



técnicas avanzadas de gráficos
ingeniería multimedia

Seminario 3

Gestor de recursos

¿Qué es un gestor de recursos?



¿Qué es un recurso?

¿Cómo debemos gestionar los recursos?



- En informática, los recursos son las aplicaciones, herramientas, dispositivos (periféricos) y capacidades con los que cuenta una computadora.
- Para un motor gráfico, los recursos son los elementos gráficos que permiten construir la escena:
 - Elementos geométricos: vértices, polígonos, mallas...
 - Imágenes: cielo, texturas...
 - Otros recursos: materiales, animaciones, ...
- Otros recursos, como el sonido, pueden manejarse de igual manera aunque no son propiamente del motor gráfico



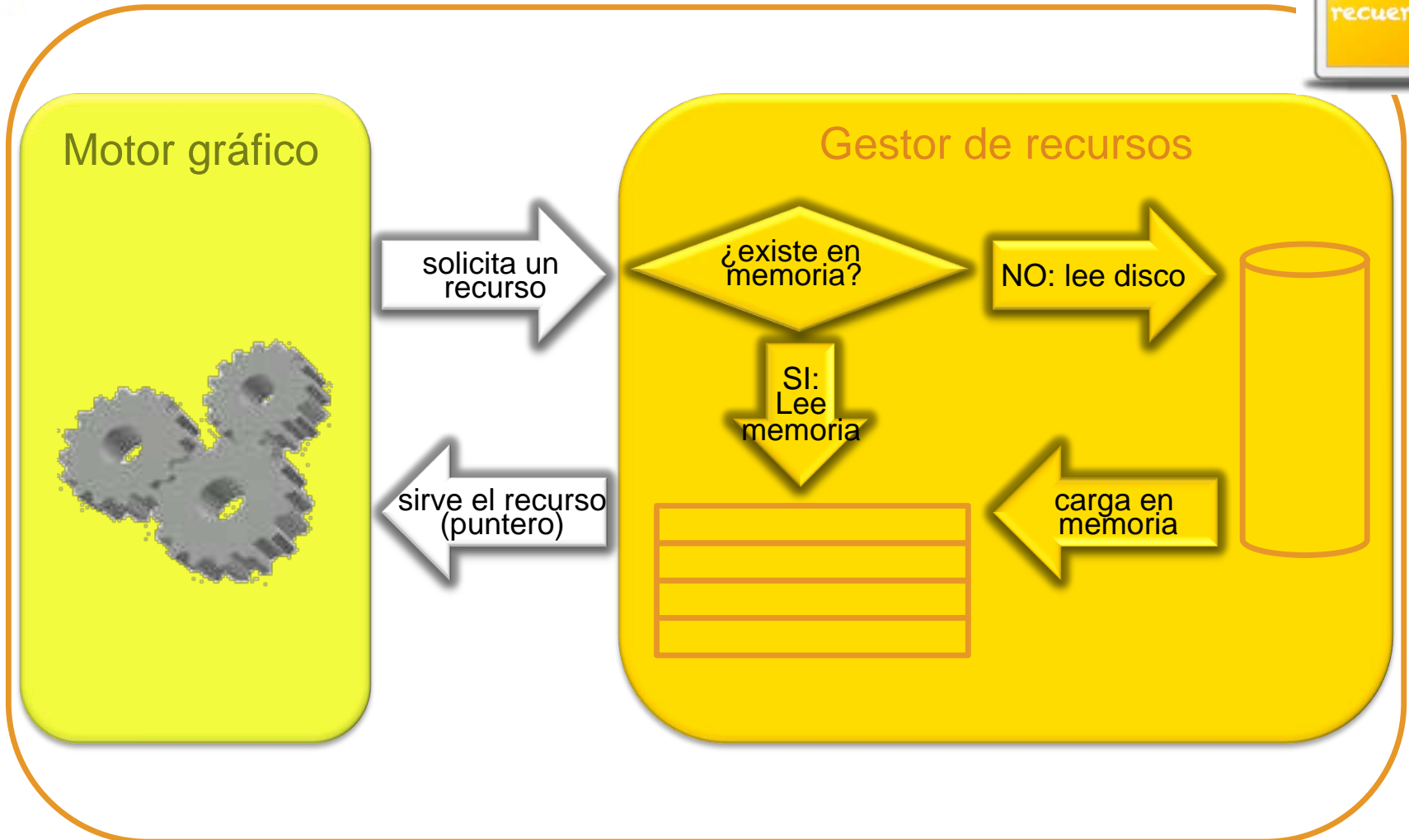
- Características de nuestros recursos:
 - Habitualmente se obtienen a partir de un fichero (mallas, texturas, animaciones...) y deben cargarse en memoria
 - Suelen ocupar mucho espacio en la memoria
 - Muchas veces se visualiza más de una instancia del mismo recurso
- Por lo tanto, su buena gestión es fundamental para que el motor sea eficiente, temporal y espacialmente

Gestor de recursos



- ¿Por qué un gestor de recursos?
 - Independiza el motor de los recursos concretos
 - Los recursos se leen de fichero. La lectura es lenta, por lo tanto debemos:
 - Evitar que un recurso se lea más de una vez
 - Leer en el momento oportuno y de la forma adecuada, para que el sistema no se pare esperando un recurso
 - Los recursos ocupan mucha memoria, por lo tanto debemos:
 - Optimizar el espacio que ocupan
 - Mantener una sola copia de cada recurso

Gestor de recursos





- Aspectos clave
 - ¿cómo servimos el recurso cuando se solicita?
 - ¿cómo se almacenan los recursos en el disco?
 - ¿cómo se cargan los recursos en memoria?
 - **¿cómo y cuándo gestionamos la lectura de disco y la carga en memoria?**

Solicitar y servir el recurso



- Cuando el motor gráfico (u otro módulo) necesita un recurso, invoca al gestor de recursos solicitándolo. Puede hacerlo llamando a una función del gestor, o mediante un sistema de mensajes
- Independientemente de cómo lo gestione el gestor de recursos, debe servir un identificador que permita acceder directamente al recurso cargado en memoria
- El identificador puede ser un identificador único propio del sistema, o un puntero a memoria

Leer el recurso del disco



- Los recursos, en general, estarán almacenados como ficheros
- Cada tipo de recurso utilizará el formato de fichero adecuado para ese recurso, por ejemplo, OBJ para objetos 3D, MTL para materiales, JPG para imágenes, WAV para sonidos...
- El gestor leerá el fichero y cargará el contenido en memoria. Para leerlo, podemos escribir nuestro propio *parser* o recurrir a librerías de terceros

El recurso en memoria



- El gestor debe gestionar la reserva y liberación de la memoria para almacenar el recurso
- Cada tipo de recurso necesitará de una estructura adecuada en memoria
- La estructura en memoria debe adaptarse al uso que se vaya a hacer del recurso, para que su gestión sea lo más eficiente posible
- Típicamente, el gestor mantendrá un almacén de recursos en memoria (un array, una lista...) listos para ser servidos

Gestión de los recursos



- Es la parte fundamental que dota de sentido al gestor de recursos
- La gestión puede ser:
 - Síncrona: el recurso se lee de disco y se carga en memoria cuando el motor lo solicita
 - Asíncrona: el recurso se lee de disco y se carga en memoria en un momento distinto de cuando el motor lo solicita

Gestión de los recursos



- ¿Qué ventajas e inconvenientes tiene una gestión síncrona?
- ¿Qué ventajas e inconvenientes tiene una gestión asíncrona?
- ¿Qué variantes en la gestión de los recursos podemos proponer?
- ¿Qué variante se adecúa más a cada tipo de aplicación?

Gestor síncrono



- El recurso se lee del disco y se carga en memoria cuando el motor lo solicita
- El motor espera a que se cargue el recurso para continuar
- Ventajas:
 - Sencillo de implementar
 - Bajas necesidades de memoria
- Inconvenientes:
 - Si hay muchos recursos o son muy grandes, el sistema se queda esperando

Gestor asíncrono



- El recurso se lee del disco y se carga en memoria en otro momento distinto de su solicitud: por anticipado o en diferido
- El motor no espera a que se cargue el recurso: si está disponible lo utiliza y si no, continua y lo incorpora cuando esté disponible
- Ventajas:
 - El sistema nunca se queda esperando
- Inconvenientes:
 - Es más complejo de implementar
 - Puede tener altas necesidades de memoria

Gestor síncrono simple



- Ante una solicitud de recurso:
 - Si el recurso no está en memoria
 - Lee el recurso de disco y lo carga en memoria
 - Se sirve el recurso
- El sistema se queda esperando a que se sirva el recurso
- Los recursos repetidos sólo se leen una vez de disco, la primera vez que son requeridos
- Adecuado para sistemas sencillos, con recursos pequeños y rápidos de leer

Gestor asíncrono simple

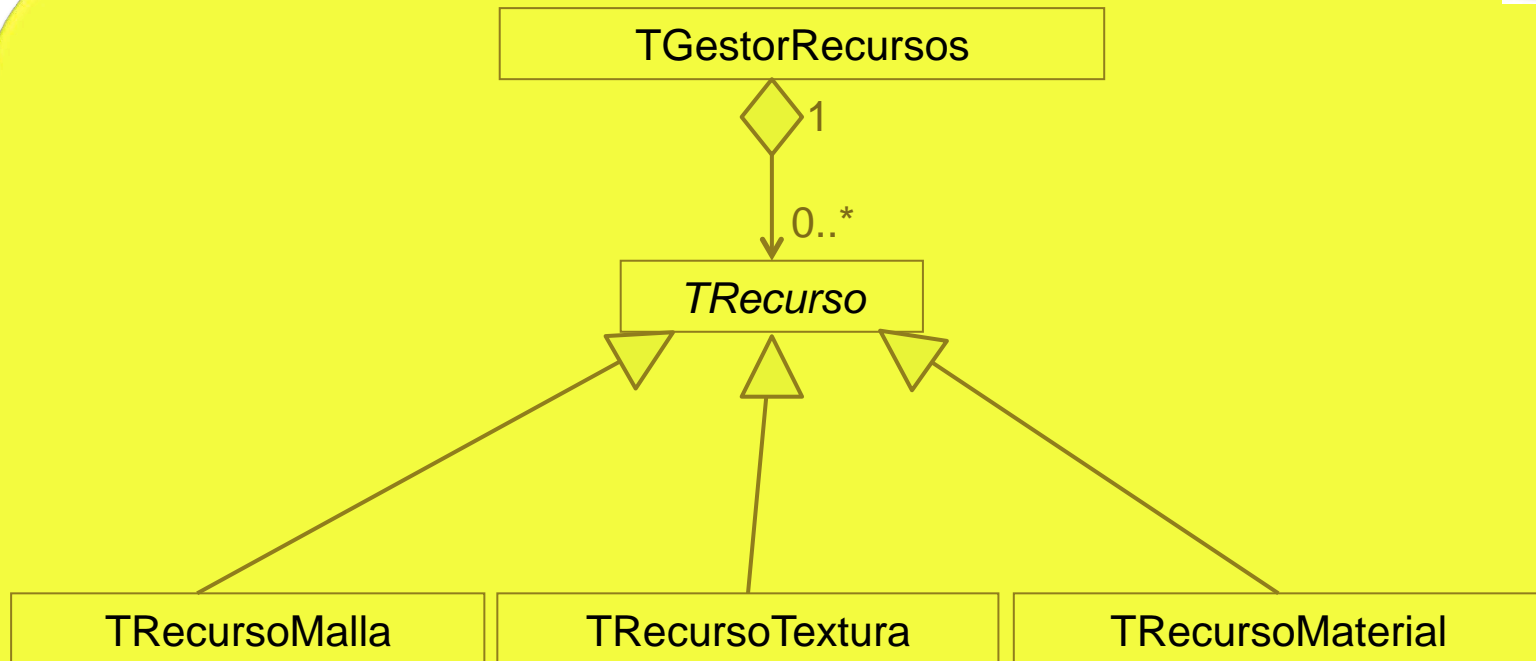


- Ante una solicitud de recurso:
 - Si el recurso no está en memoria
 - Se lanza un hilo que lee el recurso de disco y lo carga en memoria
 - Se sirve el recurso cuando esté disponible
- El sistema no espera a que se sirva el recurso, sino que trabaja con recursos vacíos mientras no esté disponible
- Los recursos repetidos sólo se leen una vez de disco, la primera vez que son requeridos
- Adecuado para que puedan trabajar con recursos vacíos



- Durante tiempo de carga del sistema, se cargan los principales recursos de forma síncrona
- Durante la ejecución, los nuevos recursos se cargan de forma asíncrona
- Hay que hacer un estudio sobre las necesidades del sistema y la disponibilidad de memoria, para decidir qué recursos se cargan al principio o cuáles durante la ejecución
- Podemos aprovechar tiempos de espera (entre niveles, entre decisiones del usuario...) para cargar los recursos
- Es el modelo más general (podemos parametrizarlo para equilibrar la parte síncrona y la asíncrona) y el más adecuado para sistemas grandes

Estructura del gestor



TGestorRecursos



TGestorRecursos

```
vector<TRecurso*> recursos;
```

```
TRecurso *getRecurso (char *nombre);
```

```
TRecurso *TGestorRecursos :: getRecurso (char *nombre)
{
    TRecurso *rec = Buscar el recurso en el vector
    si rec es null
        rec = crear nuevo TRecurso;
        rec->cargarFichero (nombre);
        recursos->Añadir(rec);
    fin si
    return rec;
}
```



TRecurso



TRecurso (virtual)

```
char * nombre;
```

```
char *GetNombre ();  
void SetNombre (char*);
```



TRecursoMalla

```
float* vertices, normales, texturas;  
float* vertTriangulos, normTriangulos, textTriangulos;  
long nTriangulos;
```

```
cargarFichero (char *nombre);  
draw (...)
```

Métodos de TRecursoMalla



- Método cargarFichero
 - Lee el fichero con el recurso y rellena los buffers de datos (vértices, triángulos, texturas...)
 - Para la lectura del fichero podemos implementar un parser propio o utilizar librerías de terceros
- Método draw
 - Vuelca los buffers de datos en OpenGL
- Consejo: utilizar buffers de datos adaptados a los que maneja OpenGL:
 - La carga puede ser menos eficiente (pero se ejecuta en tiempo de carga)
 - La visualización tiene máxima eficiencia (más crítica)