09
Contribution le : 9/4/2020 23:39:50

Je voulais te remercier mais ça me semblait être du spam de juste écrire "Merci".

Alors j'en profite pour dire que j'ai fait un changement assez important dans la façon de coder un programme avec le moteur. Je n'avais pas pensé aux répercussions mais ce changement majeur m'a fait monter le niveau de version afin de souligner l'impact (et le côté "incompatible" de l'ancien code).
Je m'attaque donc aux optimisations plutôt qu'au matériaux PBR comme prévu. Je préfère changer les plans car ce changement va invalider tout programme réalisé avec le moteur. Ce serait bête de coder plus de choses et de devoir les modifier ensuite... En particulier, je pensais intégrer officiellement une caméra de type "fly cam" comme dans la dernière vidéo où j'ai bidouillé grossièrement.

On passe donc à la v2.0 pour la prochaine release !

Dans la v1.0, le scene tree est constitué d'entités. Ces entités comportent des composants (mesh, lumière, caméra, matériaux, ...) et une matrice de transformation locale. C'est la transformation qui posait le gros souci !

En effet le souci de cette structure, c'est que pour le rendu et plein d'autres choses (dont l'éditeur graphique du moteur), c'est pas bon. Il y a plein de cas où j'ai besoin d'un accès à la matrice de transformation globale (world). Pour le rendu seul, je me débrouillais pour calculer ça petit à petit.

Premier souci : à chaque frame, je parcours le scene graph en entier et je refais les mêmes calculs y compris pour les objets qui n'ont pas bougé. Ce manque d'optimisation était tout à fait assumé, passant au second plan. Seulement, j'ai besoin aussi de la matrice world en dehors du rendu !!

La nouvelle façon de faire va améliorer tout ça dans un premier temps, avec un gain de performance qui va avec normalement. Impossible à évaluer pour le moment mais je ferai des benchmarks avant/après.
J'irai ensuite plus loin.

La nouvelle structure inverse les rôles. Le scene tree n'est plus constitué d'entités. Il est constitué de transformations locales. Comme dans Unity, une "transform" est un noeud dans un arbre. Elle a donc accès à son parent direct ET à ses enfants.
La transform possède l'entité qui, elle, possède toujours ses composants.

J'aurais très bien pu ne pas modifier le code et laisser la transform accéder à ses parent/enfants via son entité. Ca faisait quand même un gros détour assez bizarre pour quelque chose de pourtant essentiel au bon fonctionnement du code. Je m'explique.
Une entité a besoin de connaitre ses composants, c'est un fait. Par contre, elle n'a pas énormément d'intérêt à savoir si sa fille a un matériau blanc à pois roses ou une lampe verte.
Une transform par contre a vraiment besoin de connaître sa mère et ses filles. La mère pour lui demander sa matrice world, les filles pour leur dire qu'elle a changé.

Là où ça va apporter un gros bénéfice du coup, c'est que quand une transformation locale change dans l'arbre, ça va se répercuter automatiquement sur ses enfants.
Ainsi, demander la matrice world d'un élément de l'arbre sera beaucoup plus simple et rapide.
C'était un changement indispensable pour la suite du projet.