Forum: Moteur de jeu GameBlender et alternatives
Topic: Questions techniques sur le bge.
Subject: Re: Questions techniques sur le bge.

PostÃ© par: Matpi
Contribution le : 19/6/2016 10:39:50

Salut,

(Je me demande si ce n'est pas déjà le cas par défaut, que les animations ne sont jouées que si visibles, mais je ne suis pas sûr... panzergame, si tu passes par ici...)

Les méthodes getScreenPosition et getScreenVect sont plus ou moins l'inverse l'une de l'autre: getScreenPosition prend un vecteur "réel" 3D en argument et rend un vecteur 2D de l'écran*. Bref elle projette le vecteur dans l'espace caméra (alternativement un objet peut être donné en argument, dans ce cas-là c'est sa position qui est utilisée); getScreenVect prend un vecteur 2D de l'écran en argument et rend un vecteur 3D de la scène.

Elles ne sont pas exactement l'inverse l'une de l'autre, toutefois, puisque getScreenVect rend seulement un vecteur de **direction**** (pas de position) et en particulier non lié à la position de la caméra.

Admettons que la caméra soit à la position A et l'objet/le point à la position B (vue dans la caméra aux coordonnées écran (x, y)), alors:

1) getScreenPosition prend OB et rend (x, y)
2) getScreenVect prend (x, y) et rend normalize(OB - OA)

Quant à getScreenRay, elle fonctionne avec une "incertitude" en plus, c'est-à-dire qu'elle cherche l'objet situé, vu de la caméra, "sous" la position (x, y). C'est un peu comme un rayCast, mais en utilisant les coordonnées caméra en entrée. Typiquement on utilise cela pour récupérer un objet sous un clic souris dans le viewport (façon Mouse Over).

Les paramètres dist et property permette d'affiner la recherche en spécifiant respectivement une distance limite à la caméra de l'objet (plutôt que l'infini) et une Game Property que l'objet doit contenir pour être détecté.

Dans ton cas - et en imaginant que cette fonctionnalité ne soit pas présente par défaut dans le BGE, ce dont je ne suis pas sûr - il faut plus ou moins:

1) boucler sur tous les objets susceptibles d'avoir une animation

2) prendre leurs coordonnées écran avec getScreenPosition et s'assurer que les deux dimensions sont chacune dans l'intervalle [0.0, 1.0]

3) en tirer des conséquences sur l'animation ou non.

Je vois les problèmes suivants:

- ça risque d'être lourd, tant le bouclage que la gestion des anims en conséquence
- une partie des objets risque de "déborder" sur l'écran même si leur centre défini est à l'extérieur -> utiliser la Bounding Box??
- comme déjà dit, pas sûr que ce ne soit pas déjà implémenté en dur.

Tu peux également regarder du côté de l'Occlusion Culling, ça peut peut-être aider.

* les vecteurs 2D d'écran sont définis de façon à avoir le point (0.0, 0.0) en haut à gauche et (1.0, 1.0) en bas à droite - donc le centre du viewport à (0.5, 0.5).

** c'est normal, la fonction ne sait pas à quelle distance on doit regarder, donc par défaut le vecteur est **normalisé** - longueur 1.0; on peut aussi voir ça comme une conséquence du manque d'informations: on donne deux arguments scalaires en entrée - donc deux degrés de liberté - or on ressort un 3-vecteur, donc trois scalaires; par conservation de l'information, il manque un degré de liberté et les trois scalaires ne sont donc pas indépendants.

EDIT: Quand on parle du loup...

Salut panzer'!

EDIT 2: oui, j'ai vraiment mis plus d'une heure à rédiger cette réponse...