

Interacció i disseny d'interfícies

Examen OpenGL 2022-23 / Q2

Dept. de Ciències de la Computació

Professors INDI

1 Instruccions

1. Podeu usar el codi que heu elaborat en les classes de laboratori i que tingueu al vostre compte, però **sols el codi que hagueu generat vosaltres**; no podeu fer servir codi que altres estudiants hagin compartit amb vosaltres (ni que hagueu compartit amb d'altres estudiants). Altrament correu el risc que es pugui considerar el vostre examen com una còpia.
2. Partireu del codi que teniu a `examen.tgz` (adjunt a aquesta tasca). Heu de desplegar aquest arxiu en un directori vostre. Us crearà un subdirectori `examen` on tindreu tots els fitxers amb els que heu de treballar. Els exercicis que es demanen només requereixen canvis a la classe `MyGLWidget`, als shaders i al fitxer `MyForm.ui` usant el `designer`. **No cal modificar cap altre fitxer dels que us donem.**
3. Per a fer el lliurament heu de generar un arxiu TGZ que inclogui tot el codi del vostre examen que es digui `INDI_ex_opengl_DNI.tgz`, on substituïreu `DNI` pel vostre DNI incloent la lletra. Per exemple, l'estudiant amb dni `11111111H` (des d'una terminal en la que s'ha col·locat dins del directori `examen`):

```
make distclean
tar cvzf INDI_ex_opengl_11111111H.tgz *
```

És important el `'make distclean'` per a esborrar els arxius binaris generats; que el DNI sigui el correcte (el vostre); i que hi hagi el sufix `.tgz`.

4. **Si el vostre lliurament no segueix el format indicat, està comprimit usant una altra eina, no compila o dóna error d'execució, l'avaluació serà un 0 sense excepció.**
5. Un cop fet això, al vostre directori `examen` tindreu l'arxiu `INDI_ex_opengl_DNI.tgz` que és el que heu de lliurar. **Feu la comprovació**, descomprimint aquest arxiu **en un directori completament buit**, que el codi que lliureu compila (fent `qmake`; `make`) i executa correctament.
6. Finalment, lliureu el fitxer a <https://atenea.upc.edu> mitjançant la tasca "Prova OpenGL".

2 Enunciat

El codi que proporcionem crea i visualitza una escena d'un joc tipus *endless running* formada per:

- una roda formada per cubs de diferent mida que roda constantment (el codi per a que la roda giri constantment està ja inclòs en l'esquelet). Aquesta roda té el seu centre a l'origen de coordenades.
- un fantasma que medeix 10 en les X del model (mida que fa d'orella a orella) i que està situat amb el centre de la base de la seva capsula contenidora a l'origen de coordenades i mirant en direcció Z+.

Els paràmetres inicials de la càmera i de l'escena són completament arbitraris, la `viewMatrix` està mal inicialitzada i només es pot modificar interactivament l'angle ψ . La imatge de l'arxiu `EscIni.png` mostra la visualització inicial.

Hi ha un mètode `creaBuffers` per als models del Fantasma i del Cub. Aquests mètodes tenen inicialitzades totes les dades de material i normals necessàries per poder implementar el càlcul de la il·luminació. També proporcionem les rutines `Ambient`, `Difus` i `Especular` que es troben al Fragment Shader.

Observació: Analitza el codi donat abans d'implementar els exercicis demanats.

En la valoració de l'exercici 6 tindrà molta importància el disseny i la usabilitat de la interfície.

1. (1.5 punts) Modifica l'escena per a que hi hagui tres rodes enlloc d'una (heu d'incloure les dues que falten). La configuració dels cubs que forma cada roda es troba a la matriu `alcades`, ja inicialitzada amb els valors que toca a l'arxiu `MyGLWidget.h`. La roda del mig és la que l'esquelet ja pinta, i correspon a la configuració de cubs que es troba a la fila 1 d'aquesta matriu.

Per a la primera de les dues rodes que falten usarem la configuració de cubs de la fila 0 i aquesta roda estarà desplaçada -2 unitats en Z respecte de la roda del mig (fixa't en els paràmetres del mètode `modelTransformCub`).

La segona de les rodes que falten usará la configuració de cubs de la fila 2 de la matriu `alcades` i estarà desplaçada +2 unitats en Z respecte de la roda del mig.

Nota: Fixa't que la construcció de la roda a partir de la configuració dels cubs ja la tens implementada en l'esquelet quan es pinta la roda del mig en el mètode `paintGL()`.

També has de modificar la mida i posició del fantasma per a què faci mida 2 en les X del model (mida que fa d'orella a orella) i que estigui situat amb el centre de la base de la seva capsula contenidora inicialment al punt (0, 21, 0) i mirant en direcció X+.

2. (1.5 punts) Implementa correctament el càlcul de la càmera (`viewMatrix` i `projectMatrix`) per tenir una càmera perspectiva en 3a persona de manera que l'escena es vegi en tot moment sencera, centrada, sense deformació i ocupant el màxim del viewport, encara que l'usuari modifiqui la finestra gràfica.


Nota: Per al càlcul de la capsula contenidora tingues en compte que el radi de l'interior de la roda és 20 i que la màxima alçada que pot tenir un cub és 3, per tant hauries de comptar la capsula tenint en compte la roda més els cubs més alts tant en X com en Y. Per a les dimensions en Z pots comptar que les tres rodes juntes t'ocupen un total de 6 (2 per cada roda).



Cal afegir que es pugui fer també rotació de l'angle θ , de manera que quan l'usuari mou el ratolí de baix a dalt en el viewport, l'angle θ es decrementa i per tant la càmera es mou cap a baix respecte l'escena. Quan l'usuari mou el ratolí de dalt a baix en el viewport, l'angle θ s'incrementa. Els valors inicials dels angles han de ser: $\psi = 0$ i $\theta = -1.0$ (en radians). Una imatge de la solució als exercicis 1 i 2 la pots veure a l'arxiu `EscEx1-2.png`.



3. (2 punts) Modifica adientment el **Vertex Shader** i el **Fragment Shader** per afegir a l'escena el càlcul d'il·luminació usant el model d'il·luminació de Phong calculat al **Fragment Shader**. El focus de llum ha de ser blanc i de càmera, situat exactament a la posició de la càmera.

Modifica també el material del model del cub (mètode `iniMaterialCub()`) per a què passi a ser de color blau i brillant.


4. (1.5 punts) Afegeix una segona càmera perspectiva, Càmera-2, que estigui situada darrera del fantasma, a la posició (-5, 25, 0) i que miri cap al punt (0, 23, 0). L'angle d'obertura d'aquesta càmera serà fix de 60 graus, i els plans de retallat han de permetre veure tot el que està davant d'aquesta càmera. Aquesta càmera no permet interacció amb el ratolí.

Mitjançant l'ús de la tecla  (`Qt::Key_C`) l'usuari ha de poder canviar de la Càmera-1 a la Càmera-2 i a la inversa.

5. (1 punt) Afegeix un moviment per al fantasma per a què pugui saltar d'una roda a la següent mitjançant les tecles  (`Qt::Key_Left`) i  (`Qt::Key_Right`).

- Utilitzant la tecla  (`Qt::Key_Left`) el fantasma es mourà cap a la roda que té a la seva esquerra, si n'hi ha cap. Si no hi ha més rodes a la seva esquerra no es mourà.
- Utilitzant la tecla  (`Qt::Key_Right`) el fantasma es mourà cap a la roda que té a la seva dreta, si n'hi ha cap. Si no hi ha més rodes a la seva dreta no es mourà.

6. (1.5 punts) Afegeix a la interfície:

- a) Un element d'interfície que permeti decidir quina càmera volem tenir activada en cada moment. Aquest element ha d'estar coordinat amb l'efecte de la tecla  (`Qt::Key_C`).

b) Un o més elements d'interfície que permetin aturar i tornar a engegar el moviment de les rodes. Per fer que les rodes es puguin aturar i engegar mira i utilitza els slots `atura()` i `engega()` ja implementats en l'esquelet.

7. (1 punt) Afegeix la possibilitat de fer un “reset” (reinici) de tot el comportament de l'aplicació:

- Escena tal i com està descrita en els exercicis 1 i 2 (càmera en 3a persona, fantasma a la posició inicial i angles d'Euler en valors inicials).
- Rodes girant.

Aquest “reset” es farà mitjançant la tecla  (Qt: `:Key_R`) i ha de reiniciar també, si s'escau, els elements d'interfície adientment.