Examen de laboratorio de IDI 2018-2019, Q1. Tiempo: 1h 50'

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

Instrucciones

1. Puedes usar el código que has 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 conotros. 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-1819Q1` donde encontrarás todos los ficheros con los que has de 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**.

3. **Si tu código no compila o da error de ejecución, la evaluación será un 0**, sin excepciones.

4. Para hacer la entrega tienes que generar un archivo que incluya todo el código de tu examen y que se llame `<nombre-usuario>.tgz`, substituyendo `<nombre-usuario>` por tu nombre de usuario. Por ejemplo, el estudiante Pompeu Fabra (desde el directorio `examen-1819Q1`):

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

   Es importante el '`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-1819Q1` tendrás el archivo `<nombre-usuario>.tgz` que es lo que tienes que entregar. **Comprueba** que todo es correcto descomprimiéndolo **en un directorio 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: Si abres el fichero `~/examen/assig/idi/man_3.3/index.html` desde el navegador tendrás acceso al manual de OpenGL 3.3, y en `~/examen/assig/idi/glm/doc/api/index.html` tienes el manual de la librería `glm`. También tienes el `assistant-qt5` para dudas de Qt.

Enunciado

El código que proporcionamos crea y visualiza una escena formada por un campo de fútbol de 20x12 unidades ubicado sobre el plano XZ y centrado en el origen, una portería de altura 6 a un lado del campo, un balón con el centro de la base de su caja contenedora en el punto (6, 0, 0) y un Patricio. La cámara está inicializada arbitrariamente y sólo se puede modificar interactivamente el ángulo ψ. La imagen del archivo `EscIni.png` muestra la visualización inicial de la escena.

Hay un método *createBuffers* para cada modelo. Este método tiene inicializados todos los datos de material y normales necesarios para poder implementar el cálculo de la iluminación. También proporcionamos las rutinas `Lambert` y `Phong` que se encuentran en el Vertex Shader. Además **se da ya implementado el movimiento básico del balón**, de modo que si se pulsa la tecla Key_Up (chutamos) el balón se pone en movimiento hasta que llega al fondo de la portería. Mediante la tecla Key_I el balón vuelve a su posición inicial y se puede volver a chutar. **Observación: Analiza el código dado antes de implementar funcionalidades**.

En la valoración de los ejercicios 4, 5 y 6 tendrá mucha importancia el diseño y la usabilidad de la interfaz.

1. *(1.5 puntos)* Modifica la escena para que, en lugar de un Patricio haya 2 legomans (modelo `legoman.obj`): Lego1, Lego2. Ambos legomans deben tener altura 4. El primer legoman, Lego1, es el jugador que chuta y estará situado con el centro de la base de su caja contenedora en el punto (8, 0, 0) y mirando en dirección X-. El segundo legoman, Lego2, será quien hará de portero y estará situado inicialmente con el centro de la base de su caja en el punto (-7, 0, 0) y mirando en dirección X+. Observación: el modelo legoman.obj mira en dirección Z+.

2. *(2 puntos)* La escena se debe poder inspeccionar con una cámara en tercera persona que permita ver la escena centrada, entera, sin deformar y ocupando el máximo del viewport (siendo el viewport toda la ventana gráfica). La cámara debe tener una óptica perspectiva. En caso de redimensionamiento de la ventana (resize) la escena no debe deformarse ni recortarse. Esta cámara también debe permitir la inspección mediante rotaciones de ángulos de Euler (ángulos ψ y θ), es decir, el usuario debe poder modificar estos ángulos utilizando el ratón como se ha hecho en laboratorio. La cámara debe tener inicialmente ángulos $\psi = M\_PI/4.0$ y $\theta = 0$.

   Puedes ver una imagen de la solución a estos dos primeros ejercicios en `EscSol1.png`.

3. *(1 punto)* Añadir a la escena el cálculo de iluminación **en el Vertex Shader** usando el modelo de iluminación de Phong y con un foco de luz blanca situado siempre exactamente en la posición de la cámara.

4. *(1.5 puntos)* Agregar una segunda cámara que será una cámara en primera persona. Esta cámara se situará en la posición (8, 5, 0) y tiene que estar orientada en la misma dirección en la que mira el Lego1 y con vector up (0,1,0). La óptica de esta cámara debe ser perspectiva con ángulo de abertura fijo de $M\_PI/2.0$ radianes (90 grados). Los valores de `Znear` y `Zfar` deben permitir ver la parte de la escena que hay delante del legoman. La cámara no se modificará con interacción del ratón y no deformará la escena en caso de redimensionamiento del viewport.

   Esta nueva cámara debe poder activarse/desactivarse mediante un elemento de interfaz. Cuando se desactiva hay que volver a la vista anterior de la cámara en tercera persona.

5. *(1.5 puntos)* Para poder modificar la trayectoria del balón y mostrar mejor cuándo se marca un gol hay que hacer lo siguiente:

   (a) Define un elemento a la interfaz que permita indicar/modificar el ángulo en el que se chuta el balón. Este ángulo inicialmente es 0 y debe poder variar entre -15 y 15 grados. Observa que este ángulo lo recibe el método `mouPilota(int alfa)` en grados, que ya lo tiene implementado.

   (b) Cuando el balón entra a portería el foco de luz debe cambiar de color a rojo. Implementa el método `tractamentGol()` para que haga este cambio de color del foco.

   (c) Cuando el usuario devuelve la pelota al inicio (con la tecla Key_I) el foco debe volver a ser blanco.

6. *(1 punto)* Añade la posibilidad de que el legoman portero, Lego2, se mueva en la dirección del eje Z, entre los puntos pmin = (-7,0, -3) y pmax = (-7,0,3). Haz que el movimiento se realice por las flechas laterales (`Key_Left` y `Key_Right`) de modo que la izquierda lo haga mover en dirección Z+ y la derecha lo haga mover en dirección Z- (corresponde a izquierda y derecha desde el punto de vista del Lego1).

   Completa el método `aturaPorter()` haciendo que compruebe si el portero ha parado el balón. Para comprobar esta parada se deben cumplir las dos condiciones siguientes:

$$posPilota.z \in [posPorter.z - 1, posPorter.z + 1]$$

$$posPilota.x \in [-6.3, -5.8]$$

7. *(1.5 puntos)* Añade a tu interfaz un botón de "Reset"que devuelva a la situación inicial del programa. Es decir, la cámara está en tercera persona, con los parámetros iniciales descritos en el ejercicio 2, el balón y el portero deben estar también en la posición inicial y con el ángulo inicial y el color del foco debe ser blanco. Del mismo modo, todos los elementos de interfaz implicados deben tienen que volver a sus valores iniciales.