Introducció

En aquests exercicis us demanen que implementeu plugins (en C++) pel visualitzador bàsic de l'assignatura. Alguns exercicis requereixen escriure shaders (en GLSL).

Farem servir aquesta nomenclatura:

- **Effect plugin:** plugin que implementa els mètodes **preFrame** i/o **postFrame** (a banda d'altres mètodes que pugueu necessitar). Teniu un exemple a **plugins/effectcrt.**

- **Draw plugin**: plugin que implementa el mètode **drawScene** (a banda d'altres mètodes que pugueu necessitar). Teniu un exemple a **plugins/drawvbong.**

- **Action plugin:** plugin que implementa mètodes d'entrada com ara **keyPressEvent** (a banda d'altres mètodes que pugueu necessitar). Teniu un exemple a **plugins/navigatedefault**

- **Render plugin:** plugin que implementa el mètode **paintGL**, (a banda d'altres mètodes que pugueu necessitar). Teniu un exemple a **plugins/renderdefault.** Per desenvolupar shaders a la primera meitat del curs heu fet servir el plugin que podeu trobar a **plugin/shaderLoader/**

- VS, GS i FS fan referència a vertex shader, geometry shader i fragment shader, resp.

Warming up

Desplega les fonts del visualitzador (viewer) en un directori teu i prova construir els binaris, seguint les instruccions que us hem donat. Executa el viewer i prova de carregar algun model (en format .obj).

ModelInfo (1)

Escriu un **Effect Plugin** que escrigui (**postFrame**) la següent informació sobre l'escena: número total d'objectes carregats, número total de polígons, número total de vèrtexs, i el percentatge de polígons que són triangles. Feu una implementació que tingui un impacte negligible en el frame rate. Teniu un exemple de recorregut de l'escena a **plugins/drawvbong**

En aquesta versió només cal mostrar aquests valors pel canal de sortida estàndard (cout...).

Animate vertices plugin

Escriu un **effect plugin** que activi un VS per tal d'obtenir el mateix efecte de l'exercici **Animate Vertices (1).** Podeu mirar **plugins/effectcrt** com a exemple.

El mètode **onPluginLoad** haurà de carregar, compilar i muntar el shader. El mètode **preFrame**() els haurà d'activar i donar un valor apropiat a **l'uniform float time** (podeu feu servir timers de Qt) i a la resta d'uniforms que feu servir; el mètode **postFrame**() els haurà de desactivar.

ModelInfo (2)

Escriu un **Effect Plugin** que escrigui la mateixa informació que ModelInfo (1), però en aquesta versió cal mostrar aquestes dades per pantalla. Podeu mirar **plugins/showhelp** per veure l'ús de QImage, QPainter, QFont per dibuixar text en la finestra OpenGL.

Framerate

Escriu un **Effect Plugin** que mostri el *Frame Rate* actual de l'aplicació. Pots fer servir un `QElapsedTimer` per a calcular el temps usat en dibuixar els *frames*. Una alternativa és incrementar un comptador de frames a preFrame o postFrame, i crear un QTimer que emeti un signal cada segon, connectat a un slot que copiï el valor d'aquest comptador a la variable que conté els fps.

Podeu mostrar el resultat (per exemple, cada segon) pel cout.

Il·luminació per fragment amb shaders

Escriu un **effect plugin** que activi un VS i un FS per tal de tenir il·luminació de Phong per fragment. El mètode onPluginLoad haurà de carregar, compilar i muntar els shaders. El mètode preFrame() els haurà d'activar, i el mètode postFrame() els haurà de desactivar.

Recordeu enviar tots els uniforms que usin els shaders.

Show-degree

Escriu un **effect plugin** que escrigui el grau mig dels vèrtexs de l'escena (podeu assumir que l'escena contindrà **un únic objecte**). El grau d'un vèrtex és, simplement, el número de cares que incideixen en el vèrtex (el número de cares que fan servir el vèrtex). Per exemple, en un cub de sis quads, els vèrtexs tenen grau 3.

El mètode **onPluginLoad**() haurà de calcular el grau mig del primer objecte de l'escena. El mètode **postFrame**() l'haurà de mostrar a la finestra OpenGL.

Aquí teniu un exemple del resultat esperat amb els models boid.obj i glass.obj.