Práctica: Dominó

Sistemas Gráficos Interactivos

Máster en Ingeniería Informática

2015-2016

Autor:

Gorka López

Índice

1. Objetivos

1.1. Objetivo

El trabajo realizado está centrado en torno a la caída de fichas de dominó, creando las físicas requeridas para ello. Estas físicas corresponden a la caída de cada ficha y la colisión que realizará cada una con la siguiente. Esta secuencia comienza en una ficha concreta, donde el usuario puede elegir en qué punto golpear para comenzar.

La aplicación desarrollada tiene dos estados: creador de recorridos y ejecución. El primero es una pantalla donde ir colocando piezas de dominó para formar un recorrido, mientras que el segundo cambia de editar a derribar las piezas, donde el usuario golpea la pieza que quiera, como se ha descrito anteriormente.

1.2. Motivación personal

La idea principal era la de crear una aplicación que tocara la mayor parte de conceptos tratados en la teoría de la asignatura, entre otras cosas. La elección del tema del dominó fue una sugerencia para tratar el tema de las físicas en objetos, convirtiéndose en el eje central de la aplicación. A partir de ahí, se han introducido algunas funcionalidades que jueguen con otros conceptos: coordenadas, materiales y texturas, cámara y sus controles...

1.3. Objetivos específicos

En un principio, se definieron tres fases para la aplicación a desarrollar. La primera fase era la creación de la aplicación base, es decir, sin lo definido en ello no habría aplicación. La segunda consistía en mejoras básicas para la aplicación, o las mejoras más críticas/necesarias en la aplicación. La tercera eran mejoras más avanzadas con el propósito de dar más posibilidades al usuario.

Fase 1:

- Físicas de caída y colisión de fichas de dominó.
- Posibilidad de escoger posición y fuerza preestablecidas de "golpe" en la primera pieza.
- Primera pieza creada por defecto, no se puede cambiar.
- Crear piezas, colocarlas seleccionando la posición con el ratón, todas a la misma altura.
- Cámara móvil, cuya posición se pueda cambiar constantemente.

Fase 2:

- Poder rotar la cámara, mediante teclas o clic+ratón (botón a definir).
- Poder acercar y alejar la cámara mediante teclas o rueda del ratón.
- Poder seleccionar libremente en qué parte de la primera pieza golpear y con qué fuerza, esta última acotada con valores mínimo y máximo.
- Posibilidad de rotar las piezas colocadas, cogiendo como eje el centro de la pieza.
- Posibilidad de cambiar el tamaño de las piezas colocadas.
- Control de posición de las piezas en el editor, no puede haber más de una pieza en el mismo espacio. Posibilidad de establecer una separación mínima entre las piezas.

Fase 3:

- Colocación rápida de fichas: en una distancia recta, piezas cada cierta distancia; en semicírculo con un radio, una cantidad de piezas...
- Diferentes alturas: Escaleras de piezas hacia arriba o abajo, plataformas simples para colocar las piezas en altura...
- Otros objetos golpeables con físicas: Esferas, cilindros...
- Otras opciones de personalizar las piezas: colores, materiales...

En la aplicación final, se cumplen bastantes de estos objetivos, cambiando algunos de estos por cuestiones relacionadas con la tecnología utilizada .

1.4. Plan de trabajo

No se ha seguido ningún plan de trabajo concreto, ni ningún tipo de seguimiento. Se ha ido avanzando en la práctica mediante el cumplimiento de objetivos, según su importancia. Por ejemplo, los primeros cuatro objetivos han sido los siguientes, en este orden: primero se han creado fichas con físicas, creando una fila pequeña de fichas para probarlo; después se ha creado el poder colocar fichas mediante clic del ratón; el tercer paso ha sido hacer funcionar que se aplique impulso a una pieza seleccionada; y por último se han separado el segundo y tercer objetivos en dos estados distintos.

2. Técnicas usadas

En este apartado se describe la práctica realizada. Por ello, antes de nada, estas son las librerías utilizadas:

- **Three.js**, versión 73.
- **FirstPersonControls.js**, la cual ha sido editada para cambiar algunos funcionamientos del ratón, como se explica más adelante en este documento, en el apartado de cámara.
- **dat.gui**, para crear interfaces simples.
- **physi.js**, librería principal de físicas. Requiere la librería **ammo.js** y el fichero adicional **physijs_worker.js**.

2.1. Objetos

Descripción

Sólo se han utilizado cuatro tipos de objetos distintos: Plano y tres tipos de piezas.

- **Plano**: plano de physi.js para las colisiones de las piezas con este, y que no se caigan al vacío.
- **Piezas Mesh**: instanciadas como objeto de Three.js, para utilizar en el editor, sin que se caigan ni interfieran con otras fichas y así poder editar sin complicaciones.
- **Piezas Physi**: objetos de physi.js con colisiones. Son las piezas que aparecerán en el modo ejecución, para poder colisionar como está indicado en los objetivos.
- **Piezas *Glow***: dos piezas semitransparentes creadas mediante shaders, una azulada-blanca y otra verde. Se utilizan para saber qué pieza se ha elegido o a cual se está apuntando en el editor.

Sólo hay un plano y dos piezas Glow, pero múltiples piezas Physi y Mesh, tantas como quiera el jugador, por lo que ha habido que agruparlas para gestionar los eventos de la aplicación. Las fichas Mesh estarán guardadas en dos grupos, el principal de *fichas_sinf* (de sin fisicas) donde se guardarán por defecto, y el temporal de *fichas_temp*, donde se guardarán en una ocasión concreta, explicada más adelante. Por otro lado, las fichas Physi estarán en un array llamado *fichas*, puesto que había problemas para guardarlas en un grupo.

Descarga

La mayoría del código de la aplicación se ha obtenido mediante pruebas, documentación y discusiones de internet (github, stackoverflow...). Los objetos utilizados tienen tres fuentes distintas:

- Documentación de Three.js, en caso de las fichas Mesh.
- Documentación de Physi.js, en caso de las fichas Physi y del plano.
- Las fichas Glow se han obtenido a partir de la demo del siguiente enlace: https://stemkoski.github.io/Three.js/Shader-Glow.html

Movimientos

Los objetos solo tienen movimientos en sí cuando está ejecutando:

- Al aplicarle impulso a una pieza, el movimiento que se ocasiona
- Al ser golpeada por otra pieza.

En el modo edición es posible editar la posición de cualquier pieza seleccionada:

- Con un slider de dat.gui para mover a lo largo de todo el plano.
- Con teclas (i,j,k,l) para moverla en los ejes X y Z, con mayor precisión.

2.2. Escenario

La presentación de esta aplicación es muy simple: un fondo uniforme de color grisáceo claro, en lugar de blanco, y un plano, con la cámara enfocando al centro, donde por defecto habrá algunas piezas de dominó creadas. Sólo aparecen los objetos indicados en la sección anterior del documento, con un par de restricciones:

- Las piezas Glow sólo aparecerán si hay fichas Mesh, en función de los eventos descritos más adelante en este documento.
- Nunca estarán en pantalla a la vez las piezas del grupo *fichas_sinf* y las del array *fichas*. Si ha de haber fichas Mesh junto piezas Physi, las primeras serán las del grupo *fichas_temp*.

2.3. Luces

En la aplicación sólo se han utilizado dos fuentes de luz, la ambiental y un foco de luz, tal y como se describe a continuación:

- Luz ambiental de color claro pero poco intenso, para que pese a las sombras generadas por el foco de luz sea posible ver y distinguir las caras de cada pieza.
- Foco de luz blanco en las coordenadas (400, 1200, 200) y apuntando hacia el centro (0,0,0). Al estar en esa posición, ilumina todo el plano sin dejar zonas demasiado oscuras; y al estar alejado del centro, crea distintas sombras en las piezas.

2.4. Texturas y Materiales

Por parte de las texturas, se han utilizado dos tipos distintos:

- Una textura de madera para el plano, obtenida de una demo de Physi.js, accesible desde el siguiente enlace: https://chandlerprall.github.io/Physijs/examples/jenga.html
- 24 texturas de fichas distintas, obtenidas a partir de una imagen mayor. Se cargan al principio y se van asignando de manera aleatoria a las piezas según se crean.

Por otro lado, los materiales utilizados son todos materiales Lambert:

- El material del plano es la textura indicada en un material Lambert, donde se repite la textura para cubrir todo el plano.
- Las fichas utilizan FaceMaterial, para utilizar un array de materiales (uno para cada cara):
  - Material Lambert con una textura de ficha de dominó. Mediante prueba y error, se ha determinado que la cara que se buscaba es el cuarto material del array utilizado para crear la pieza.
  - El resto de posiciones del array, utilizan un material Lambert del mismo color, aleatorio o definido por el usuario.

2.5. Cámara

Se ha creado una cámara normal a una altura suficiente como para ver todo el plano, la cual utiliza los controles *FirstPersonControls* mediante una librería modificada para conseguir los resultados deseados:

- Se han eliminado los eventos de clic de ratón para acercar y alejar la cámara. Es molesto que se acerque la cámara cuando se quiere realizar otra acción de clic, como elegir mediante clic una pieza, y al no ser necesario, se han eliminado estos eventos.
- Se ha añadido un evento de rueda de ratón, para mover la cámara en su eje Y, al igual que las teclas por defecto R y F, las cuales se siguen manteniendo. Este evento es un poco complejo de añadir y manejar, ya que todavía no está demasiado estandarizado, al parecer: según el navegador, es necesario vincularlo de un modo u otro, incluso cambia el cómo coger el movimiento realizado.

En el código principal, se ha añadido mayor funcionalidad a la cámara: por defecto, la cámara estará bloqueada, es decir, no se podrá mirar alrededor mediante el movimiento del ratón, para así facilitar el uso de la aplicación. Sin embargo, al mantener pulsado el clic derecho del ratón, la cámara se desbloqueará y se podrá mirar alrededor, bloqueandose de nuevo al soltar el botón.

2.6. Eventos

En esta aplicación, los eventos tienen mayor peso que otras partes, aunque muchas veces son eventos repetidos bajo distintas situaciones. Es por ello que se explicaran de manera general los más importantes en esta sección, sin entrar en demasiado detalle de cuándo se dan. Los eventos están ordenados según el evento del navegador donde ocurren, en inglés.

Mousedown

El evento más simple, sólo desbloquea la cámara si se ha presionado el clic derecho del ratón.

Mouseclic

El evento de clic del ratón, o cuando se suelta el botón del ratón. Abarca varios eventos distintos, aunque se utiliza la función Raycast en varias ocasiones. Esta función devuelve la lista de objetos con los que "colisiona" el ratón y en qué punto lo hace, objetos de un array que hay que pasarle como parámetro.

- Bloquear la cámara, si el evento lo ha disparado el botón derecho del ratón. Se dará al soltar este botón, después de lanzarse el *mousedown* al pulsarlo.
- Si la aplicación está en modo ejecución, aplicará una fuerza a una pieza Physi, utilizando un Raycast y la función de Physi.js *applyCentralImpulse*, pasándole el vector de fuerza. Este vector de fuerza se le pasa descomponiendo la fuerza a ejercer (seleccionable desde una interfaz dat.gui) en los componentes X y Z, en función del ángulo Y de la pieza.
- Si la aplicación está en modo edición y colocar piezas, se utiliza un Raycast hacia el plano (utilizando un array que sólo contiene el plano) para sacar el punto de colisión, y colocar ahí una pieza. Si se ha colocado una pieza, no se podrá volver a colocar otra sin haber hecho clic en otro lugar, para evitar colocar por accidente. También se realizará este Raycast al plano para mover una pieza al punto seleccionado, si se elige en algún menú la opción de mover por clic.
- En caso de estar en edición y selección, al clicar en una pieza Mesh, utilizando un Raycast, se colocará el Glow verde en las coordenadas y rotación de la pieza seleccionada, escalando la geometría en función de la diferencia de tamaño. También se creará un menú para editar la pieza. El Glow y el menú también se crearán al colocar una ficha nueva, como si se hubiera seleccionado.

Mousemove

Cuando se esté en modo edición y selección, el Glow azulado se irá colocando en la pieza por la cual pase el ratón, utilizando un Raycast para ello. Este Glow, al igual que el verde, se adaptará en posición, rotación y tamaño a la pieza correspondiente.

Keydown

Se han añadido varias teclas para cambiar la posición y rotación de una ficha seleccionada, así como de los Glows para que mantengan la relación con la ficha:

- **I**: mueve la ficha hacia el lado negativo del eje X.
- **K**: mueve la ficha hacia el lado positivo del eje X.
- **J**: mueve la ficha hacia el lado positivo del eje Z.
- **L**: mueve la ficha hacia el lado negativo del eje Z.
- **Q**: rota la ficha en el eje Y en sentido agujas del reloj.
- **E**: rota la ficha en el eje Y en sentido contrario agujas del reloj.

Excepciones

Se ha añadido una excepción a los eventos de clic y movimiento: cuando se hace clic o el ratón pasa por una interfaz dat.gui, no se realizará ninguna tarea, con el fin de que no ocurran sucesos inesperados. Para ello, se crean eventos para las interfaces, con mayor prioridad al ser para un objeto en lugar de toda la página, donde una variable cambiará su valor a 1, a modo de barrera. Así, al llegar a los eventos principales, si esa variable tiene el valor de bloqueo, 1 en este caso , se cambiará al otro valor, 0, y no se realizará ninguna tarea del evento.

3. Problemas

A lo largo del desarrollo, se han dado varios errores, los cuales se plantean en esta sección:

- **Impulsos y fuerza:** ha requerido un gran esfuerzo hacer que funcione. Está implementado con la función de aplicar impulso central. Se han encontrado los siguientes problemas y soluciones:
  - Los objetos tienen que tener masa o no se les aplica impulso.
  - Problemas para aplicar impulso a una pieza en un punto exacto, en función del punto seleccionado mediante el clic. Como solución, se ha utilizado impulso central siempre hacia el mismo sentido de la pieza, el de la cara de ficha de dominó, utilizando esta cara a modo de decoración y de indicador de hacia dónde cae.
  - Problemas con los ángulos de las piezas physi. Si se cogía durante la ejecución el valor de la rotación del eje Y, daba resultados inesperados, con su derivado malfuncionamiento. Esta propiedad funciona correctamente al crear las piezas o en modo editar, así que se ha añadido otro campo a los objetos physi de las piezas, para que guarden esta rotación, y así utilizarlo al aplicar impulso.
- El listener **mouseup** no funcionaba con el código requerido. Sin embargo, el listener de **mouseclic** si que funciona, así que se ha utilizado este último.
- Al tener las fichas Physi en un grupo, ocurrían malfuncionamientos, como no poder aplicarles impulso. Esto, añadido a que cada vez que se ejecuta hay que recrear las piezas para que estén en su estado inicial, ha favorecido la elección de utilizar un array para estas piezas en concreto.
- **Dat.gui** tiene varios problemas de implementación, ajenos a la aplicación, pero que han afectado:
  - No funciona adecuadamente el "step" del slider al trabajar con decimales, por ello, se han utilizado grados en lugar de radianes, y se han aumentado los valores decimales utilizado (por ejemplo, por diez).
  - El color picker tiene un problema importante, donde a veces la pieza seleccionada se queda de color negro y no se puede cambiar. En la última versión de la aplicación, esto pasa con todas las piezas, por lo que se ha borrado la funcionalidad de cambiar color de piezas creadas. No obstante, se puede elegir el color deseado antes de crear una pieza.

4. Conclusiones

La librería Three.js junto con otras librerías que la complementan, como Physi.js, ofrecen la oportunidad de crear aplicaciones gráficas sin entrar demasiado en profundidad en el funcionamiento interno. Esto es una ventaja para crear aplicaciones, pero una desventaja en el ámbito educativo, ya que se pierde la oportunidad de trabajar a ese nivel.

En el contexto de la asignatura, donde hemos visto en la teoría varios conceptos matemáticos, el no poder manejar el funcionamiento interno antes mencionado impide ver con mayor claridad cómo se está aplicando cada concepto. No obstante, al olvidarse de esto, permite utilizar también conceptos de mayor nivel como los shaders o utilizar texturas, sin profundizar en exceso, con lo cual ofrece oportunidades en ese aspecto.

Resumiendo, se puede decir que la librería se puede utilizar para aplicar conceptos de mayor complejidad, olvidándose de funcionamientos matemáticos.