Exemple Examen de laboratori d'IDI *(del curs 2023-2024, Q1).* Temps: 1h 50'

Cal que llegiu detingudament les instruccions i tot l'enunciat abans de començar a fer res!

Instruccions

1. Disposes d'un arxiu pdf a `~/examen/assig/idi/doc/LabIDI_Index.pdf` que conté tota la informació de les transparències de laboratori indexades de manera que siguin fàcils de trobar.

2. Partiràs del codi que tens a `~/assig/idi/blocs/examen2.tgz`. Has de desplegar aquest arxiu en un directori teu. Es crearà un subdirectori `examen-2324Q1` on tindràs tots els fitxers amb els quals has 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 has de modificar cap altre fitxer, NO pots modificar la classe `ExamGLWidget`! Si ho fas es penalitzarà.**

3. **Si el teu codi no compila o dóna error d'execució, l'avaluació serà un 0**, sense excepció.

4. Per fer l'entrega has de generar un arxiu que inclogui tot el codi del teu examen i que es digui `<nom-usuari>.tgz`, on substituiràs `<nom-usuari>` pel teu nom d'usuari. Per exemple, l'estudiant Pompeu Fabra (des d'una terminal en la que s'ha col·locat dins del directori `examen-2324Q1`):

   ```
   make distclean
   tar zcvf pompeu.fabra.tgz *
   ```

   ALERTA: si el que segueix a `zcvf` és el nom d'un arxiu que existeix, es sobreescriurà!

   És important el '`make distclean`' per esborrar els arxius binaris generats; que el nom d'usuari sigui el correcte (el teu); i que hi hagi el sufix `.tgz`

5. Un cop fet això, al teu directori `examen-2324Q1` tindràs l'arxiu `<nom-usuari>.tgz` que és el que has d'entregar. **Fes la comprovació**, desplegant aquest arxiu **en un directori completament buit**, que el codi que entregues compila (fent `qmake-qt5; make`) i executa correctament.

6. Finalment, el dia de l'examen, haureu de lliurar aquest fitxer al lloc indicat.

Nota: Obrint `~/examen/assig/idi/man_3.3/index.html` des dels navegadors firefox o konkeror tindràs accés a les pàgines del manual d'OpenGL 3.3, i amb `~/examen/assig/idi/glm/doc/api/index.html` tindràs accés a les pàgines del manual de la llibreria glm. També tens l'`assistant-qt5` per a dubtes de Qt.

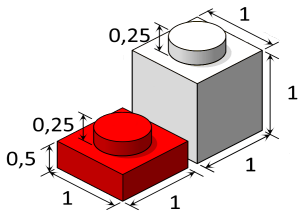
Recordeu que el dia de l'examen no tindreu accés a internet.

Enunciat

Lego és una línia de joguines de construcció plàstica, fabricada pel Grup Lego, una empresa privada establerta a Dinamarca. El producte Lego consisteix en vàries peces intercanviables de colors, simulant ser maons de plàstic que acompanyen una extensa varietat d'engranatges i figuretes. Les peces de Lego poden ser reunides i connectades de moltes maneres per construir objectes, incloent vehicles, edificis, robots o el que se'ns acudeixi. Qualsevol cosa construïda pot ser agafada a part un altre cop, i les peces es poden reutilitzar per fer coses noves.

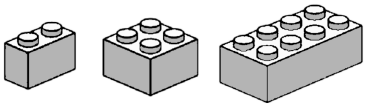
En aquest examen de laboratori volem simular un editor d'escenes amb peces de Lego. L'usuari haurà de ser capaç de seleccionar cada cop un tipus de peça, assignar-li un color, i col·locar-la on vulgui amb la orientació desitjada.

Les peces de Lego tenen unes dimensions que es multipliquen segons si el bloc té més files o columnes, o si son prims o normals. A més a més, tots tenen una mena de capçal que és el que serveix per acoblar les peces. La següent imatge il·lustra les dimensions que assumirem en aquest examen per als models que es faran servir.



El codi que us proporcionem crea i visualitza una escena amb:

- Un terra format amb blocs de Lego prims, de color blanc, transformat per a tenir coordenades que van del punt mínim (-10, -1, -10) al punt màxim (10, -0.25, 10).
- Un bloc de Lego centrat al punt (0, 2.5, 0), de color blanc, amb els seus capçals mirant cap a les Z-. El tipus de bloc varia al prémer la tecla 'M', iterant sobre els 3 tipus de peça següents:
  - Una peça de 1 x 2
  - Una peça de 2 x 2
  - Una peça de 4 x 2



- També es pot canviar el color del bloc a blau amb la tecla 'B'. Si es torna a prémer torna al blanc inicial.

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-a.png` mostra la visualització inicial amb el bloc de 1x2 blanc, i la imatge de l'arxiu `EscIni-b.png` la mostra amb el bloc de 4x2 blau.

Hi ha un mètode *creaBuffersModels* que crea els 4 models requerits. Aquests models 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 i hi ha un `uniform vec3 colorMul` al Vertex Shader que permet canviar el color del bloc de lego abans de pintar-lo. **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.

Una solució a tots els exercicis excepte el 6 la pots trobar a `~/assig/idi/blocs/examen2`

1. *(1.5 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.

2. *(1.5 punts)* Modifica l'escena per a que:

   - El terra es pinti de color vermell amb intensitat 0.7.
   - El bloc de Lego escollit amb la tecla 'M' es pinti correctament tombat i acoblat al terra. Fixa't que això vol dir que per als blocs de 2x2 i 4x2 els has de posar amb el centre de la seva capsa contenidora al punt (0,0,0), però que per al bloc de 1x2 has d'afegir un desplaçament extra per a que l'acoblament sigui correcte amb els capçals del terra.
   - Fes que també es pinti el bloc de Lego escollit amb GL_LINES a més a més de amb GL_TRIANGLES, però de color negre. És a dir fes un segon pintat del bloc usant la primitiva GL_LINES en lloc de GL_TRIANGLES.

3. *(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 capsa contenidora considera que l'escena va del punt mínim definit pel terra al punt màmim del terra en X i Z, i en Y l'alçada a la que arribem amb una torre de 10 blocs d'alçada 1 afegits damunt del terra.

   Cal afegir que es pugui fer també rotació de l'angle θ de la càmera, de manera que quan l'usuari mou el ratolí de baix a dalt en el viewport, l'angle θ s'incrementa i per tant la càmera es mou cap a dalt respecte l'escena. Quan l'usuari mou el ratolí de dalt a baix en el viewport, l'angle θ es decrementa. El valor inicial dels dos angles ha de ser de 15 graus. Una imatge de la solució als exercicis 1, 2 i 3 la pots veure a l'arxiu `EscEx1-3.png`.

4. *(2 punts)* Afegeix moviment per al bloc escollit:

   - El bloc es podrà moure en les direccions X-, X+, Y-, Y+, Z-, Z+ mitjançant les tecles 'A', 'D', `key_Down`, `key_Up`, 'S' i 'W' respectivament. Cada moviment ha de moure el bloc exactament 1 unitat. Limita també els moviments per a que el centre de les peces estigui acotat pel cub invisible que va del punt (-10,0,-10) fins al punt (10,10,10), evitant que aquest centre surti del terra, que el travessi, o que es mogui massa amunt.
   - El bloc també es podrà rotar respecte l'eix Y de l'escena que passa pel seu centre per canviar la seva orientació. Volem que en prémer la tecla 'Q' el bloc giri 90 graus sobre sí mateix. Aquest gir ha de ser acumulatiu i per tant prement dues vegades la tecla 'Q' farem rotar el bloc 180 graus.

   Afegeig també la possibilitat de col·locar de forma definitiva el bloc que s'està movent mitjançant la pulsació de la tecla `Key_Space`. Això implica modificar adientment els vectors definits a la classe `ExamGLWidget` per guardar: el tipus de model que està seleccionat (al vector `brickModelIndex`), la transformació geomètrica TG actual (al vector `bricksTGs`), el color actual (al vector `brickColors`) i afegir el pintat d'aquesta peça amb aquesta informació guardada.

   Un cop guardem com a definitiu un bloc, immediatament generem un d'igual (usant les mateixes dades que té l'últim guardat - TG actual, model actual i color actual) que serà la següent peça que volguem editar/afegir. Es poden afegir un màxim de `NUM_BRICKS` a l'escena (definit a `ExamGLWidget.h`). Pots veure dos imatges de la solució a aquest exercici als arxius EscEx4a.png i EscEx4b.png.

5. *(1 punt)* Afegeix una segona càmera ortogonal, Càmera-2, que mantingui la mateixa viewMatrix que amb la càmera amb perspectiva. Els plans de retallat han de permetre veure tot el que està davant d'aquesta càmera, aprofitant al màxim l'espai del viewport sense retallar l'escena ni deformar-la.

   Mitjançant l'ús de la tecla `C` l'usuari ha de poder canviar de la Càmera-1 a la Càmera-2 i a la inversa.

6. *(1.5 punts)* Afegeix a la interfície:

   a) Un element d'interfície que permeti decidir quin tipus de peça estem editant. Aquest element ha d'estar coordinat amb l'efecte de la tecla 'M'.
   b) Un o més elements d'interfície que permetin definir un color RGB per aplicar-lo a la peça que estem editant. Si fas aquest element d'interfície pots inhabilitar (eliminar) l'efecte de la tecla 'B' que ja us donàvem implementat.

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, 2 i 3: escena inicial, càmera en 3a persona, cap bloc col·locat, model, color, posició i orientació del bloc a editar inicials i angles d'Euler en valors inicials.

   Aquest "reset"es farà mitjançant la tecla `R` i ha de reiniciar també, si s'escau, els elements d'interfície adientment.