

# M2-Images

Rendu Temps Réel - OpenGL 3 et transform feedback

J.C. Iehl

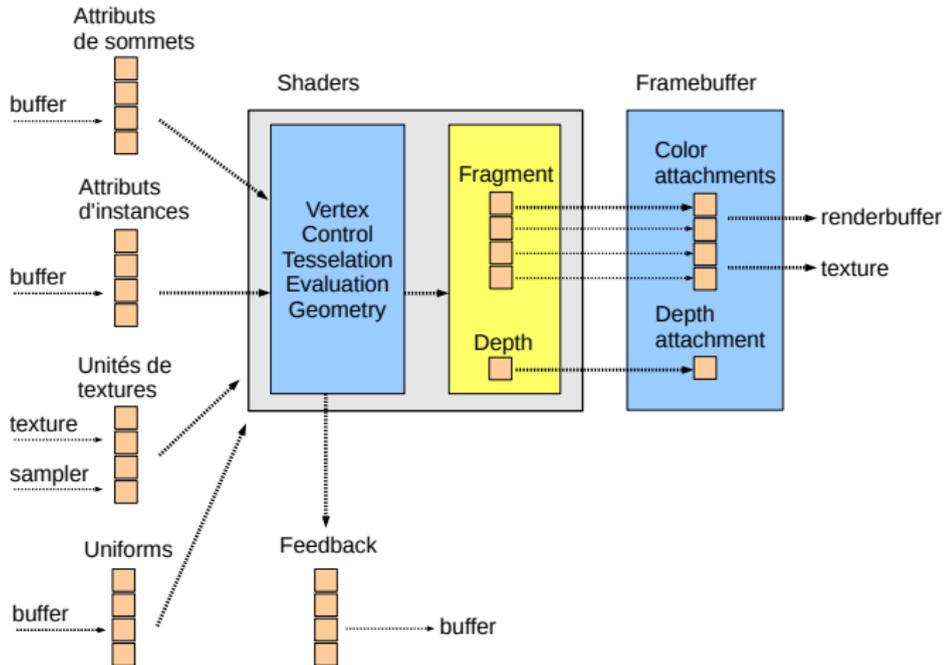
November 7, 2012

# Résumé des épisodes précédents

résumé :

- ▶ création de buffers,
- ▶ création de maillages indexés ou non,
- ▶ affichage de maillages,
- ▶ affichage de plusieurs maillages,
- ▶ vertex, géométrie et fragment shaders,
- ▶ textures, framebuffer,
- ▶ notions de traitement en plusieurs passes ...

# Résumé de l'api opengl 3



# Transform Feedback

même idée que pour les framebuffer objects :

- ▶ permet de stocker les résultats du geometry shader dans un/des buffers,
- ▶ et de faire plusieurs passes de transformations...
- ▶ ou autre chose...

# Transform feedback

## utilisation :

- ▶ déclarer dans le shader les `varyings` à enregistrer (comme d'habitude...),
- ▶ sélectionner le mode de remplissage des buffers,
- ▶ compiler / re-linker le shader,
- ▶ créer et activer les buffers pour stocker les `varyings`,
- ▶ dessiner des primitives...

# Transform feedback et shader

uniquement les varyings out déclarés dans le vertex/geometry shader peuvent être enregistrés.

```
#version 330    // vertex shader core profile

uniform mat4.mvpMatrix;

in vec4 position;
out vec4 vertex_position;

void main( void )
{
    gl_Position=.mvpMatrix * position;

    vertex_position= position + vec4(10.f, 10.f, -100.f, 0.f);
    // deplace le sommet et enregistre sa position ...
}
```

# Préparer le transform feedback

pour l'application :

- ▶ `glTransformFeedbackVaryings()`,
- ▶ permet de déclarer quels varyings enregistrer,
- ▶ et de quelle manière :
- ▶ `GL_SEPARATE_ATTRIBS` : dans des buffers séparés,
- ▶ `GL_INTERLEAVED_ATTRIBS` : dans un seul buffer.
- ▶ activer le/les buffers :
- ▶ `glBindBufferBase(GL_TRANSFORM_FEEDBACK_BUFFER, index, buffer)`

créer et dimensionner le bon nombre de buffers.

# Utiliser le transform feedback

utiliser :

- ▶ `glBeginTransformFeedback( feedback_primitives ),`
- ▶ `glDrawXXX( draw_primitives ),`
- ▶ `glEndTransformFeedback( ).`

`feedback_primitives` doit correspondre aux primitives émises par le geometry shader, ou `GL_POINTS`, s'il n'y a pas de geometry shader (vertex shader).

## et alors ?

- ▶ combien de données sont effectivement écrites dans le/les buffers ?
- ▶ et si le stockage est trop petit ?
- ▶ et si dessiner le résultat n'a pas de sens ?

recupérer la quantité de données correctement écrites par le feedback.

# Requêtes

- ▶ mécanisme générique d'OpenGL pour obtenir des informations sur l'exécution d'une opération,
- ▶ primitives produites, primitives écrites par le feedback, temps d'exécution, visibilité, etc.

# Requêtes

- ▶ création : `glGenQuery( )`,
- ▶ utilisation : `glBeginQuery( )`, + `glEndQuery( )`,
- ▶ résultat : `glGetQueryObject( )`.

```
GLuint feedback;
glGenQuery(1, &feedback);

glBeginQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN, feedback);
glBeginTransformFeedback( ... );
glDrawXXX( ... );
glEndTransformFeedback();
glEndQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN);

GLint n;
glGetQueryObjectiv(feedback, GL_QUERY_RESULT, &n);
```

## et la même chose en plus simple ?

### OpenGL 4 :

- ▶ transform feedback object :
- ▶ garde automatiquement tout l'état nécessaire, y compris le nombre d'éléments écrits,
- ▶ +
- ▶ `glDrawTransformFeedback( primitives, transform_feedback_object )`,
- ▶ équivalent à :
- ▶ `glDrawArrays( primitives, 0, n )` avec `n` le résultat de la requête...

## et la même chose en plus simple ?

- ▶ `glGenTransformFeedbacks( )`
- ▶ `glBindTransformFeedback( )`
- ▶ `glBeginTransformFeedback( )`
- ▶ `glDrawXXX( )`
- ▶ `glEndTransformFeedback( )`
- ▶ `glDrawTransformFeedback( )`

et c'est mieux ?

et si dessiner le résultat n'a pas de sens ?

- ▶ désactiver la fragmentation...
- ▶ `glEnable(GL_RASTERIZER_DISCARD);`