Examen de laboratorio de IDI 2016-2017, Q1 Tiempo: 1h 45'

Lee detenidamente las instrucciones y el enunciado antes de empezar a trabajar!

Instrucciones

1. Puedes usar el código que hayas elaborado en las clases de laboratorio y que tengas en tu cuenta, pero **solamente el código que hayas generado tu**. No puedes usar código que otros estudiantes hayan compartido contingo ni que tu hayas compartido con otros estudiantes. De lo contrario se considerará copia.

2. Partirás del código contenido en el fichero `examen.tgz` (adjunto a esta práctica). Tienes que descomprimirlo en un directorio tuyo. Se creará el subdirectorio `examen-1617Q2` donde encontrarás todos los ficheros con los que tienes que trabajar. Los ejercicios a resolver sólo requieren cambios en la clase `MyGLWidget`, en los shaders y en el fichero `MyForm.ui` usando el `designer`. **No tienes que modificar ningún otro fichero de los que te proporcionan**.

3. **Si el código que entregas no compila o da error de ejecución, la evaluación será un cero**, sin excepciones.

4. Para hacer la entrega tienes que generar un archivo tar que incluya todo el código de tu examen. El nombre del fichero será `<nombre-usuario>.tgz`, donde substituirás `<nombre-usuario>` por tu nombre de usuario. Por ejemplo, si `examen-1617Q2` es el directorio de trabajo, el estudiante Pompeu Fabra ejecutará desde un terminal los comandos

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

   Es importante: ejecutar `make distclean` para borrar los archivos binarios generados, que el nombre de usuario sea el tuyo y que tenga el sufijo `.tgz`.

5. Una vez hecho esto, en tu directorio `examen-1617Q2` tendrás el archivo `<nombre-usuario>.tgz` que es lo que tienes que entregar. **Comprueba** que todo es correcto descomprimiendo este archivo **en un directorio nuevo completamente vacío** i mirando que el código compila (haciendo `qmake-qt5; make`) y ejecuta correctamente.

6. Finalmente, entrega el fichero en `https://examens.fib.upc.edu`.

Nota: Recuerda que si abres el fichero `~/examen/assig/idi/man_3.3/index.html` desde el Firefox o el Konqueror tendrás acceso al manual de OpenGL 3.3, y que en `~/examen/assig/idi/glm/doc/api/index.html` tienes el manual de la librería `glm`. También tienes, como ya sabes, el `assistant` para dudas de Qt.

Enunciado

Tal y como está, el código que te proporcionamos pinta sobre el plano XZ un suelo centrado en el origen, de 20×20 unidades, con un Legoman de una unidad de altura y el centro de la base de la caja que lo contiene situado en el origen de coordenadas. Mira la imagen del archivo `EscIni.png`. La escena se representa con una cámara arbitraria de la que solamente puede modificarse interactivamente el ángulo ψ.

El código también inicializa todos los datos de material y normales necesarios para poder implementar el cálculo de la iluminación. También se proporcionan las funciones `Lambert` y `Phong` que se encuentran en el `Vertex Shader`.

Fíjate que se ha definido un método `createBuffers` para cada modelo, con el propósito de simplificar la lectura del código y su posterior utilización.

1. *(2 puntos)* Modifica la escena dada de manera que el Legoman sea substituido por 4 Patricios (modelo `Patricio.obj`) uniformemente escalados de altura 5, formando un corro y todos mirando hacia el centro del corro. Concretamente: un primer Patricio tendrá el centro de su base en el punto (-2.5, 0, 0) y mirará en dirección X+, el segundo Patricio tendrá el centro de la base en (0, 0, 2.5) i mirarà en dirección Z-, el tercero tendrá el centro de la base en (2.5, 0, 0) y mirará en dirección X- y el cuarto y último tendrá el centro de la base en (0, 0, -2.5) y mirará en dirección Z+.

   También debes modificar el material del suelo para que pase a ser un material brillante (no hay que cambiar el color).

2. *(1.5 puntos)* Para poder inspeccionar la escena descrita, programa una cámara en tercera persona y óptica perspectiva que inicialment permita ver la escena completa, centrada, sin deformar y ocupando el área máxima del viewport. En caso de redimensionar la ventana (`resize`) la escena no se deformará. La cámara permitirá la inspección de la escena mediante rotaciones de los ángulos de Euler (ángulos ψ y θ), es decir, el usuario tiene que poder modificar dichos ángulos utilizando el ratón como se ha visto en el laboratorio. Inicialmente la cámara debe estar mirando en dirección paralela al eje Z- de la aplicación.

   Una imagen ilustrativa de la solución a estos dos primeros ejercicios puedes verla en el archivo `EscSol1.png`.

3. *(1.5 puntos)* En el **Fragment Shader**, añade el cálculo de la iluminación de la escena usando el modelo de Phong con un foco de cámara de luz blanca situado siempre exactament en la posición de la cámara.

4. *(1.5 puntos)* Programa que el corro de los 4 Patricios gire respecto del eje Y de la escena. El ángulo de giro vendrá dado por un dial que deberás añadir a la interfaz. El dial generará valores en el rango [0..359] grados de manera que, cuando el dial complete una vuelta, los 4 Patricios (el corro de Patricios) han completado una vuelta en torno del eje Y de la escena. El valor inicial del ángulo de giro del corro será 0.

   El giro del corro debe funcionar independientemente de la cámara que esté activada.

5. *(2 puntos)* Añade una segunda cámara fija en la posición (0, 2.5, 0) de la escena y enfocando hacia el eje Z+. La óptica de la cámara será perspectiva con ángulo de apertura de `M_PI/2.0` radianes (90 grados) con posición y orientación fijas. Los valores de `Znear` y `Zfar` permitirán ver todo lo que pueda verse de la escena desde la posición de la cámara y en la dirección antes indicada. Caso de redimensionado del viewport, la cámara no deformará la escena.

   La nueva cámara y la programada en el Ejercicio 2 podrán intercambiarse mediante la tecla `C`.

   La posición y orientación de esta segunda cámara puedes programarla, si quieres, usando la función `lookAt` de la libreria `glm`.

6. *(1.5 puntos)* Incluye en la interfaz un `slider` que permita modificar el canal verde (G) del color RGB del foco de luz. El valor inicial del `slider` será el máximo y corresponderá a un foco de color blanco (componente G=1), pero el usuario debe poder modificar su valor, con lo que el color del foco pasará a tener menos componente verde hasta llegar al valor mínimo del slider (en este valor mínimo la componente G deberá ser 0).