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:

- **draw plugin**: cal que desenvolupeu un plugin que implementi la interfície **DrawInterface**. Teniu un exemple a plugins/draw-immediate.

- **render plugin** -> cal que desenvolupeu un plugin que implementi la interfície **RenderInterface**. Teniu un exemple a plugins/render-default

- **action plugin** -> cal que desenvolupeu un plugin que implementi la interfície **ActionInterface**. Teniu un exemple a plugins/navigate-default

- **effect plugin** -> cal que desenvolupeu un plugin que implementi la interfície **EffectInterface**. Teniu un exemple a plugins/effect-alphablend

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

Pintat amb Vertex Arrays (versió flat)

Escriu un **draw plugin** que pinti els objectes de l'escena amb Vertex Arrays (VAs). Per motius d'eficiència, els arrays (cada objecte tindrà els seus arrays) s'han de crear fóra del mètode drawScene.

Haureu d'implementar onPluginLoad() perquè construeixi els arrays dels objectes que conté l'escena en el moment en què es carrega el plugin, i onObjectAdd() perquè faci el mateix per cada nou objecte que s'afegeixi posteriorment a l'escena. El mètode drawScene() només tindrà les crides OpenGL per pintar els VAs.

En aquesta versió ens interessa tenir normals "per-corner"; un corner és una parella (v, c) on v és un vèrtex de l'objecte i c és una cara de l'objecte que incideix a v. Els VAs tindran, per cada corner (v,c) de l'objecte, les coordenades del vèrtex v i les components de la normal de la cara c.

Pintat amb Vertex Arrays (versió smooth)

Escriu un **draw plugin** que pinti els objectes de l'escena amb Vertex Arrays com a l'exercici anterior.

En aquesta versió però, ens interessa tenir normals "per-vèrtex". Els VAs tindran, per cada vèrtex v de l'objecte, les coordenades de v i el promig (normalitzat) de les normals de les cares que incideixen al v.

Pintat amb Vertex Arrays (versió avançada)

Escriu un **draw plugin** que pinti els objectes de l'escena amb Vertex Arrays com a l'exercici anterior.

En aquesta versió però, ens interessa tenir normals per-corner o "per-vèrtex" depenen de si considerem que l'aresta és suau o està prou definida.

Per exemple, en un cilindre aproximat amb cares planes, ens interessa que les cares laterals facin servir vèrtexs amb normals suavitzades, i que les tapes facin servir vèrtexs amb la normal de la cara. Observeu que això implica replicar alguns vèrtexs al VA.

Pintat amb Vertex Buffer Objects (versió flat)

Escriu un **draw plugin** que pinti els objectes de l'escena amb Vertex Buffer Objects, amb normals no suavitzades.

Vigileu de fer el mètode drawScene() el més eficient possible.

Pintat amb Vertex Buffer Objects (versió smooth)

Escriu un **draw plugin** que pinti els objectes de l'escena amb Vertex Buffer Objects, amb normals suavitzades.

Vigileu de fer el mètode drawScene() el més eficient possible.

Pintat amb Vertex Buffer Objects (avançada)

Escriu un **draw plugin** que pinti els objectes de l'escena amb Vertex Buffer Objects, amb normals suavitzades o no segons l'aresta.

Vigileu de fer el mètode drawScene() el més eficient possible.

Il·luminació per fragment amb shaders

Escriu un **effect plugin** que activi un VS i un FS per tal de tenir il·luminació Blinn-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.

Caldrà que feu servir l'API d'OpenGL per carregar els shaders. Opcionalment podeu fer servir les classes QGLShader i QGLShaderProgram, que faciliten la càrrega de shaders des de fitxers, http://doc.qt.digia.com/qt/qglshaderprogram.html

Plugin de selecció d'objectes

Escriu un **action plugin** per tal que quan l'usuari faci clic amb el mouse (per exemple, LMB + Ctrl), es seleccioni l'objecte visible més proper a l'observador que estigui sota el cursor (si n'hi ha cap, és clar).

Per tal d'aconseguir això, haureu d'implementar el mèdode mouseReleaseEvent(). Aquest mètode haurà de:

(a) pintar l'escena assegurant-se que cada objecte es pinta amb un color únic que permeti identificar l'objecte (i diferent del color de fons);
(b) llegir el color del buffer de color sota la posició del cursor,
(c) obtenir l'identificador de l'objecte corresponent i, si no és color de fons, establir l'objecte seleccionat amb el mètode setSelectedObject de la classe Scene, i
(d) cridar a updateGL per tal que es repinti l'escena.

Si l'usuari fa Ctrl-clic on no hi ha cap objecte, cal posar l'objecte seleccionat a -1.

Per tal de provar el funcionament correcte d'aquest plugin, cal combinar-lo amb el plugin de l'exercici anterior que indiqui d'alguna manera l'objecte seleccionat.

La implementació de la selecció haurà d'estar basada en la lectura del buffer de color (no feu servir el mode GL_SELECT d'OpenGL).

Podeu suposar que l'escena tindrà com a màxim 255 objectes. Nota: podeu afegir més objectes a la vostra escena amb la tecla 'L'.

Resaltat de l'objecte seleccionat

Escriu un **effect plugin** que resalti d'alguna manera l'objecte seleccionat, si n'hi ha cap (per exemple, pintant la seva capsa englobant en filferros). Caldrà que ho feu re-implementant el mètode postFrame().

L'index de l'objecte seleccionat es pot obtenir amb el mètode selectedObject() de la classe Scene (vigileu que un valor -1 indica que no hi ha cap objecte seleccionat). Per defecte, l'objecte seleccionat és el primer (índex 0), si s'ha carregat algun objecte.

Reflexió bàsica

Escriu un **effect plugin** que activi un **geometry shader** que mostri les cares en la seva posició real, i en la seva posició reflectida respecte el pla horitzontal Y=*yMin*, on *yMin* és la coordenada *y* mínima de la capsa contenidora de l'escena (en *world space*). Això vol dir que un triangle amb vèrtexs:

$$(x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2)$$

s'haurà de pintar també com el triangle

$$(x_2, 2yMin\text{-}y_2, z_2), (x_1, 2yMin\text{-}y_1, z_1), (x_0, 2yMin\text{-}y_0, z_0).$$

Degut a les limitacions de les GPUs del laboratori, feu que el VS calculi la il·luminació per vèrtex, de forma que el GS només enviï color i posició dels vèrtexs.

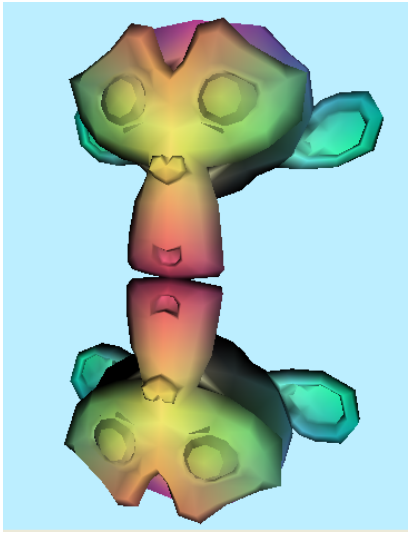

Figura 1. Objecte reflectit respecte el pla de la seva base

Framerate

Escriu un **Effect Plugin** que sobreescrigui en la cantonada superior dreta de l'àrea de dibuix el *Frame Rate* actual de l'aplicació. Pots fer servir un `QElapsedTimer` per a calcular el temps usat en dibuixar els *frames*, i el mètode `renderText` del `QGLWidget` per a pintar el resultat. Proveu-lo mesurant els temps dels diferents **Draw Plugins** que heu escrit.