# Ejercicio 4. Componentes.

A partir de la solución al ejercicio anterior, la idea es dotar de una mayor estructura a las clases creadas descomponiéndolas en componentes, así como de la creación del sistema de mensajería para la comunicación entre ellos.

Se utilizará como referencia la documentación asociada a la teoría impartida.

Ejercicio 4.1.

Creación de la clase abstracta asociada al concepto Componente.

Será una clase de la cual heredarán todos los componentes que se creen.

Contendrá la función Slot como principal método que sobrescribirán las clases hijas, la cual servirá para el procesado del componente.

Ejercicio 4.2.

Creación de la clase Entidad.

La clase contendrá una lista de Componentes. La idea es poder crear diferentes tipos de entidades a partir de la composición de componentes diferentes.

Contendrá los métodos principales:

* AddComponent. Inserta un componente.
* FindComponent. Obtiene un componente.
* Slot. Llama a las funciones correspondientes para cada uno de sus componentes.

Ejercicio 4.3.

La idea es descomponer la entidad Ball en componentes asociados a diferente funcionalidad. El código de dicha funcionalidad es el mismo que el del ejercicio 3. Los componentes a crear serían:

* Componentes asociado a la Colisión. Se encargará de detectar la colisión con otras Balls y con los límites del escenario.
* Componente asociado al Desplazamiento. Se encargará de actualizar la posición de la Ball.
* Componente asociado al Render. Se encargará del renderizado.

Cada Clase contendrá las variables miembro que necesite para operar, por ejemplo tanto la clase asociada a la Colisión como la del Desplazamiento contendrá una variable asociada a la posición.

La comunicación entre los componentes tanto de la misma Entidad como para la obtención de información entre diferentes entidades se hará a partir de la llamada a FindComponent en la clase Entidad.

Todo lo implementado servirá para sustituir la clase monolítica cBall por otra compuesta por sus diferentes componentes.

Ejercicio 4.4.

Introducir en cWorld una lista de punteros a Entidades, la cual sustituirá a la lista de Balls.

Creación en cWorld de las Balls con sus componentes, así como llamadas a su destrucción donde sea pertinente.

Probar que tras los cambios sigue funcionando la aplicación.

Ejercicio 4.5.

La idea es cambiar la comunicación entre Componentes de una misma Entidad para que cada componente no sepa de la existencia de los demás componentes y opere de manera independiente.

Para ello crearemos un sistema de mensajería simple, visto en la clase de teoría, a partir de los ejercicios siguientes.

Nota: La obtención de información entre Entidades diferentes seguirá gracias al método FindComponent.

Ejercicio 4.5.1.

Creación de la clase Mensaje. Los mensajes son contenedores de información, gracias a sus variables miembro, diferente según su funcionalidad.

Será la clase de la cual hereden los diferentes tipos de mensajes.

Creación de las diferentes clases asociadas a los diferentes tipos de mensajes a implementar:

* CollisionMsg: Asociada al mensaje que se crea cuando hay una colisión.
* EntCollisionMsg: Hereda de CollisionMsg y contiene información asociada a la colisión entre entidades.
* LimitWorldCollMsg: Hereda de CollsionMsg y contiene información asociada a la colisión con los límites de la ventana.
* NewPosMsg: Asociada al cambio de posición de la Entidad.

Dichos mensajes serán enviados por los componentes asociados para que los demás componentes estén informados, por ejemplo, el componente que detecta colisiones enviará un mensaje que será recibido por el componente que controla el desplazamiento para variar su trayectoria.

Ejercicio 4.5.2.

Implementaremos el envío del mensaje y recepción a partir de los siguientes pasos:

* Implementación dentro de la clase Componente del método ReceiveMessage.
* Implementaremos un nuevo método en la clase Entidad: SendMessage. Se encargará de llamar al ReceiveMessage de todas los componentes.
* Sustitución de las llamadas a FindComponent para comunicación entre componentes **de la misma Entidad** por las nuevas llamadas SendMessage.
* Las llamadas con FindComponent para obtención de información **de otras Entidades** se mantienen.
* Implementación de los métodos ReceiveMessage de los componentes asociados a la Colisión y al Desplazamiento, los cuales tendrán como argumento el mensaje de entrada y actuarán según el tipo de mensaje.

Probar que tras los cambios sigue funcionando la aplicación.