# Pràctica 2 – Fase 0 (I)

## Instal·lació de projectes bàsics

GiVD - curs 2022-23

**Objectiu general de la Pràctica 2:** Realitzar visualitzacions de dades utilitzant tècniques projectives (Z-Buffer), basades en la targeta gràfica (o GPU) programant *shaders* amb glsl.

**FASE 0 (1ª part): Introducció a l'arquitectura d'una aplicació gràfica. Comprovació de la instal·lació dels *drivers* usant uns primers programes d'exemple.**

La fase 0 té com objectiu testejar la teva instal·lació de Qt i GL per a ser capaç de realitzar la pràctica 2. Per això instal·laràs uns projectes ja implementats que visualitzen un model poligonal (o *Mesh*). Aquests projectes usen **OpenGL** i **C++.** En aquests exemples el model concret que es visualitza és un cub.

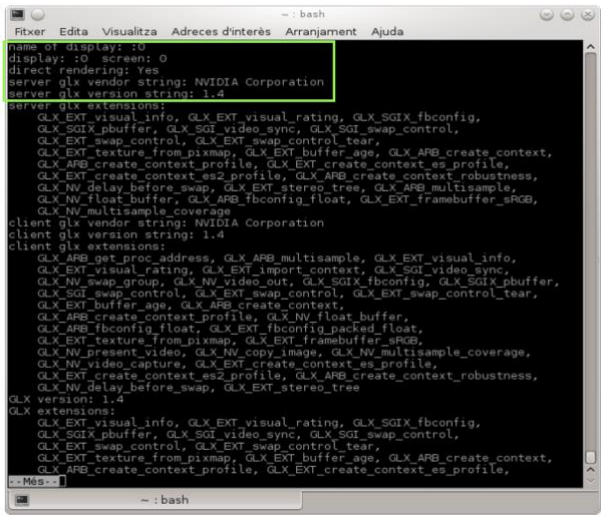
En aquesta primera fase, cal provar i testejar els *drivers* instal·lats per la teva targeta gràfica. Per això baixaràs i executaràs **tres** projectes. Si tens problemes amb la teva instal·lació, dimarts dia **11 d'abril** estarem a l'aula IF per a resoldre els problemes que tingueu en una sessió de tutories. Podeu venir independentment del grup al que pertanyeu.

### Índex



## PAS 0: Comprovació de la instal·lació de la targeta gràfica i d'OpenGL

Instal·lació de OpenGL. Reviseu si teniu instal·lat en el vostre computador les llibreries d'OpenGL. Des de Linux pots executar, des de la consola, la comanda glxinfo. Si tens una targeta Nvidia, t'hauria de sortir un missatge com:



Fixa't que el flag de *Direct Rendering* estigui activat.

Per a veure les versions instal·lades de GL i dels *shaders*, fes la comanda glxinfo | grep version i t'hauria de donar un missatge com:


Per anar bé, la instal·lació de GL hauria de ser 2.1 o superior. A l'exemple dóna la versió 4.4.0

Si tens un Mac, pots consultar en el següent enllaç si la teva targeta gràfica la suporta: https://support.apple.com/es-es/HT202823.

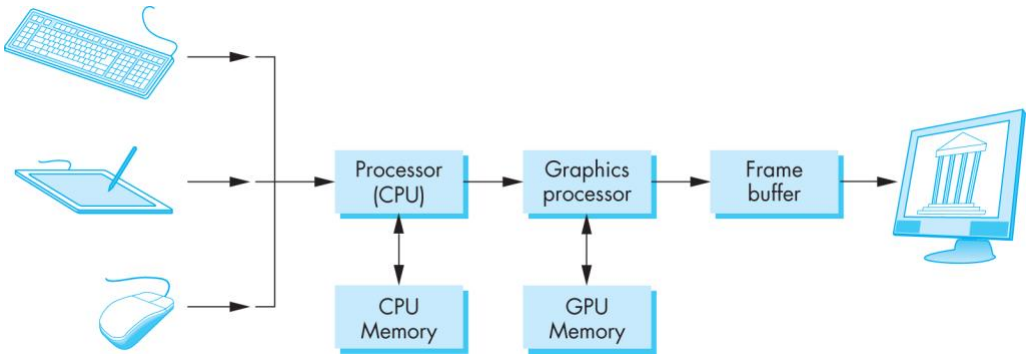
Dins ubuntu, existeix l'eina anomenada additional-drivers per instal·lar els drivers de la gràfica: https://itsfoss.com/install-additional-drivers-ubuntu/

Si no ho tens instal·lat el driver, vés a la plana https://www.khronos.org/opengl/wiki/Getting_Started on hi han les instal·lacions per a les diferents plataformes i targetes. A la wiki en el Campus Virtual pots veure les instal·lacions que fins ara tenim notícia que funcionen.

Si veus que tens una configuració nova que no hi és a la llista, si us plau, afegeix-la a la wiki de l'assignatura, ja que pot ajudar els teus companys i companyes ara i en anys futurs.

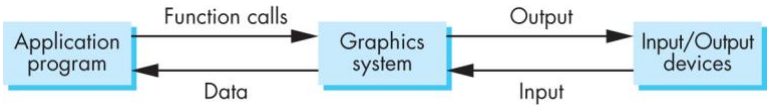
### PAS 1: CubGL: La primera aplicació GL: visualització d'un model poligonal (cub)

L'arquitectura en la què es basa tota aplicació gràfica és la següent (veure secció 1.1.1 del llibre de referència bàsica):




OpenGL és una llibreria que treballa en la memòria de la CPU i encapsula tot l'enviament de dades i gestions a la GPU. El traspàs a la GPU de les dades gràfiques (vèrtexs i colors) pot ser transparent al programador d'OpenGL, només usant crides a OpenGL d'alt nivell. Les aplicacions gràfiques clàssiques funcionen d'aquesta manera.

OpenGL ens ofereix diferents possibilitats per a visualitzar dades. Existeixen diferents funcions que permeten dibuixar punts, rectes, triangles, etc.




Les crides a aquests mètodes (o Function calls) es fan des de la CPU i les dades s'emmagatzemen en la memòria de la CPU. Quan es fa la crida a les funcions d'OpenGL de *display()*, es realitza la transferència a la GPU.

Baixa l'aplicació del campus CubGL.tgz de l'enllaç del campus virtual, descomprimeix-la i obre el projecte amb l'IDE QtCreator, seleccionant el fitxer CubGL.pro.

Observeu que en el fitxer .pro (o makefile) hi ha una línia que inclou la utilització de OpenGL. A vegades, si feu un projecte nou, cal editar especialment aquest fitxer per afegir aquesta opció:

```
QT       += opengl
```

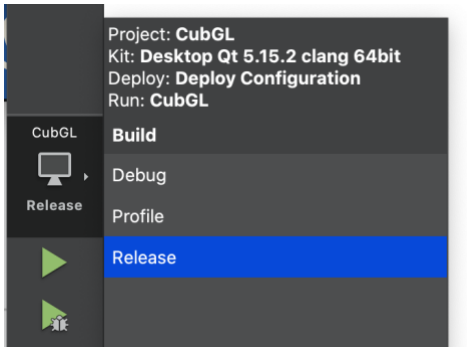
Sense ella, no és possible compilar amb els widgets de gl dins de Qt.

En la versió de Qt5.0 o superior cal incloure també la següent opció:

```
QT       += widgets
```

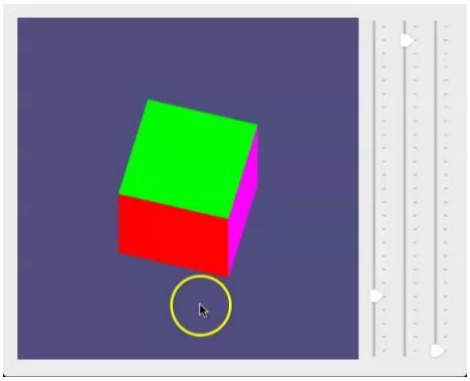
En aquest primer exemple no es programa la GPU o targeta gràfica, però si s'utilitza per la llibreria GL.

Pots provar a posar-te en mode Release en el projecte per a executar-lo més ràpidament:




Quan a la pràctica hagis de depurar errors, és millor posar-te en mode Debug.

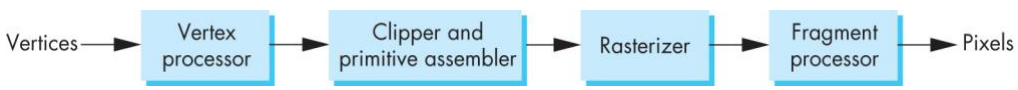
Executa el projecte i interacciona amb el cub tal i com es fa en el vídeo següent:



## PAS 2: CubGPU: Visualització del model re-programant la GPU

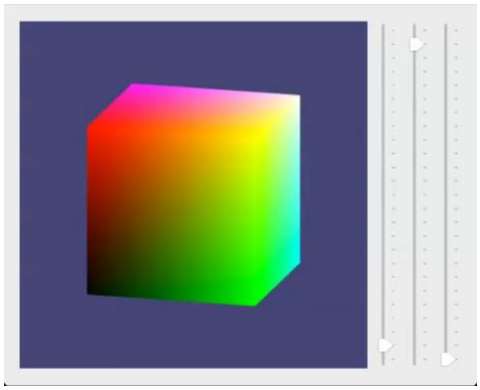
En els darrers anys, s'ha obert la possibilitat de treballar i programar directament en la GPU mitjançant el llenguatge GLSL. Això significa la possibilitat de fer la reprogramació directa d'algunes funcions que segueix oferint OpenGL, però tenint constantment la geometria i els colors a la GPU, evitant les transferències de memòria entre els dos processadors (CPU i GPU) i optimitzant així el temps de visualització (per introduir-te, si vols, llegeix la secció 1.7 i 2.8  del llibre de referència bàsica).

Una vegada es tenen els vèrtexs de l'objecte i les seves connexions, es poden realitzar les transformacions geomètriques en el processador de vèrtexs de la GPU. Aquest programa s'anomena *vertex shader.* A més a més, després de rasteritzar els vèrtexs en píxels, es pot programar el càlcul del color de cada píxel en el processador de la GPU de fragments (el programa s'anomena *fragment shader*).



Baixa ara l'aplicació CubGPU.tgz, descomprimeix el fitxer i obre un nou projecte amb l'IDE QtCreator. En la versió per a Qt 5.0 s'utilitzen unes estructures auxiliars per a passar informació a la GPU que són els **Vertex Buffer Objects (VBO)**, que són buffers que contenen els vèrtexs de l'Objecte**.** Amb aquestes estructures permeten passar els vèrtexs i la informació associada a cada vèrtex, com per exemple el color, la normal, etc. per a ser usada en el *vertex shader*.

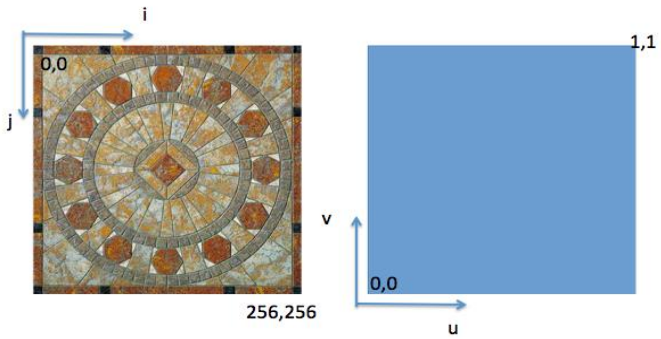
Executa el projecte. Hauries de veure la pantalla següent i podries interaccionar amb el cub:



## PAS 3: CubGPUTextures: Visualització del model utilitzant textures a la GPU

Fins ara, el model poligonal del cub es visualitza directament amb els colors associats a cadascun dels vèrtexs. En aquest exemple es veurà com es passa una textura a la GPU i s'utilitza per a visualitzar el color a cada píxel. En GL, **les dimensions en píxels de les imatges que formen les textures han de ser potències de 2.**

Disposem de la textura (o imatge mosaic.png), tal que cadascun dels seus píxels es mapegen en un espai 2D entre el (0,0) i el (1,1), tal i com havíem vist fins ara. Aquestes noves coordenades de píxel entre 0 i 1, s'anomenen **coordenades de textura**:




Baixa ara l'aplicació CubGPUTextures.tgz del Campus Virtual, descomprimeix el fitxer i obre un nou projecte amb l'IDE QtCreator. Executa-la i mira el que obtens.