La technique d'instanciation

L'instanciation est une technique permettant de dessiner plusieurs copies du même objet. Cette technique sera intéressante dans des scènes contenant des objets similaires : une forêt d'arbre, le feuillage d'un arbre, une foule, l'herbe d'une pelouse.

L'instanciation

Lorsque l'on souhaite afficher de nombreux objets, la première approche pourrait être de faire plusieurs appels à glDrawArrays et glDrawElements. Le nombre d'appels aux fonctions de dessin peut cependant diminuer fortement les performances. Une optimisation possible est d'utiliser glMultiDrawArrays ou glMultiDrawElements, qui prennent des tableaux de données et permettent donc de faire qu'un seul appel. Cependant, il est nécessaire d'envoyer les données correspondantes à tous les objets que l'on souhaite créer, ce qui peut saturer la bande passante du GPU.

```
// Extension GL_ARB_draw_instanced pour DrawArraysInstanced()
et glDrawElementsInstanced()
// Extension GL_ARB_instanced_arrays pour
VertexAttribDivisor()

// Pour dessiner plusieurs objets
void glMultiDrawArrays(GLenum mode, GLint *first, GLsizei *
count, GLsizei primcount);
void glMultiDrawElements(GLenum mode, GLsizei *count, GLenum
type, GLvoid **indices, GLsizei primcount);
```

L'instanciation proprement dite permet de créer plusieurs instances d'un objet. On envoie les données que pour un seul objet, ce qui réduit la bande passante.

Pour personnaliser chaque instance, on fournit un tableau de données (Instanced Array), chaque élément du tableau correspondant à une instance. On pourra passer ainsi modifier la position des instances, les

paramètres de texture pour personnaliser les instances, ajouter du bruit, etc. Le paramètre divisor permet de spécifier quand on passe à l'élément suivant dans un buffer. Pour divisor = 0, chaque vertex lit un nouvel élément dans le buffer. Avec divisor = 1, chaque instance lit un nouvel élément. Avec divisor = 2, un nouvel élément est lu toutes les 2 instances, etc.

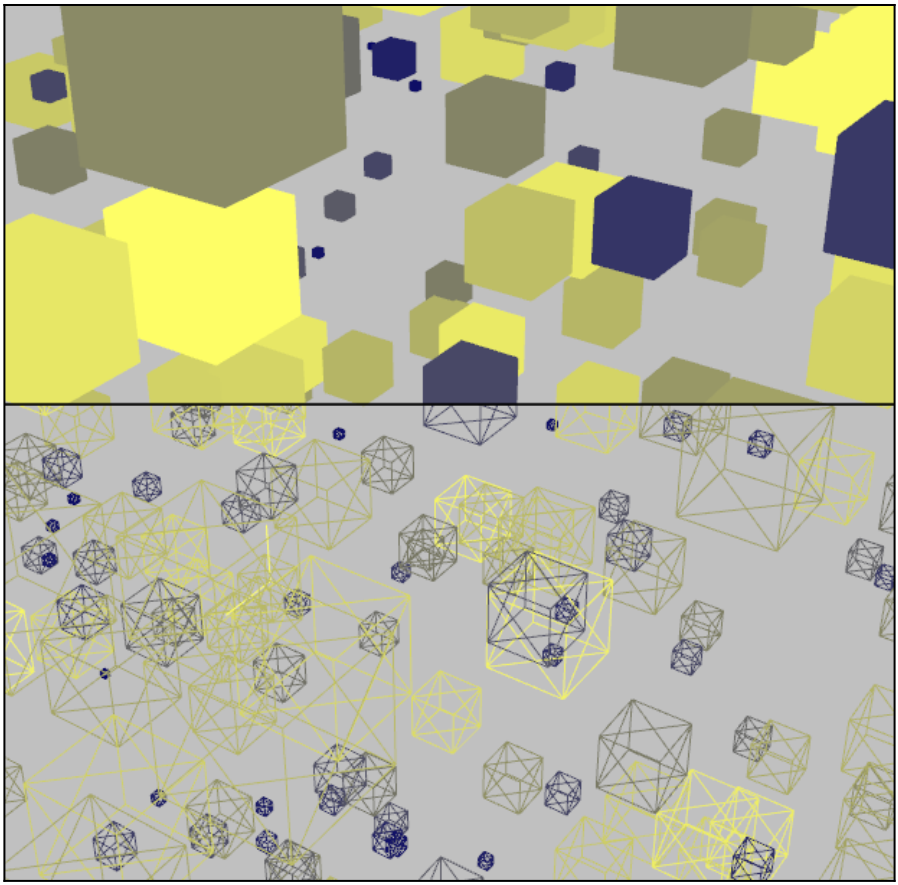
```
// Création de plusieurs instances d'un objet
void glDrawArraysInstanced(GLenum mode, GLint first, GLsizei
count, GLsizei primcount);
void glDrawElementsInstanced(GLenum mode, GLsizei count,
GLenum type, const void *indices,
    GLsizei primcount);

// Pour connaître l'identifiant de l'instance (GLSL)
gl_InstanceID; // entre 0 et primcount-1

// Définir le mode de lecture des buffers
glVertexAttribDivisor(location, divisor);
```

Un exemple simple

Le code d'exemple affiche des cubes en utilisant glDrawElementsInstanced. Le curseur permet de sélectionner le nombre d'instances à créer. Le bouton permet de basculer entre la vue des cubes pleins et en fil de fer. Les FPS et le nombre d'instances sont indiqués dans le titre de la fenêtre. Je peux obtenir plus de 10 000 instances en même temps à 30 FPS (soit plus de 120 000 triangles) sur mon portable équipé d'une 8600M GT.

Pour ceux qui testeraient avec du matériel plus puissant, il est possible d'augmenter le nombre d'instance maximal en modifiant le facteur multiplicateur à la ligne 127 du fichier glwidget.cpp ou la valeur maximale du QSlider à la ligne 22 du fichier mainwindow.cpp. J'ai pu ainsi créer plus de 400 000 instances (4,8 millions de triangles) avant d'arriver à 30 FPS sur une tour équipée d'une GTX 460.

Exemple d'affichage des cubes pleins (en haut) et en fil de fer (en bas)



Télécharger le projet d'exemple

À lire

- gpu gems chap 7
- Instancing in OpenGL

OpenGL