



MOTOR FÍSICO
SHIFT videogame

GAMELAB
UNIVERSIDAD DE OVIEDO

Alberto Carlos del Blanco Maraña
Madrid 28 de junio de 2007



Tabla de Contenidos

1. Introducción	4
2. Parámetros de la clase <i>object</i>	6
3. Parámetros de la clase <i>mesh</i>	9
4. Parámetros de la clase <i>physic</i>	10



Lista de Figuras

Imagen 1 : Límites del motor físico.....	4
Imagen 2 : Clase “ <i>object</i> ” como centro de un modelo radial	6
Imagen 3 : Objetos simples (carcasa y aspas)	6
Imagen 4 : Objeto compuesto (ventilador)	7
Imagen 5 : Diagrama jerárquico de instancias para el ventilador.....	7
Imagen 6 : Diagrama de instancias para dos ventiladores	9



1. Introducción

El presente documento detalla el módulo físico del videojuego SHIFT, como parte del proyecto fin de carrera realizado por Alberto Carlos del Blanco para la E.P.S.I.G. de la Universidad de Oviedo.

El módulo físico puede considerarse el módulo principal del motor gráfico. Permite gestionar los objetos en sí, definiendo su jerarquía de construcción, ubicación, geometría, texturas y parámetros físicos básicos.

Dentro del esquema general del motor gráfico, el módulo físico queda limitado por el ámbito rayado:

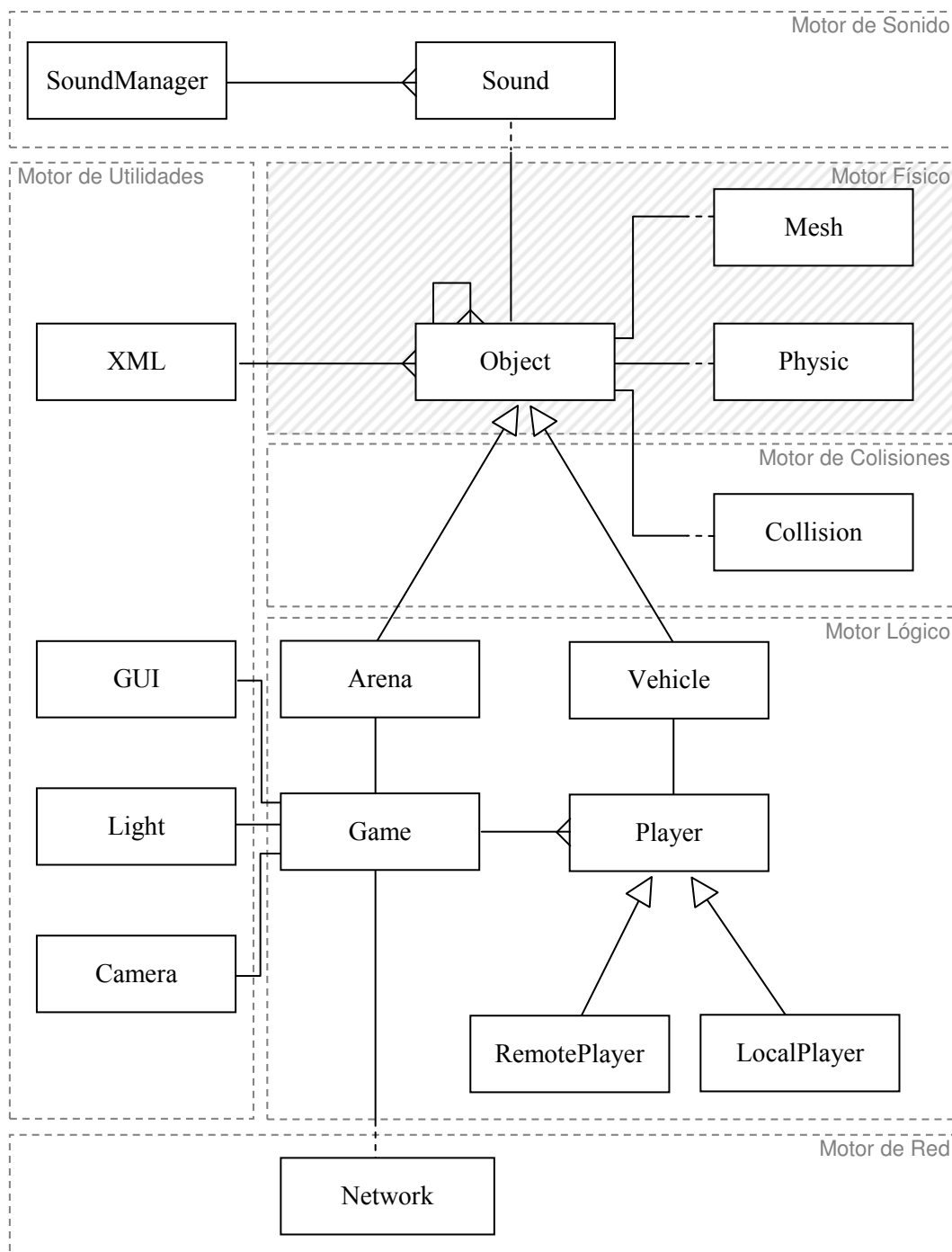


Imagen 1 : Límites del motor físico



Cómo refleja en el diagrama, el módulo físico consta de las siguientes clases:

- **Object**
Clase para abstraer el concepto de objeto dentro del motor gráfico. Es la clase que centraliza todas las propiedades técnicas y lógicas de cada elemento del videojuego.
- **Mesh**
Permite gestionar la geometría y texturas del objeto.
- **Physic**
Permite mantener los datos físicos del objeto, como son posición, escalado, rotación velocidades, aceleraciones y fricciones lineales y angulares, etc..

Por tanto este módulo implementa los objetos en sí, su ubicación en la escena, geometría, texturas y parámetros físicos que tendrá en cada momento del videojuego.

A continuación veremos en detalle cada uno de estos aspectos controlados por el motor físico.

2. Parámetros de la clase *object*

La clase que abstrae el concepto de objeto es el centro de un modelo de clases radial, dónde el resto de instancias están asociadas de algún modo a cada objeto.

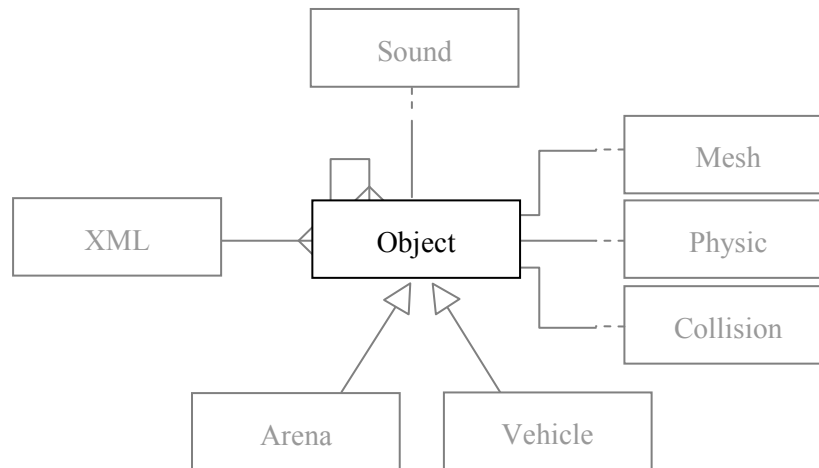


Imagen 2 : Clase “*object*” como centro de un modelo radial

Los objetos constituyen las piezas unitarias más pequeñas manejadas por el interfaz gráfico, piezas que tendrán que ser transformadas y renderizadas por separado para componer la escena. El motor físico está diseñado para establecer una composición jerárquica de objetos. De esta forma, varios objetos pueden componer otros más complejos, y cada uno de ellos por separado o todos ellos en conjunto pueden mantener diferentes propiedades físicas, de ubicación, colisión o sonido.

Es decir, cada objeto se puede construir jerárquicamente en base a otros más pequeños, dónde en cada nivel de la jerarquía se van aportando propiedades comunes para todos los objetos que pertenecen a esa rama. Esta estructuración de elementos ofrece una gran versatilidad para el resto del motor gráfico, ya que si desde un módulo se actúa sobre un objeto de la jerarquía, se estará actuando sobre todas sus dependencias.

Por ejemplo, un ventilador es un objeto del videojuego, compuesto por dos objetos gráficos más simples, la carcasa y las aspas, cada uno de ellos con propiedades específicas y en conjunto con algunas características comunes.

El aspa tendrá una velocidad angular constante y un sonido asociado referente a su movimiento rotativo.

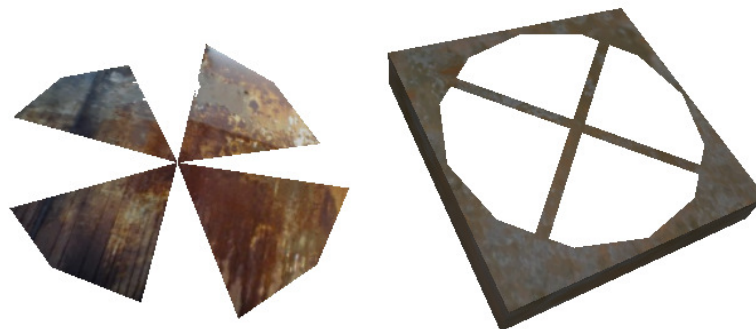


Imagen 3 : Objetos simples (carcasa y aspas)

En conjunto, carcasa y aspa, serán colisionables, tendrán una ubicación determinada (posición rotación y escalado) y una física concreta (fricciones lineales y angulares).

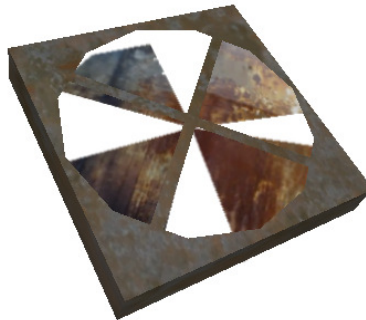


Imagen 4 : Objeto compuesto (ventilador)

Gráficamente, la jerarquía de instancias construida para el ventilador tendrá la siguiente forma:

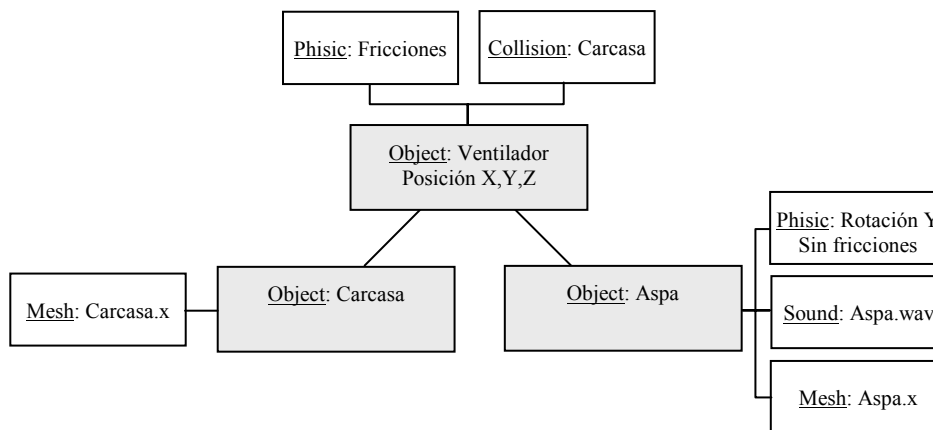


Imagen 5 : Diagrama jerárquico de instancias para el ventilador

En adelante podremos hacer referencia a cada objeto específico por separado o a los dos en conjunto como un todo. Es decir, si colisionamos sobre la instancia del ventilador, estaremos moviendo aspa y carcasa conjuntamente en la escena, y al moverse actuarán las fricciones lineales y angulares asociadas. Sin embargo, en el movimiento rotativo del aspa no intervienen esas fricciones, ya que esta instancia tiene sus características físicas propias que prevalecen sobre las del objeto padre.

Resumiendo, los objetos definen sus propiedades de una forma jerárquica, dónde en cada nivel podremos definir de forma opcional:

- **Ubicación**
Posición, rotación y escalado. Si no se indica se aplica la matriz identidad.
- **Geometría y Texturas**
Geometría y texturas asociadas al objeto.
- **Física**
Velocidades, aceleraciones y rozamientos lineales y angulares, así como la masa del objeto
- **Colisiones**
Área de colisión asociada al objeto
- **Sonidos**
Buffer de sonido 3D que se debe ubicar en el origen del objeto.



Además, cada objeto puede ser:

- **Un objeto simple**
Cuando no se compone a partir de objetos hijos
- **Un objeto compuesto**
Cuando se compone a partir de otros objetos hijos
- **Vehículo**
Objeto que implementa gran parte de la lógica del videojuego. Está compuesto por cinco objetos, un fuselaje y cuatro ruedas, y cada uno de ellos con una física concreta. Sus propiedades se definirán más adelante, en el documento del motor lógico.
- **Arena**
Objeto que implementa gran parte de la lógica del videojuego. Está compuesto por diferentes objetos encerrados bajo un firmamento. Sus propiedades se definirán más adelante, en el documento del motor lógico.



3. Parámetros de la clase *mesh*

Esta clase gestiona tres buffer por cada instancia:

- **Geometría** Como un conjunto de vértices, uniones y normales.
- **Materiales** Como colores sólidos asociados a ciertos vértices.
- **Texturas** Como un buffer de la imagen y un mapeado a vértices.

Una vez en memoria, cada uno de los tres buffer permanecerá constante, con acceso sólo de lectura, ya que la geometría, materiales y texturas de los objetos permanecerán constantes durante la existencia de cada objeto en escena, esto permitirá compartir buffer en memoria para todos los objetos que compartan geometría, materiales o texturas.

La geometría y materiales se cargan externamente desde un fichero DirectX (.X), mientras que las texturas se cargan aparte desde un fichero de imagen (.JPG), lo que permitirá identificar cada geometría, material y textura para que sólo sea cargada una única vez desde disco.

Teniendo en cuenta las dos consideraciones anteriores, cada fichero externo sólo será leído una vez, y cada buffer sólo residirá en memoria una única vez, esto para todos los objetos de escena que compartan geometría, materiales o textura.

Esta idea aporta gran eficiencia a la aplicación, ya que podremos cargar infinidad de objetos en escena, leyendo y manteniendo en memoria unos pocos modelos externos, todo ello implementado en el motor físico y de forma transparente al programar la lógica del videojuego.

Por ejemplo, siguiendo el diagrama de instancias, si añadimos un nuevo ventilador igual que el anterior tendremos el siguiente diagrama, dónde se comparten las instancias de la geometría del aspa y la carcasa. También se compartiría del mismo modo el buffer de sonido, pero esto se verá más adelante, en el documento de motor de sonido.

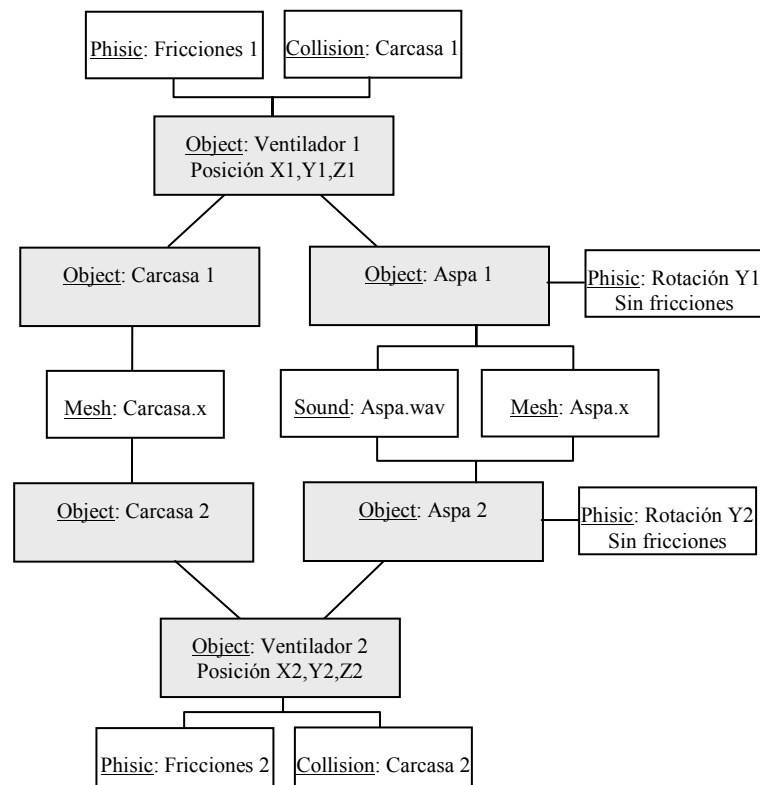


Imagen 6 : Diagrama de instancias para dos ventiladores



4. Parámetros de la clase *physic*

Esta clase centraliza los datos físicos del objeto, como son:

- Módulo de la velocidad lineal.
- Módulo de la aceleración lineal.
- Módulo del rozamiento lineal.
- Vector normalizado de la dirección de desplazamiento lineal.
- Velocidad angular en cada uno de los tres ejes.
- Aceleración angular en cada uno de los tres ejes.
- Rozamiento angular en cada uno de los tres ejes.

En adelante, el resto de módulos del motor gráfico actuarán sobre estos datos, modificándolos en cada iteración que el motor gráfico realiza sobre las instancias manejadas.

Alberto Carlos del Blanco Maraña
Madrid 28 de junio de 2007