Prácticas Corutinas en Lua

Práctica 1

Utilizad una corutina para pintar una criatura distinta cada vez que se haga click en pantalla, pero sin repetir las criaturas hasta que se acaben.

Práctica 2

Crear un iterador mediante wrap para obtener las criaturas de un array de criaturas. Estas criaturas tendrán una propiedad de vida.

Modificarla para obtener sólo las que tengan más de 50 de vida

Práctica 3

Otro uso típico de las corutinas en videojuegos es implementar cinemáticas in-game. Vamos a hacer una de ellas usando nuestro motor y nuestras criaturas.

Para ello disponemos de varias cosas nuevas:

function addEntity(texture, posX, posY, sizeX, sizeY, pivotX, pivotY)

- devuelve un objeto entity que se puede usar después para realizar acciones

function removeEntity(entity)

- Elimina la entidad de la pantalla

function getEntityPosition(entity)

- Devuelve x, y con la posición de la entidad

function setEntityPosition(entity, x, y)

- Pone a la entidad pasada como parámetro en la posición indicada

function getEntityRotation(entity)

- Devuelve la rotación de la entidad. Ángulo en grados

function setEntityRotation(entity, angle)

- Pone la rotación de la entidad. Ángulo en grados

function getEntityScale(entity)

- Devuelve la escala de la entidad en X e Y. La escala en el tamaño original sería 1.0

function setEntityScale(entity, scaleX, scaleY)

- Pone la escala de la entidad en X e Y

function moveEntity(entity, x, y, time)

- Mueve la entidad los pixeles indicados en el tiempo suministrado. X e y indican desplazamiento, no posición destino.

function rotateEntity(entity, angle, time)

- Rota la entidad los grados indicados en el tiempo suministrado

function scaleEntity(entity, scaleX, scaleY, time)

- Escala la entidad lo indicado en scaleX y scaleY en el tiempo suministrado. No es escala destino, se indica cuánto se va a escalar la entidad en el estado actual.

Además se dispone de un módulo: sequence.lua.

Este módulo contiene una clase llamada Sequence que define las siguientes funciones:

function Sequence:new()

- Función para crear una instancia de la clase Sequence. Tiene el comportamiento típico que hemos visto en clase

function Sequence:start()

- Función para comenzar a ejecutar la secuencia

function Sequence:sleep(seconds)

- Función para parar la ejecución de la secuencia durante los segundos indicados mediante seconds. Cuando haya pasado ese tiempo se continuará la ejecución después del sleep

function Sequence:run(seconds)

- Función que se llama para ejecutar la secuencia. En esta clase no hace nada, debe ser reimplementada en clases derivadas para definir el comportamiento de la secuencia.

El objetivo de la práctica es crear una clase derivada de Sequence que implemente una secuencia in-game del juego.

Secuencia 1

Aparece un dragon en la posición 100, 100, mirando a la derecha. Se gira a la izquierda, luego a la derecha y a la izquierda de nuevo, como si estuviera mirando. Aparece un mago en la posición 600, 500. El dragón se gira mirando hacia él, tras dos segundos de espera se dirige hacia él. Cuando está a punto de llegar el mago se sorprende (se hace grande y luego vuelve a su escala rápidamente) y trata de huir hacia la posición 800, 100. Tras un segundo de duda, el dragón lo sigue. Cuando llega hasta la cercanía del mago, éste triplica su tamaño, dispara una bola de fuego hasta la posición del dragon y éste muere.

Secuencia 2

Diseña tú una secuencia con las imágenes que quieras y el comportamiento que quieras. Si te falta alguna funcionalidad para conseguir lo que quieres y no puedes implementarla habla con el profesor por si se puede extender la librería para permitirlo.

Opcionales

Práctica 4

En esta práctica vas a intentar implementar el modulo sequence.lua para que haga lo que hace la clase proporcionada por el profesor. Para implementar la funcionalidad de sleep tendrás que hacer uso de corutinas.