Introduction aux geometry shaders

A la demande de LittleWhite, mon premier bloc technique est consacré à l'utilisation des geometry shader. Il n'y a rien de tres compliqué donc je présente un exemple simple apres quelques rappels. Pour ceux qui ne connaissent pas du tout les geometry shaders, je ferais un article plus detaille sur ce sujet. Pour faire au plus simple, j'utilise Qt et en particulier QShaderProgram, qui gère les geometry shaders depuis Qt 4.7.

Quelques rappels

Les geometry shaders sont optionnels et s'intercalent entre les vertex shaders et les fragment shaders. Ils travaillent sur des primitives (triangle, ligne, points, avec un maximum de 6 vertices) et permettent de générer d'autres primitives. Ils sont disponibles dans le Core depuis OpenGL 3.2 / GLSL 1.5 ou avant avec l'extension GL_ARB_geometry_shader4 (en ajoutant le code #extension GL_EXT_geometry_shader4 : enable dans le geometry shader).

```
// Rechercher la présence de GL_EXT_geometry_shader4 pour
// tester si les geometry shaderssont pris en charge
glGetString(GL_EXTENSIONS);
glGetStringi(GL_EXTENSIONS, i); // GL 3.2

// Créer un geometry shader
glCreateShader(GL_GEOMETRY_SHADER);
QGLShader shader(QGLShader::Geometry); // Qt 4.7

// Gérer les informations de position
gl_PositionIn[i]; // position du vertex i en entrée
gl_Position; // position du nouveau vertex émis
gl_in[i].gl_Position; // GLSL > 330
gl_in.length(); // GLSL > 330, nombre de vertices en entrée

// Générer des vertices et primitives (GLSL)
EmitVertex();
```

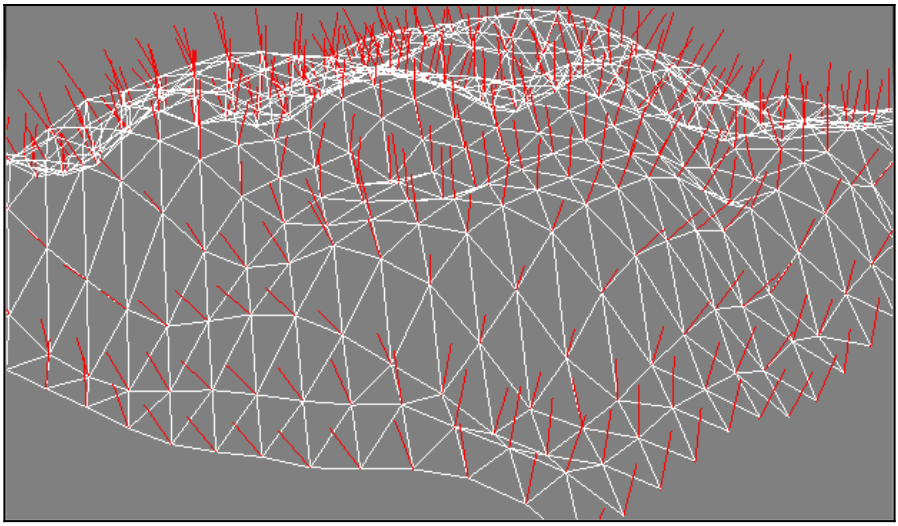
```
EndPrimitive();

// Spécifier le type de primitives en entrée et sortie
// (GLSL)
layout (triangles) in; // points, lines, triangles,
lines_adjacency ou triangles_adjacency
layout (triangle_strip) out; // points, line_strip ou
triangle_strip
layout (max_vertices = 3) out; // maximum =
GL_MAX_GEOMETRY_OUTPUT_VERTICES

// (Qt)
program_shader.setGeometryInputType(GL_TRIANGLES);
program_shader.setGeometryOutputType(GL_LINE_STRIP);
program_shader.setGeometryOutputVertexCount(6);
```

Lors de l'appel à glDrawArray ou glDrawElement, il faut que le type spécifié corresponde au type d'entrée du geometry shader. Les correspondances autorisées sont :

- points : GL_POINTS
- lines : GL_LINES, GL_LINE_LOOP ou GL_LINE_STRIP
- triangles : GL_TRIANGLES, GL_TRIANGLE_FAN ou GL_TRIANGLE_STRIP
- lines_adjacency : GL_LINES_ADJACENCY
- triangles_adjacency : GL_TRIANGLES_ADJACENCY

Les nouveaux types ADJACENCY (GL_LINES_ADJACENCY, GL_LINE_STRIP_ADJACENCY, GL_TRIANGLES_ADJACENCY et GL_TRIANGLE_STRIP_ADJACENCY) permettent d'accéder aux vertices des primitives voisines de la primitive en cours de traitement.

Exemple d'affichage des vecteurs normaux



Un exemple simple

L'exemple choisit est tres tres classique. Il permet d'afficher le mailage d'un mesh, les vecteurs normaux des faces et de calculer les vecteurs normaux de chaque face (flat mode). Cet exemple me permet également d'introduire quelques classes simples, pour lire un fichier .obj (version simplifiee), calculer les vecteurs normaux, initialiser un contexte GL avec Qt.

Pour l'affichage des vecteurs normaux et des triangles, on a besoin des positions et des normales, que l'on passent donc aux shaders en attributs.

Pour afficher les vecteurs normaux dans le geometry shader, on prend un seul vertex en entree (input = points) et l'on retourne une ligne (output = lignes, size = 2), dont le premier point est la position du vertex et le second point est la position du vertex auquel on ajoute le vecteur normal (multiplie par une constante, passee comme uniform, pour modifier la longueur du vecteur)

Pour afficher les triangles, on recupere un triangle en entree (input = triangles) et l'on retourne 3 lignes (output = lignes, size = 6). Chaque points correspond a la position d'un vertex decale legerement par le

vecteur normal (pour que les lignes soient bien visibles et ne soient pas cachees par les triangles).

Télécharger le projet d'exemple

À lire

- Shrink, explosion, subdivision, silhouette, hedgehog
- Explosion, cheveux
- Instanciation et culling
- Instanciation et culling

C++, OpenGL, Qt