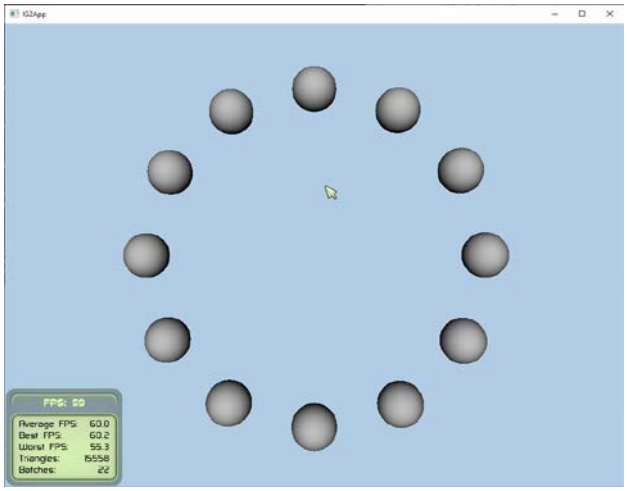
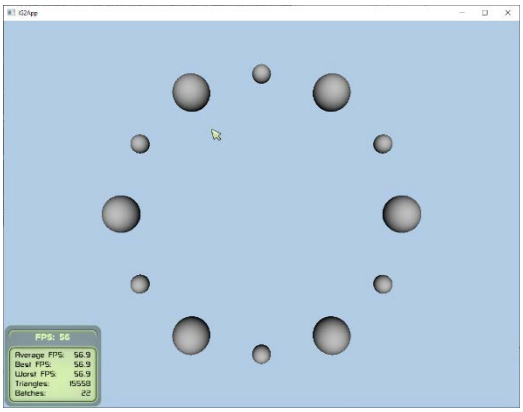
INFORMÁTICA GRÁFICA 2
Grado en desarrollo de videojuegos
Curso 2021-22
Práctica 1

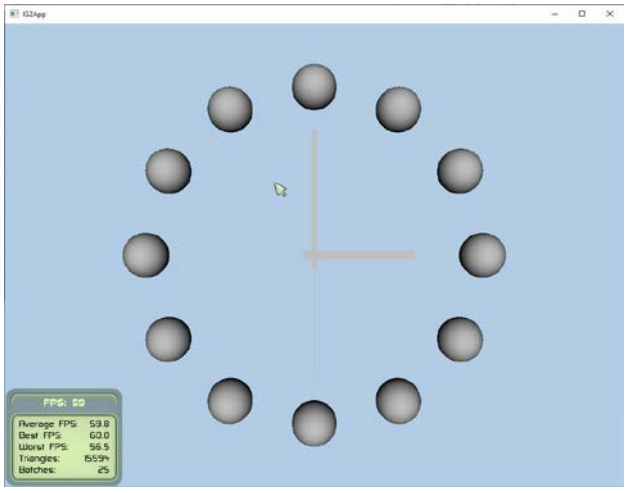
(Entrega 1, apartados 1-16)



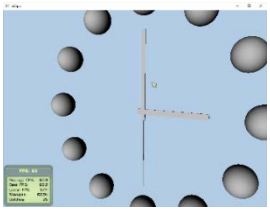
1. Crea una escena que muestre doce esferas en las horas de una circunferencia, tal como se muestra en la captura. Los nodos que contienen las esferas tienen los nombres **`Hora 1, Hora 2, …`** y son las componentes de un array **`mHourNode[12]`** de nodos. Cada una de las componentes de este array es hijo de un nodo **`Clock`** que, a su vez, es hijo de la raíz. Para colocar las esferas en su posición correcta usa las operaciones **`Ogre::Math::Cos()`** y **`Ogre::Math::Sin()`** y el método para colocar nodos en una posición dada **setPosition(…, …, …)**.



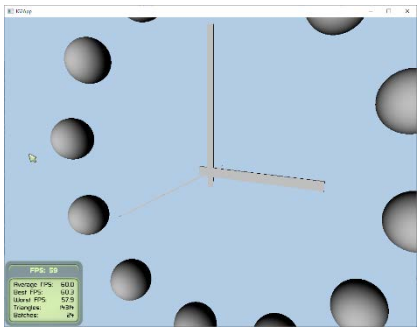
2. Añade código a la escena anterior que modifique las esferas de las horas pares de forma que se muestren más pequeñas, como en la captura de al lado. Para modificar el tamaño de un nodo, **no accedas a él a través del array de nodos `mHourNode[12]`** sino a través de su nombre, usando el método del gestor de escena **`getSceneNode("…")`**. Una vez accedido, modifica su tamaño con el método **setScale(…, …, …)**.



3. A partir de la escena con las doce esferas iguales, añade añade hijos a **`Clock`** para tener tres agujas (horas, minutos y segundos), cada una algo más delgada que la anterior, y colócalas de manera que se muestre un reloj con la hora de la siguiente captura:



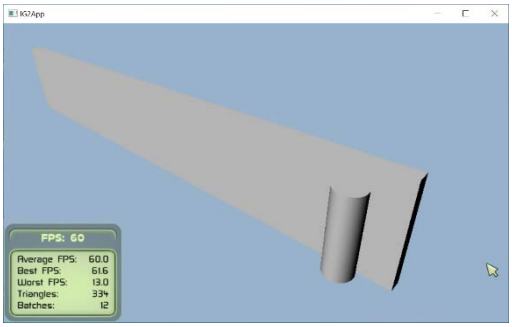
Para esta escena, a las operaciones que has usado hasta ahora puedes añadir la siguiente **`roll(Ogre::Degree(-90))`**.



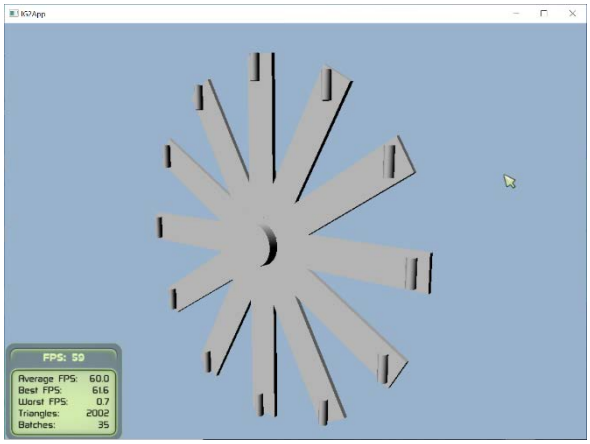
4. Añade un evento en la tecla **g** de forma que, al pulsarla, el reloj gire, esferas y horas incluidas, alrededor de su eje **Z**.

5. Añade un evento en la tecla **h** de forma que, al pulsarla, giren solamente las esferas.

6. Modifica la posición de la aguja de los segundos de forma que quede como se muestra en la captura adjunta.



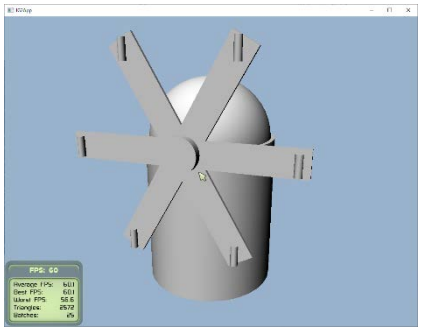
7. Define la clase **`Aspa`** cuyos objetos se renderizan como se muestra en la captura adjunta. Del **`mNode`** de esta clase cuelgan **`tableroNode`**, que contiene la malla de un cubo **`cube.mesh`**, y **`adornoNode`**, que contiene la malla de un cilindro **`Barrel.mesh`**. Estos nodos se escalan y sitúan apropiadamente y queda a tu elección crearlos con o sin nombre.



8. Define la clase **`AspasMolino`** cuyos objetos se renderizan como se muestra en la captura adjunta. Del **`mNode`** de esta clase cuelgan **`cilindroCentralNode`**, que contiene la malla de un cilindro, y **`aspasNode`**, del que cuelgan **`numAspas`** nodos, cada uno de los cuales contiene un aspa de la clase anterior. Las aspas se reúnen en un **`arrayAspas`** para facilitar su creación. Tanto el número de aspas como el array que las contiene son atributos de la clase. Cuando se construya un objeto de esta clase, los adornos de las aspas deben estar verticales.

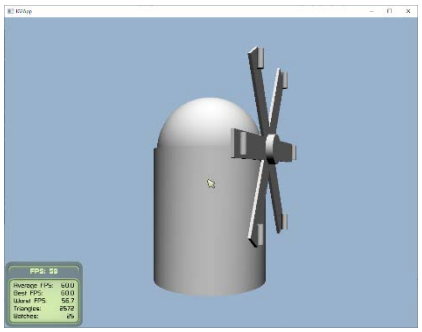
9. Añade el evento **g**, que gira las aspas, al **`keyPressed()`** de la clase **`AspasMolino`**, manteniendo verticales los adornos.

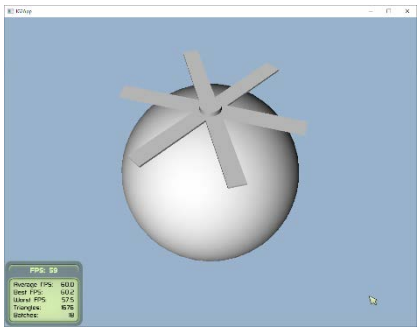
10. Define la clase **`Molino`** cuyos objetos se renderizan como en la captura adjunta. Su **`mNode`** tiene tres hijos: la esfera del techo, el cilindro del cuerpo del molino y las aspas. Las aspas se construyen como un objeto de la clase **`AspasMolino`**. Evidentemente el evento **g** continúa girando las aspas.



11. Añade el evento **c** a la clase **AspasMolino** de manera que el cilindro central se retire hacia atrás cuando la tecla es pulsada.

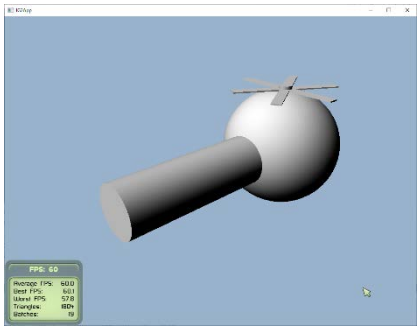
12. Añade el evento **h** a la clase **Molino** de manera que las aspas con su cilindro central roten alrededor del (techo del) molino. En la captura adjunta las aspas se han girado con esta tecla y han pasado de mirar de frente a mirar a la derecha, como aparecen. Implementa la rotación de dos formas diferentes.





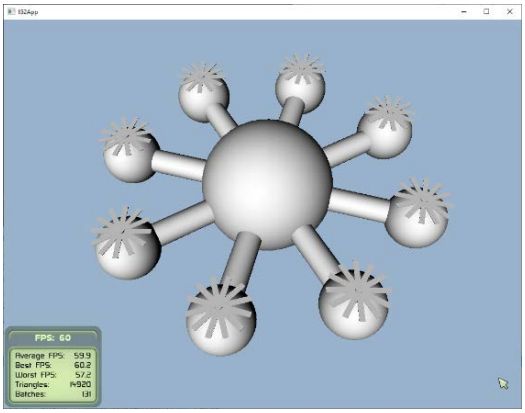
13. Define la clase **RotorDron** cuyos objetos se renderizan como en la captura adjunta. Del **mNode** de esta clase cuelgan **esferaNode**, que contiene la malla de una esfera, y **helicesNode**, que contiene un objeto de la clase **AspasMolino**, pero con aspas que no muestran adorno. El número de aspas de un rotor de dron es un atributo de la clase.

14. Haz que la tecla **g** siga girando las aspas de un rotor.

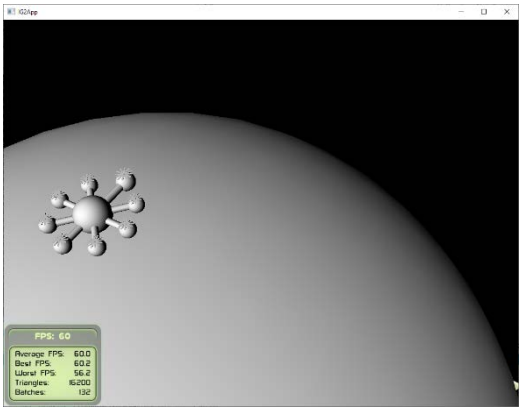


15. Define la clase **BrazoDron** cuyos objetos se renderizan como en la captura adjunta. Del **mNode** de esta clase cuelgan un nodo para el cilindro y otro para un elemento de la clase **RotorDron**. El número de aspas sigue siendo un atributo de la clase y las aspas del rotor siguen girando con la tecla **g**.

16. Define la clase **Dron** cuyos objetos se renderizan como en la captura de más abajo. Del **mNode** de esta clase cuelga un nodo para la esfera que hace de cuerpo central a la que rodean un número determinado de brazos, cada uno contenido en un nodo, todos ellos reunidos en un array. El número de aspas y el número de brazos son dos atributos de la clase. El dron de la captura tiene 8 brazos y 12 aspas cada rotor. La tecla **g** hace girar las aspas con la peculiaridad de que, si las de un rotor giran en sentido horario, las de los rotores contiguos, uno a cada lado, lo hacen en sentido anti-horario, asumiendo que el número de brazos sea par.

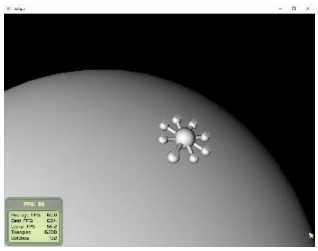


(Entrega 2, apartados 17-41)



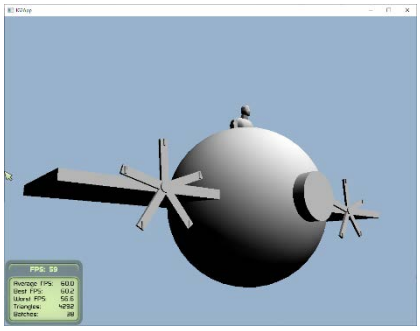
17. Para saber la dirección de avance de un dron, cuando este se mueva, haz que uno de sus brazos sea más grande. Crea una escena con dos nodos: **`planetaNode`**, que contiene una esfera, y **`ficticioDronNode`**, del que pende el **`mNode`** de un dron situado encima del polo norte de la esfera.

18. Añade el evento de la tecla **h** de **`IG2App`** que mueve el dron por encima del meridiano de la esfera que pasa por debajo de la dirección de avance del dron.



19. Haz lo mismo con el evento de la tecla **`j`** de **`IG2App`** para girar el dron sobre sí mismo y determinar una nueva dirección de avance. Fíjate que con estos dos eventos es posible hacer pasar el dron por encima de cualquier punto de la esfera.

20. Haz estos movimientos con dos de los métodos propuestos: nodo ficticio y truco.



21. Define la clase **`Avion`** cuyos objetos se renderizan como el que se muestra en la captura adjunta, a la izquierda. Del **`mNode`** de esta clase cuelgan los siguientes: **`cuerpoNode`** para la esfera de la nave, **`alaINode`** y **`alaDNode`** para las alas izquierda y derecha y que tienen, **cada una**, un cubo escalado apropiadamente (no es correcto representar las dos alas del avión mediante un listón único que atraviesa la esfera) **`frenteNode`** para el cilindro de la parte delantera del avión, **`pilotoNode`**, para el piloto ninja, y dos **`heliceNode`**'s que son elementos de la clase **`AspasMolino`** y que están situados en la mitad del borde de las alas.

22. Define el evento de la tecla **g** de manera que roten las aspas de las hélices del avión, a la vez y en el mismo sentido.

23. Construye la malla de un plano y agrégala como entidad al **`setupScene()`** de **`IG2App`** dentro de un nodo de la escena.

24. Extiende tu proyecto con la clase **`EntidadIG`** que se define tal como se ha explicado.

25. Refactoriza las clases **`Aspa`**, **`AspasMolino`**, **`Molino`**, **`RotorDron`**, **`BrazoDron`**, **`Dron`** y **`Avion`** de forma que hereden solamente de **`EntidadIG`**.

26. Define la clase **`Plano`** que hereda de **`EntidadIG`** de forma que el plano que creabas antes en el **`setupScene()`** de **`IG2App`** pase ahora a ser creado en la clase **`Plano`** y, a partir de él, crea una entidad que irá asociada al nodo principal de esta clase.

27. Tras los cambios de los apartados anteriores, asegúrate de que los eventos de teclado siguen funcionando.

28. Crea una escena con un planeta, un dron, un avión y un plano de fondo, detrás de ellos.

29. Añade un foco tanto al dron como al avión de manera que, cuando esté encendido, se ilumine la parte del planeta por encima de la que pasan.

30. Añade el método:

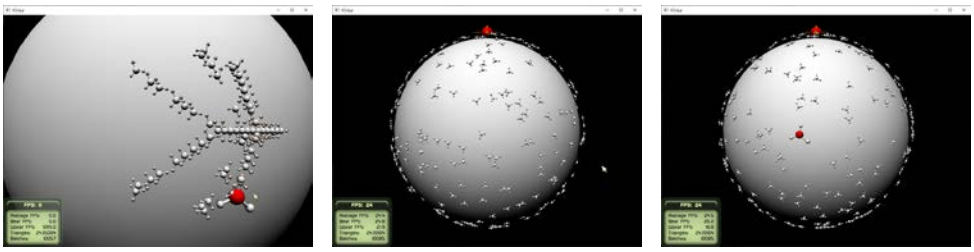
```
virtual void frameRendered(const Ogre::FrameEvent& evt) {}
```

a **`EntidadIG`**.

31. Añade el método **`frameRendered()`** a la clase **`Dron`** de forma que el dron se mueva alrededor del planeta. Cada dos segundos, el dron se para un lapso breve de menos de dos segundos y gira sobre sí mismo, para cambiar de dirección; antes de iniciar este giro, decide aleatoriamente si lo va a hacer horario o anti-horario. Para facilitar esto, puedes hacer que el dron disponga de un temporizador **`Ogre::Timer* myTimer`** y la función **`getMilliseconds()`**. Incluye **`OgreTimer.h`** para usarlo.

32. Añade el método **`frameRendered()`** a la clase **`Avion`** de forma que el avión se mueva alrededor del planeta y cambie su dirección de forma similar a como lo hace el dron.

33. **[Opcional. La caza de drones]** Define y añade un enjambre de micro-drones a la escena. Incluye un dron de control, más grande que los micro-drones, de esfera central roja.



34. **[Opcional. La caza de drones]** Añade al evento de teclado **`h`**, que movía el avión alrededor del planeta, que también vaya eliminando los micro-drones del enjambre con los que choque. Para esto usa la operación de **`node`** que devuelve la caja alineada con los ejes que lo contiene: **`AxisAlignedBox aab = node->_getWorldAABB()`**. La caja intersección de dos de estas cajas **`aab1`** y **`aab2`** se calcula con **`aab1.intersection(aab2)`**.

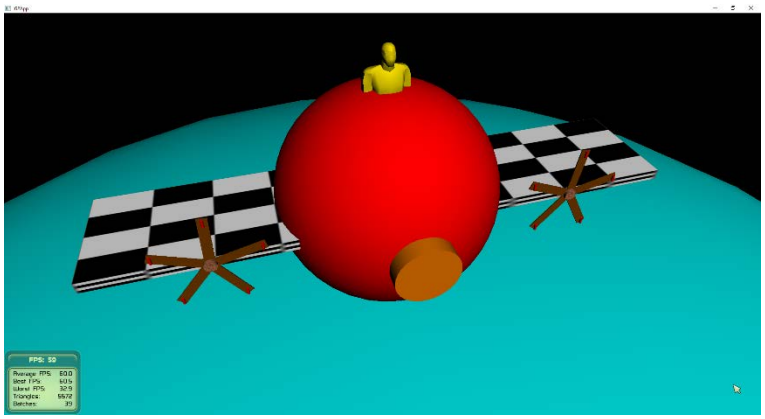
35. **[Opcional. La caza de drones]** Añade un control a la caza de forma que, cuando no queden micro-drones, la esfera central del dron de control se ponga amarilla.

36. Programa el evento que se describe a continuación, mediante el envío de eventos apropiados a través del vector estático **`appListeners`**. En una escena con el planeta, un avión y un dron, al pulsar la tecla **`r`** se producen los siguientes acontecimientos:

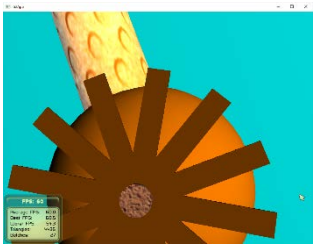
- el avión se detiene, pero no el giro de sus hélices
- el dron se detiene, pero no el giro de sus hélices
- la esfera central del dron se vuelve roja (antes no lo era)
- las alas del avión se vuelven rojas.

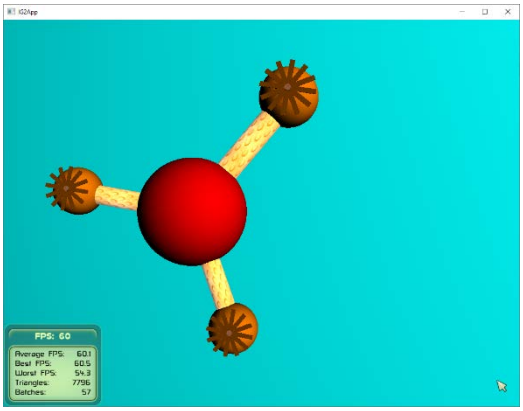
37. Añade material a la escena mediante un script `Practica1.material`. En los siguientes apartados se dotará a la escena de color y texturas.

38. Añade color y texturas al avión y al planeta tal como se muestra en la captura.

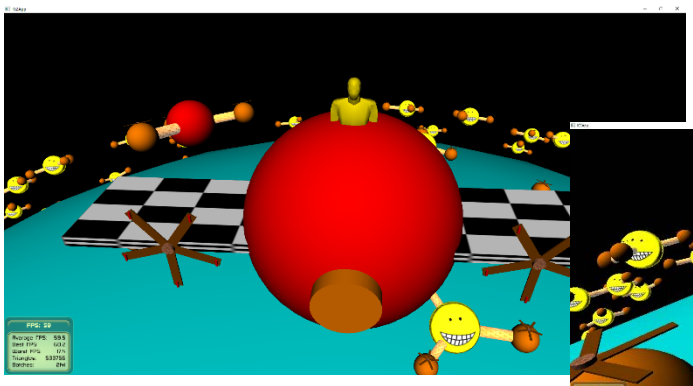


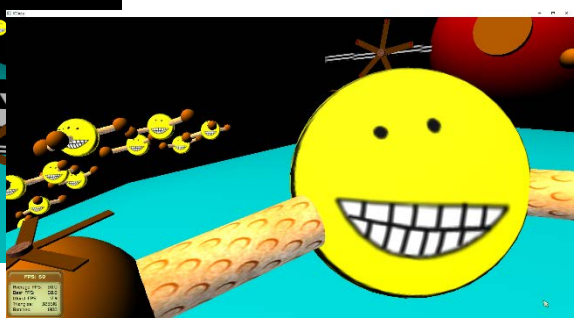
39. Da textura y color al dron tal como se muestra en la captura adjunta.





40. **[Opcional. La caza de drones]** Añade efecto billboard a la textura de la esfera central de los micro-drones tal como se muestra en la captura adjunta.

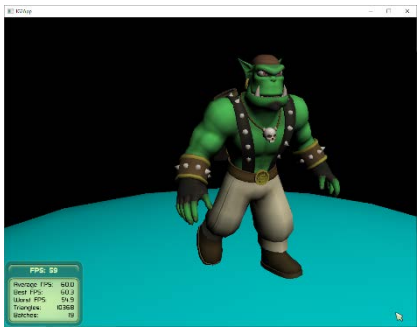




41. Define la clase **`Sinbad`**, que hereda de **`EntidadIG`**, cuyas entidades tienen como malla asociada `Sinbad.mesh`.

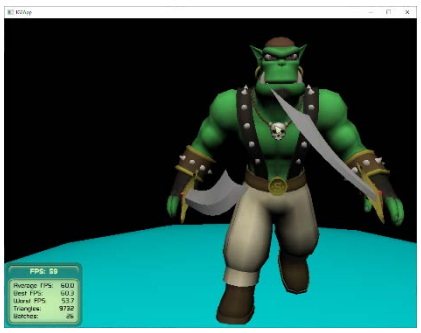
(Entrega 3, apartados 42 -)

42. Añade un Sinbad a la escena del planeta y colócalo en el lugar del avión. Anima la figura de Sinbad de manera que corra sobre la posición que ocupa, sin moverse. Los estados de animación para correr son dos y se llaman **`RunBase`** y **`RunTop`**.



43. Escribe el código para que se muestren los trece estados de animación de Sinbad.

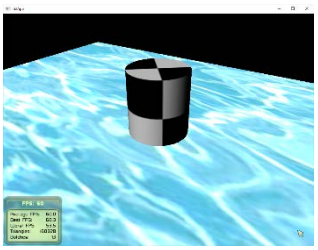
44. Programa el **`framRendered()`** de la clase **`Sinbad`** de forma que Sinbad corra aleatoriamente por toda la superficie de la esfera.



45. Define un método **`arma(bool)`** en la clase **`Sinbad`** que añada una espada a la mano derecha o izquierda de Sinbad. Define un método **`cambiaEspada()`** que cambia de mano la espada de Sinbad. Define el método **`arma()`** que añade espada a las dos manos de Sinbad. Recuerda que la malla de la espada es **`Sword.mesh`**. Recuerda también que las articulaciones de las manos se llaman **`Handle.R`** y **`Handle.L`**.

46. Haz que con la tecla **`c`** se pueda cambiar la animación de **`Sinbad`** de bailar a correr y viceversa, pulsando otra vez la tecla. Cuida que, al cambiar de animación, Sinbad haga exactamente lo que debe en su estado de animación y no se quede en una mezcla de ambos. Abajo hay dos capturas del estado de animación de Sinbad bailando. Este estado se llama **`Dance`**.

47. Define la clase **Bomba**, que hereda de **EntidadIG**, cuyos objetos tienen una entidad con **Barrel.mesh** como malla asociada y que se mostrará con **checker.png** como textura.



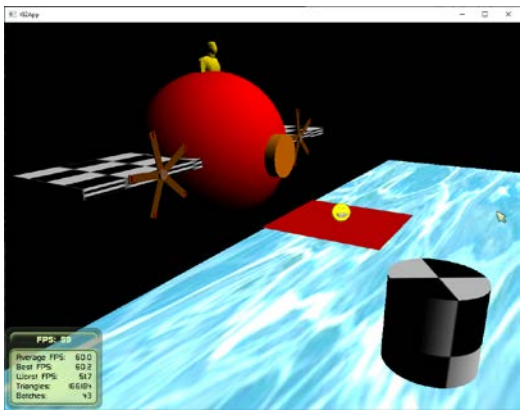
48. Programa la animación de nodo **AnimVV** explicada en las transparencias, que mueve el cilindro de la clase **Bomba** de arriba abajo, con giro de 45 grados con respecto al eje **Y**. El cilindro está en medio de un plano con un rio como muestra la captura anterior.

49. Define una nueva escena con el río, una plataforma roja, una amarilla, una bomba y un Sinbad en la plataforma amarilla. Haz que el agua del rio se mueva a lo largo de una de las diagonales del plano del rio. Sinbad corre con una espada en cada mano, sin moverse del sitio. El aspecto de la escena es el que se muestra en la captura adjunta.

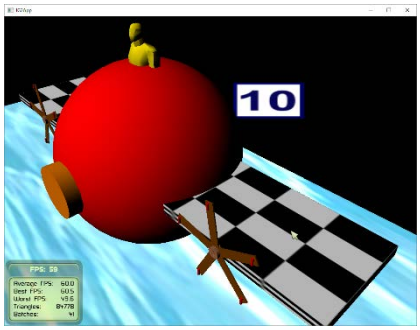
50. Programa el evento de mensajería que se describe a continuación. Cuando se pulsa la tecla **t**, la bomba se detiene en su movimiento de arriba abajo y, 5 segundos después, el agua del rio detiene su curso y su textura pasa de mostrar agua a mostrar piedras, tal como se muestra en la captura adjunta.

51. Programa la animación de nodo que hace que Sinbad vaya corriendo desde la plataforma amarilla en la que se encuentra a la roja y vuelva. Inicialmente Sinbad está mirando paralelamente a la parte positiva del eje **Z**, es decir, mira hacia adelante.

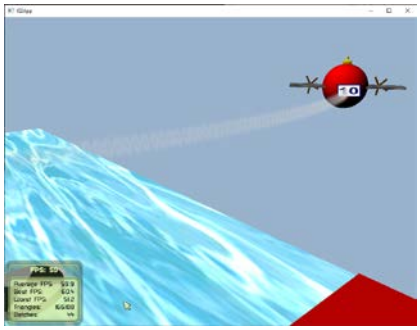
52. Añade los siguientes nuevos elementos a la escena. Una esfera con la cara feliz como textura billboard, situada sobre el centro de la plataforma roja. Y un avión como el de la escena del planeta situado como y donde aparece en la captura adjunta. Haz que el avión de vueltas alrededor de la bomba, sin cambiar de altura y rotando sus hélices.



53. Añade la textura **`10points.png`** como billboard set de un solo elemento, en la parte de atrás del avión, tal como se muestra en la captura adjunta.



54. Pon **`ColourValue(0.6, 0.7, 0.8)`** como color del background. Define un sistema de partículas para simular la estela de humo que va dejando el avión en su rotación. Usa para ello la textura **`smoke.png`**. La estela desaparece al poco de salir del avión. Mira la captura adjunta.



55. Define un conjunto de billboards para simular niebla y cielo nuboso sobre la parte derecha del rio. Usa para ello, de nuevo, la textura **`smoke.png`**. La captura adjunta intenta mostrar lo que se pretende.