TP Videojuegos

Práctica 5

Fecha Limite: 24/04/2019 a las 09:00.

Parte 1: Usar la clase InputHandler para manejo de entrada

Modifica el juego asteroids para que use la clase InputHandler (que vimos en clase) para el manejo de entrada:

1. Descargar la clase InputHandler (es parte del ejemplo del tema *Input Handler*).

2. Quitar el parámetro event del método handleInput en las clases GameObject y InputComponent (y de todas la clases que heredan de GameObject o InputComponent).

3. Modificar el método handleInput en AsteroidsGame.cpp para que use el InputHandler para el manejo de entrada (ver PingPong.cpp en el ejemplo del tema *Input Handler*)

4. Modificar todas las clases que manejan entrada para que usen el InputHandler (GunIC, GameCtrlIC, RotationIC, ThrustIC, etc.)

Parte 2: Añadir una clase Logger al juego

En esta parte vamos a añadir la posibilidad de registrar información (normalmente de depuración) en un archivo durante el juego. La funcionalidad se implementa mediante una clase Logger usando el patrón **Singleton** y un **Worker** para procesar las peticiones en una hebra.

```
class Logger {
public:
    inline static void initInstance(string filename) {...}
    inline static Logger* instance() { … }
    void log(string info);
    void log(function<string()> f);
    ...
Private:
    ...
    Worker worker_;
    ofstream log_;
};
```

TP Videojuegos, Práctica 5, Fecha Limite: 24/04/2019 a las 09:00.

Completar el código de la clase Logger. Observa que la clase usa el patrón Singleton con la *separación de crear/inicializar y consultar la instancia* (ver las transparencias del tema Singleton), eso es necesario para poder pasar el nombre del archivo en el que queremos escribir la información a la constructora. La clase Worker está disponible en el campus virtual (el ejemplo ex14 del tema programación multi-hebra).

1. En la constructora abre el archivo en modo texto (usando el atributo log_) y llamar a worker_.start(). En la destructora cierra el archivo (usando el atributo log_). En caso que creas la instancia usando memoria dinámica, recuerda que para poder ejecutar la destructora hay que usar un unique_ptr (ver las transparencias del tema Singleton).

2. En el método log(string info) añade una tarea (lambda expression) al worker_ para escribir el valor de info en log_ (y salto de línea al final).

3. En log(function<string()> f) añade una tarea (lambda expression) al worker_ para escribir el valor que devuelve la ejecución del lambda expression f (y salto de línea al final). Este método se usa para poder construir la información que queremos registrar en la hebra del worker_ (para no bloquear el juego si la construcción tarda mucho)

Inicializar el Logger en el método initGame() de la clase AsteroidsGame.cpp usando:

```
Logger::getInstance()->initInstance("log.txt");
```

Añadir al juego algunas llamadas para registrar información, por ejemplo:

```
// en GunIC.cpp: p es la posición y d es la dirección de la bala
Logger::getInstance()->log( [p,d]() {
    stringstream s;

    s << "Shooting: " << p << " " << d;
    return s.str();
});


// en Asteroids.cpp: al crear un asteroide, p es la posición y v es la velocidad del asteroide
Logger::getInstance()->log( [p,v]() {
    stringstream s;

    s << "New asteroid: " << p << " " << v;
    return s.str();
});

// en GameManager.cpp, cuando empieza/acaba una ronda
Logger::getInstance()->log("Round Start");
Logger::getInstance()->log("Round End");
```

Parte 3: Añadir agujeros negros al juego

Añadir *Agujeros Negros* al juego con el siguiente comportamiento:

1. Son objetos de tamaño 50x50, giran continuamente, pero no cambian de posición.
2. Inicialmente el número de agujeros negros es 2 y se duplican en cada ronda (es decir, en la segunda ronda son 4, en la tercera ronda son 8, y al final de la partida vuelven a ser 2).
3. Cuando empieza una ronda, colocamos los agujeros negros de manera aleatoria evitando la parte donde se encuentra el caza inicialmente.
4. Si el caza choca con un agujero negro muere el caza, si una bala choca con un agujero negro desaparece la bala, y si un asteroide choca con agujero negro desaparece el asteroide y reaparece inmediatamente en una posición aleatoria (evitando la posición actual del caza). Para dibujar un agujero negro se puede usar la imagen resources/images/black-hole.png (recuerda que para usar esa imagen hay que modificar `Resources.h` y `Resources.cpp`)

En esta parte hay que modificar lo mínimo posible del código de la práctica 4, es decir hay que definir un nuevo Container para los agujeros negros, mensajes correspondientes, etc.