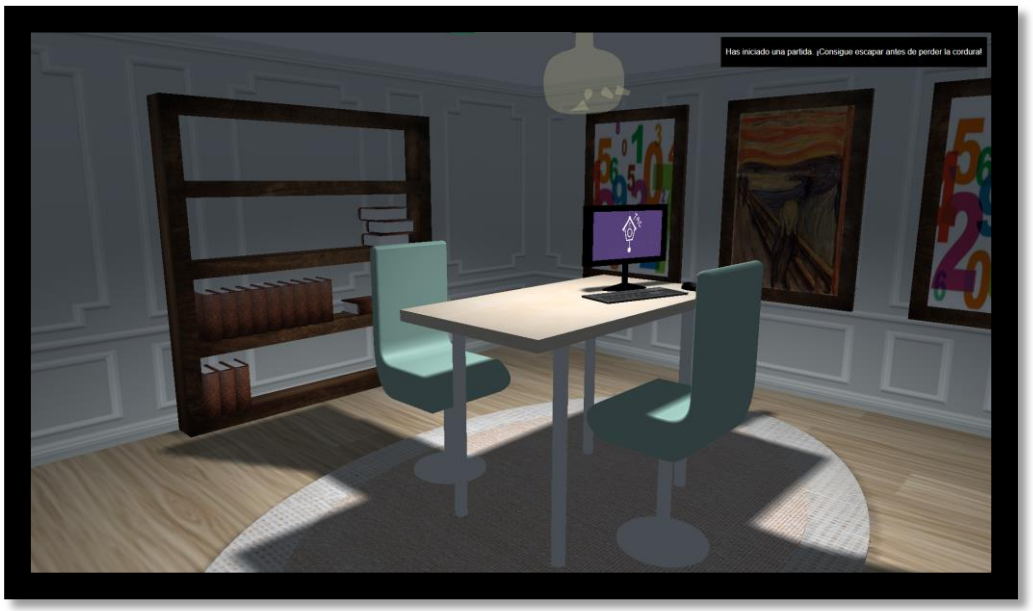
PRÁCTICA 2 - SCAPE ROOM

Autores:

Alba Guisado Farnes

Denis Valentin Stoyanov D'Antonio

"La casa de los secretos"



ÍNDICE

4. Interacción con la escena.

- **4.1. Cámara y movimiento.**
- **4.2. Colisiones.**
- **4.3. Picking.**
- **4.4. Mensajes permanentes.**

5. MyScene.

- **5.1. Componentes básicos no descritos anteriormente.**
- **5.2. Funciones auxiliares utilizadas.**

6. Diagrama de clases.

7. Referencias a modelos 3D y sus respectivas librerías.

8. Vídeo del juego.

1. Descripción general.

Nuestro ScapeRoom, llamado “La casa de los secretos”, se ambienta en una sola habitación. Al iniciar el juego, podremos observar la habitación delante nuestra, quedando a nuestras espaldas la puerta que tanta ansia tendremos por abrir. Además, observaremos el siguiente mensaje en la parte superior derecha: “Has iniciado una partida. ¡Consigue escapar antes de que te descubran!”

Esta sección de mensajes será muy importante, ya que irá actualizándose dependiendo del estado del juego en el que nos encontremos.

Hay bastantes objetos que llaman la atención de la habitación, los cuales parecen estar agrupados por zonas. ¿Será cada zona una pista? ¿Tendrán algún orden específico en el que ser explorados? Estas preguntas son respondidas en el apartado “Cómo completar el juego”.

Algunos de los objetos de los que se habla son:

- Una mesa con un ordenador.
- Una estantería con libros.
- Tres cuadros.
- Tres interruptores.
- Una caja grande con botones.
- Una caja pequeña con una calavera encima.
- La tan ansiada puerta.

Todos estos objetos y los restantes no mencionados son explicados paso a paso en la sección “Construcción de la escena -> Objetos utilizados”.

2. Cómo completar el juego.

Antes de entrar en el tema técnico de cómo se ha construido todo el juego, vamos a explicar cómo podemos completarlo; por si alguien quiere lanzarse a lo loco a disfrutar del mismo sin hacer mucho hincapié en el apartado técnico.

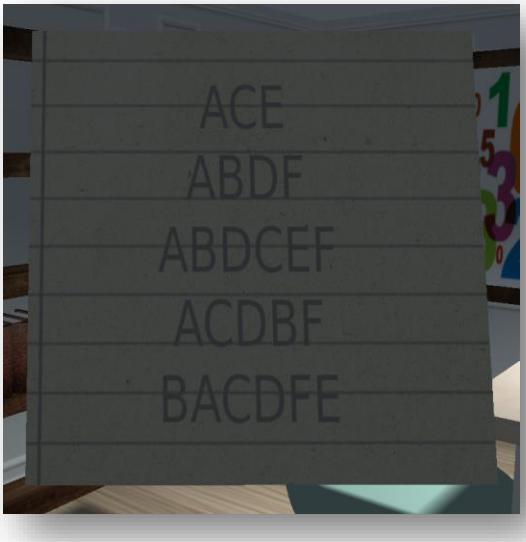
El juego consta de 3 pistas a completar antes de poder abrir la puerta para salir de la habitación. La dificultad del ScapeRoom no radica tanto en encontrar las pistas sino en cómo enlazarlas y entender para qué sirven cada una de las mismas.

2.1. Pista número 1.

Al comenzar el ScapeRoom, como hemos comentado anteriormente, podremos ver 4 zonas claramente distinguibles: “La mesa con el ordenador”, “La estantería”, “Los cuadros” y “La caja con botones”.

Pues bien, **lo primero que deberemos hacer será dirigirnos a la estantería**. En ella nos encontraremos un total de 18 libros, agrupados y distribuidos de formas distintas entre sí.

Podremos interactuar con ellos, pero no será hasta cuando **pulsemos un libro solitario tumbado en la segunda leja** que **nos aparecerá una nota**. **Dicha nota contiene seis combinaciones de letras: “ACE”, “ABDF”, “ABDCEF”, “ACDBF” y “BACDFE”**. Además, se nos actualizará el mensaje de la parte superior derecha, mostrándonos lo siguiente: “Un libro algo sospechoso. Dentro se encontraba esta nota: ACE, ABDF, ABDCEF, ACDBF, BACDFE”.



Esta es la primera pista, llegados a este punto disponemos de las combinaciones de letras, las cuales aún no sabemos para qué sirven ni dónde podrían usarse.

2.2. Pista número 2.

Una vez partimos de que hemos encontrado la nota, llega la pista más complicada, no tanto por encontrarla; sino por entender cómo relacionar la pista anterior con esta.

De nuevo, hablamos de secciones; y es que ahora **debemos irnos a la sección de los cuadros**.

Aparentemente, los cuadros no presentan nada especial. Hay dos iguales, los cuales tienen números; y uno en el centro, el cual contiene una imagen distinta (El grito de Edvard Munch). Esto puede hacer pensar al jugador que el del centro es el especial, pero esto realmente no es así.

Para encontrar la siguiente pisa, **deberemos hacerles caso a los interruptores de la derecha**. Al pulsar cada uno de ellos, se enciende una luz que apunta a un solo cuadro. Hay tres tipos de luces: azul, amarilla y roja.

Aparentemente no cambia nada en los cuadros pero, si **nos fijamos en el cuadro que está siendo iluminado por la luz roja**, nos encontraremos con que hay algo extraño en él. **Ha aparecido, proyectado por el foco, una extraña figura. Dicha figura forma un rectángulo con una línea seccionándolo en dos; así como las siguientes letras A, B, C, D, E, F. Estas letras están algunas unidas entre sí con las líneas**.

Aquí llega el punto complicado del ScapeRoom, enlazar la pista 1 con la pista 2. Para ello, lo que deberemos de hacer será **darnos cuenta de que si seguimos el orden de las letras de la nota mirando el cuadro y las líneas que se forman, veremos cómo van dándonos una combinación de números**.

Así pues, al empezar con la primera combinación "ACE", veremos que al unir A con C y C con E, formamos una especie de línea recta hacia abajo, la cual puede simular un 1.

Tras hacer lo mismo con las combinaciones: "ABDF", "ABDCEF", "ACDBF" y "BACDFE"; obtenemos los números: "7", "2", "4" y "5", respectivamente.

Quedándonos así finalmente la cadena "17245".

Ahora bien, estamos en un punto parecido al que estábamos al acabar la pista 1. Tenemos una combinación de números, pero ¿dónde se usa?, ¿para qué sirve?

2.3. Pista número 3.

Finalmente, una vez disponemos de la combinación de números, podremos **dirigirnos a** la tercera de las cuatro zonas diferenciadas; esta es, **la caja con botones**.

En ella, **deberemos introducir correctamente dicha combinación pulsando los botones correspondientes**. Una vez introducida la cadena, **se abrirá la tapa de la caja superior, dejándonos ver un pendrive en su interior**.

Al hacer clic sobre el pendrive, nos dirá que lo hemos cogido, pero sin decirnos para qué sirve. Aquí tendremos que tirar de ingenio para deducir que **debemos clicar sobre la pantalla del ordenador**, la cual se encuentra en la cuarta y última zona, “la mesa con el ordenador”.

Una vez clicado, la pantalla del ordenador cambiará, mostrándonos una imagen de “The End”; así como el mensaje en la esquina superior derecha, diciéndonos “¡Lo has conseguido! Ya puedes salir por la puerta”. Pues bien, esto es justo lo que debemos de hacer.

En este punto, si clicamos en el pomo de la tan ansiada puerta, por fin abrirá la puerta. Tras completar la animación que dura exactamente dos segundos, se nos aparecerá un pasillo. Si entramos encontraremos un botón “restart”, al hacer clic sobre él, la página del juego se reiniciará.

2.4. Resumen de cómo resolver el juego.

Primero, debemos ir a la estantería y hacer clic sobre el libro solitario de la segunda leja, el cual nos dará cinco combinaciones de letras.

Tras esto, deberemos encender el interruptor superior, que enciende una luz roja que proyecta una imagen sobre el cuadro. Dicha imagen, junto con las letras, nos traduce las combinaciones de letras a unos números.

A continuación, deberemos clicar los botones de la caja grande en el orden de la cadena obtenida gracias a la traducción del cuadro. Con esto, se abrirá la caja y podremos ver un pendrive en su interior.

Al hacer clic sobre el pendrive, podremos hacer clic sobre la pantalla; cambiando el vídeo que aparece en ella.

Con estos pasos completados, podremos ir a la puerta, pulsar el pomo y ver cómo la puerta se abre y se nos muestra un mensaje de que hemos ganado. Es posible reiniciar el juego si entramos en el pasillo que aparece y pulsamos el botón que se encuentra al final.

3. Construcción de la escena.

En este apartado se va a explicar más técnicamente todo lo que hemos tenido que utilizar para crear la escena que se puede observar en el juego. Esto incluye, el html y todas las clases JavaScript con la biblioteca three.js. Html solo hay uno, pero clases de js si hay más; las cuales serán todas explicadas, entrando en más o menos detalle según lo requiera cada clase.

Los temas de interacción que haya en cada una de las clases se dejarán para el apartado "Interacción con la escena".

3.1. Index.html.

Como se ha dicho anteriormente, solo utilizamos un archivo html. Este es "Index.html", y en él podemos observar dos secciones claramente distinguidas: el head y el body.

En el head, cargamos nuestro archivo principal MyScene.html.

En el body, cargamos los dos vídeos que más tarde utilizaremos para cargar como textura de la pantalla del ordenador. Estos se encuentran en un contenedor el cual hemos escondido para que no aparezcan fuera de la escena, "display:none". También tenemos el cuadro de texto permanente dónde actualizaremos el contenido de este.

3.2. Objetos utilizados.

En esta sección, se van a describir todas las clases js que contengan la creación de al menos un objeto que aparezca en la escena. Como se ha comentado antes, si en alguna de estas clases hubiese algo de tema interacción o luces, se dejaría para el apartado número 4, "Interacción con la escena".

Por ordenar los objetos de alguna forma, la explicación de los mismos se va a hacer en orden alfabético.

3.2.1. Alfombra.js.

Para crear esta alfombra, hemos utilizado dos cilindros con muy poca altura, pero un radio bastante grande. La parte inferior tiene el radio más grande que el superior para así crear una verdadera sensación de alfombra. Ambas partes han sido escaladas en z para generar un efecto de achatamiento por los lados.

Así mismo, les hemos aplicado una textura distinta a cada una de las partes para que se diferencien y le den un toque mayor en cuanto a realismo se refiere.

Todo esto compone un único objeto que es el que añadiremos a la escena.

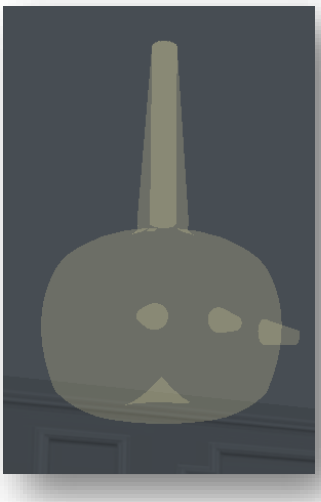
3.2.2. Bombilla.js.

Para realizar la bombilla hemos utilizado una técnica distinta a la utilizada en la alfombra, y es que, para este objeto, hemos creado una línea de puntos cerrada, a la cual luego le hemos aplicado la técnica de revolución en 2PI (360º); creándonos así una bombilla completa.

Además de la bombilla, disponemos de una cuerda que cuelga del techo, la cual la sujeta. Dicha cuerda está creada con una geometría de cilindro.

Junto a esta estructura principal, emergiendo por uno de los laterales, tenemos tres "focos". Cada uno de estos focos es un cilindro el cual tiene el radio superior más pequeño que el inferior, generando así una ilusión de dirección de por dónde sale la luz. Y es que, cada uno de estos focos, es el encargado de emitir una luz hacia su respectivo cuadro.

A todos estos objetos les hemos aplicado transparencia, simulando la transparencia de las bombillas reales. Todo esto compone un único objeto que es el que añadiremos a la escena.



3.2.3. calavera.js.

Alfabéticamente, debería de ir "caja.js" pero, como dicha clase incluye a esta, primero explicaremos este objeto.

Este objeto es un objeto importado de internet, y no se añadirá directamente a la escena, sino que, como se ha comentado anteriormente, se añadirá a la clase caja y es entonces cuando, al añadir ese objeto a la escena, aparecerá la calavera. El objeto se ha descargado desde la página https://free3d.com/

3.2.4. caja.js - nuestro modelo jerárquico.

Esta caja está formada por tres partes: la caja en sí, su tapa y la calavera explicada anteriormente encima de la tapa. Es nuestro modelo jerárquico.

Para realizar la caja en sí, hemos utilizado dos geometrías de cubos, uno más pequeño que otro; para así colocarlo dentro y hacer un csg.substract(). Y es que sí, este objeto está realizado con la técnica de CSG.

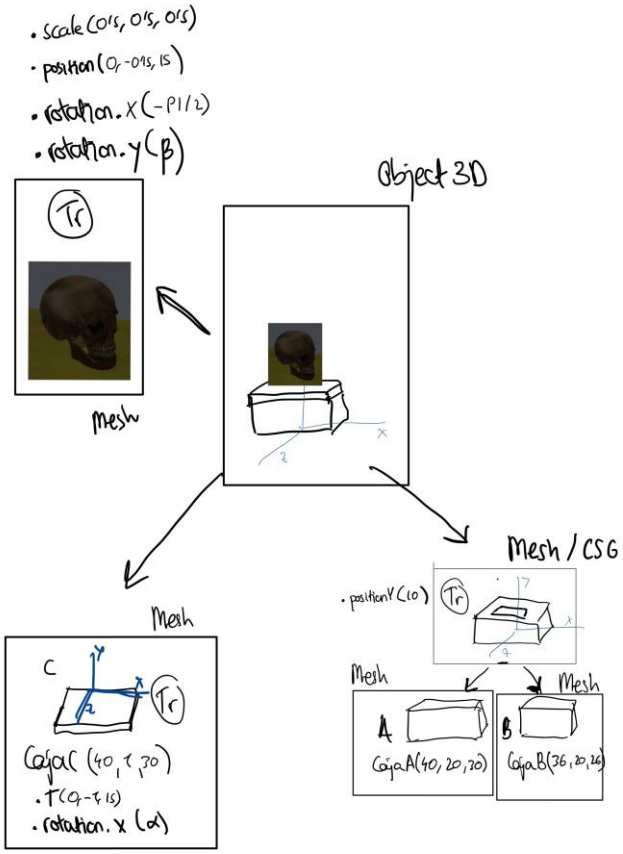
Por otra parte, hemos creado una tapa con una geometría de cubo y la hemos añadido al mismo objeto que la calavera; quedándonos así la "tapaEntera.

Una vez tenemos estos dos objetos, los añadimos a "cajaCompleta" para tener en un mismo objeto la caja, la tapa y la calavera. Este objeto será el que finalmente se añada a la escena.

Esta es la primera clase que nos encontramos que tenga animaciones o métodos auxiliares.

La animación es una animación que rota en x la tapa, abriéndola en un tiempo de 2 segundos.

Los métodos son el método update, el cual hace que la calavera esté constantemente rotando; y el método abrir, el cual será llamado una vez se haya resuelto el puzle requerido para poder abrir la caja.



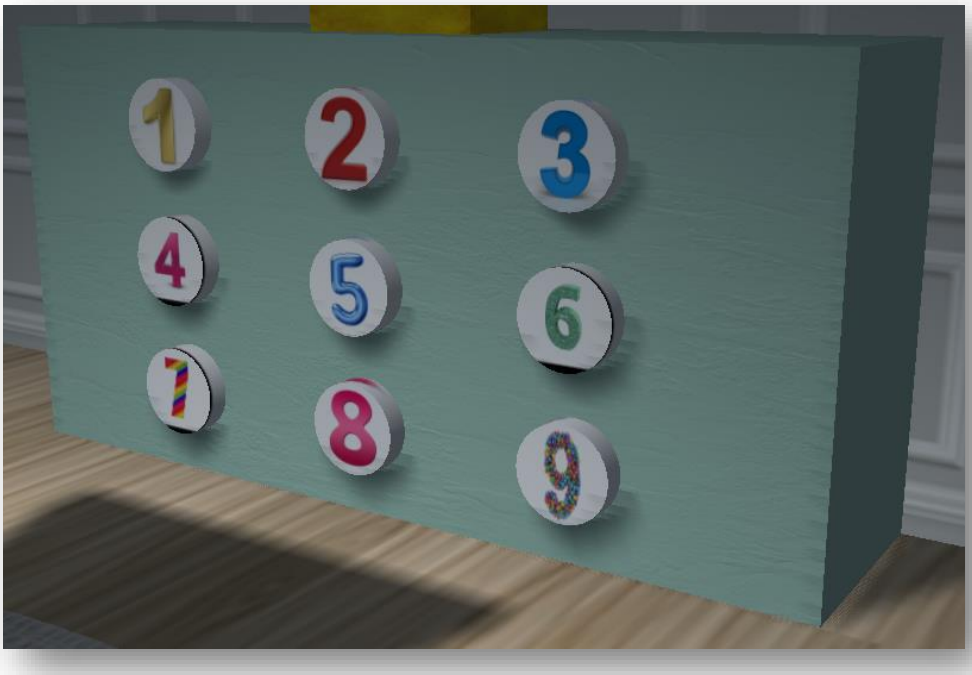
3.2.5. cajaFuerte.js.

CEn esta classe, todos los objetos (10 en total) han sido creados utilizando geometrías básicas. La caja grande es una geometría de un cubo y los botones son geometrías de cilindros.

Para el material de la caja, le aplicamos la combinación de un color (0xA4C9BB) con una textura en blanco y negro, la cual genera relieve; que es uno de los requisitos de la práctica, una **textura con relieve**. Así mismo, cada botón tiene una textura con el número específico al que representa, estas ya sin relieve.

Además, disponemos de 9 animaciones, una para cada botón; las cuales simulan que el botón está siendo pulsado. La animación de cada botón dura 0.25 segundos.

Junto a esto, tenemos 9 funciones que son llamadas cada cuando su respectivo botón ha sido pulsado para ejecutar su animación.



3.2.6. Cuadros.js.

Para crear los cuadros hemos creado tres geometrías de cubos a las que les hemos añadido dos texturas distintas, una de números para los cuadros a la izquierda y a la derecha, y otra textura del famoso cuadro "El grito" de Edvard Munch.

Los tres cuadros juntos son el objeto que finalmente se añade a la escena.

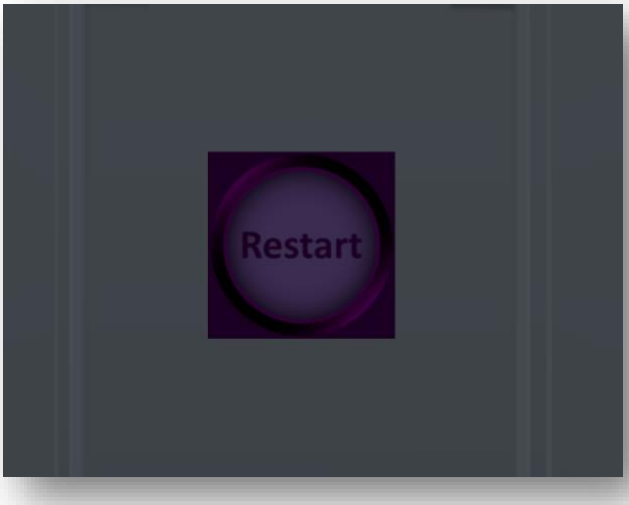
3.2.7. Estantería.js.

La estantería es de nuevo un objeto CSG. Para crearla, hemos tenido que crear primero una caja grande inicial (geometría de cubo), a la cual le hemos substraido 4 huecos (4 geometrías de cubo idénticas). Dejando así unos huecos entre leja y leja.

A toda esta estructura, le hemos añadido una textura de madera.

3.2.8. Fin.js.

Esta clase contiene una geometría de cubo simulando un botón el cual se le añade una textura mostrándonos el mensaje “Restart”. Se encuentra al final del pasillo que aparece al abrir la puerta. Si clicamos nos permitirá reiniciar una partida.



3.2.9. Interruptores.js.

En esta clase creamos los tres interruptores para poder interactuar con las luces de la escena. Cada interruptor está compuesto por dos geometrías de cubo que simulan su estructura y un material con un color básico. Cada interruptor tiene su propia geometría porque el pick funciona en base al mesh, entonces cada uno debía tener un mesh distinto.

Además, cada uno tiene su respectiva función para simular que está encendido o apagado.

3.2.10. libros.js.

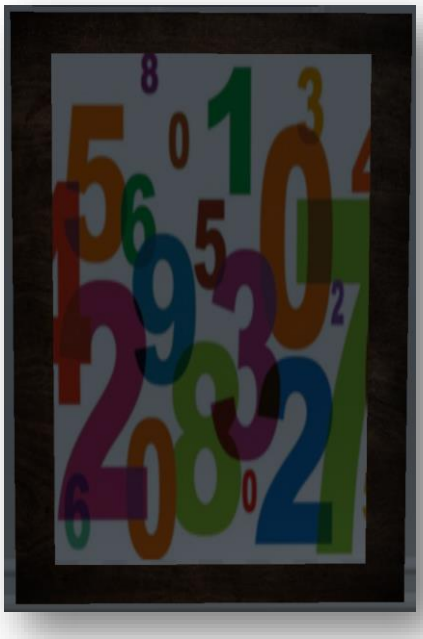
Para los libros hemos vuelto a utilizar la técnica CSG. Hemos creado la tapa con dos cubos (substract uno menor a otro más grande) y la unión con un cilindro. Las páginas del libro son una geometría de cubo. Cada uno de estos dos componentes, tapa y relleno, tiene su respectiva textura.

Una vez teníamos un libro creado, hemos ido clonándolo tantas veces como necesitásemos mientras íbamos colocándolo correctamente en la estantería.

El único que tiene un mesh distinto es el libro especial, para poder distinguirlo del resto para el pick.

3.2.11. Marco.js.

Para los marcos hemos creado un único marco y ya desde la escena creamos dos más y los colocamos correctamente. El marco es un CSG formado por una geometría de un cubo grande al que le quitamos un cubo interior más pequeño. Ese hueco formado es donde introducimos más tarde el cuadro, que tiene la medida exacta para encajar. El marco tiene su respectiva textura de madera.



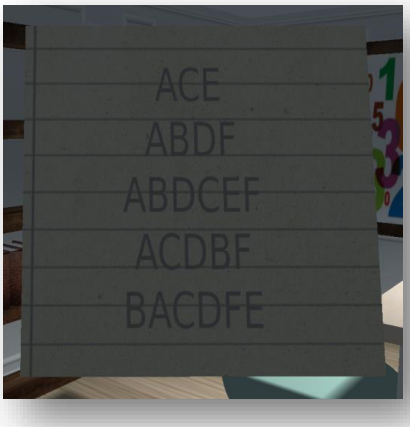
3.2.12. Mesa.js.

La mesa es una combinación de una geometría de cubo con cuatro patas (tres de ellas clonadas de la primera) con geometría de cilindro. La mesa tiene textura y las patas tienen un color básico.

Los 5 objetos anteriores forman un único objeto que es el que se añade finalmente a la escena.

3.2.13. nota.js.

Para crear “nota.js” hemos seguido el mismo patrón que para crear “Fin.js”. Disponemos de una geometría de cubo a la que le hemos añadido una textura creada por nosotros con las 5 combinaciones de letras. Dicha nota tiene dos animaciones, una para que venga justo al lugar al que estés, y otra para que vuelva exactamente a un punto determinado (al libro del que vino).



3.2.14. Ordenador.js.

Para crear todo el ordenador tenemos varios objetos separados los cuales combinamos finalmente en un solo objeto, el cual es el que se añade a la escena.

El primer objeto es la estructura de la pantalla, el cual es un CSG de un cubo grande al que le hemos quitado un cubo más pequeño interior. Su material es negro.

El segundo objeto es el pie que sujeta esta estructura, el cual es un objeto creado con puntos utilizando la técnica de revolución. Su material es negro.

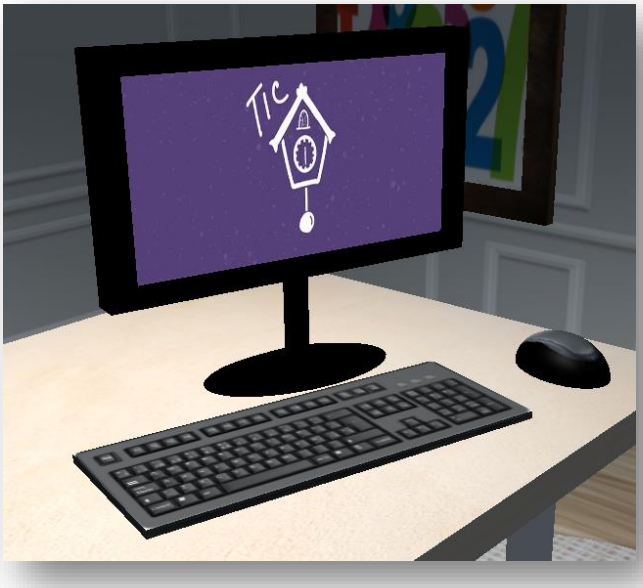
El tercer objeto es la pantalla que se introduce dentro de la estructura de la pantalla antes creada, es una geometría de cubo. Su textura es el punto clave del ordenador, pues cargamos un vídeo en ella. El primer vídeo es un TicTac estresante que aparece durante todo el juego; y la segunda textura aparece “The End” una vez nos hayamos pasado todos los puzles indicándonos justo eso, que hemos llegado al final del juego.

El cuarto objeto es el teclado, el cual es una geometría de cubo a la que le hemos aplicado una textura que simula un teclado.

El quinto objeto es el ratón, el cual es una esfera partida a la mitad a la que le hemos añadido una textura que simula un ratón.

Además de todo esto, disponemos de una función “cambiarTextura”, la cual es llamada al finalizar el juego para cambiar el vídeo que aparece en la pantalla.

En el update la variable needsUpdate se actualiza a true para poder visualizar correctamente el video.



3.2.15. Pared.js.

Aquí creamos las 4 paredes, así como el techo. Las paredes son de 500 de ancho por 250 de alto, y el texto es de 500x500. Todos los objetos son una geometría de cubo excepto la pared1, que es la pared en la que se ubica la puerta.

Dicha pared es un CSG al que se le ha restado un hueco con el tamaño exacto de la puerta, para colocarla después ahí.

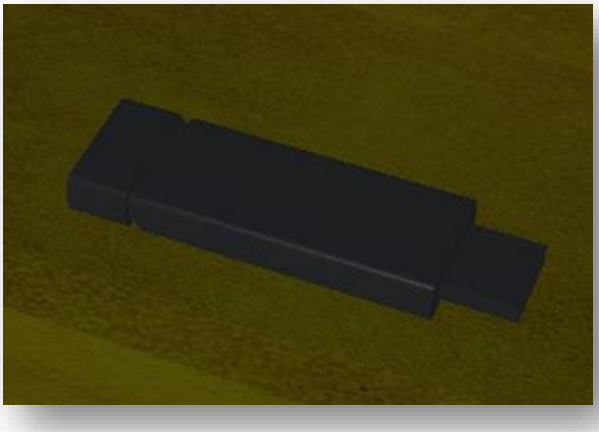
Las cuatro paredes tienen la misma textura y el techo tiene de material un color blanquecino.

Estos 5 objetos se colocan como un único objeto en la escena.

También contamos con las paredes que formarán el pasillo para terminar el juego, estas tienen el mismo material que las paredes del resto de la habitación.

3.2.16. pendrive.js.

Siguiendo la misma metodología que “calavera.js”, este es un objeto 3D importado. Sin embargo, la diferencia con el objeto calavera, más allá de claramente su apariencia, es que este objeto sí se añade directamente a la escena, y no pasa por ninguna clase intermedia como lo hacía la calavera.



3.2.17. Puerta.js.

La puerta está formada por la propia puerta, un pomo y un cilindro que une al pomo.

La puerta es una estructura CSG formada por una geometría de cubo a la que se le han restado dos geometrías de cubos más pequeñas en la parte superior e inferior simulando así la apariencia de una puerta tradicional. Su material es el color blanco.

Para el pomo se ha utilizado la geometría de una esfera y para la estructura que une el pomo y la puerta se ha utilizado una geometría de cilindro. Ambas tienen una textura dorada como la de la caja que contiene el pendrive.

Una vez tenemos estos objetos por separado, los unimos en un solo objeto llamado “puertaCompleta” que será el que finalmente añadamos a la escena.

3.2.18. silla.js.

La silla es el último objeto que compone nuestra escena. Este objeto está formado por la silla y la estructura que la sostiene.

La silla es una combinación de puntos a los que les hemos aplicado extrude, generándole así profundidad.

La pata que sostiene la silla es una geometría cilindro y el pie que sostiene toda esta estructura es otro cilindro con radio más grande, pero altura más pequeña.

Todo esto lo añadimos a un solo objeto, que será el que finalmente se añada a la escena.

3.3. Luces utilizadas.

A partir de este punto, trabajaremos solo con la clase MyScene.js, mencionando otras clases solo para decir cuáles reciben y proyectan sombras.

Dentro de MyScene.js, encontramos el método “createLights()”. Este método crea 5 luces para nuestra escena.

La luz ambiental, “ambientLight”, la cual hace que, en general, la escena no se vea completamente oscura, aunque no haya ningún tipo de fuente de luz directa.

La “spotLight”, es la luz principal que incide desde la bombilla hacia debajo. Gracias a un fragmento de código que tenemos en el método update, la luz está constantemente encendiéndose y apagándose, simulando un parpadeo en la sala.

La luz amarilla, la cual es otra spotLight, solo que esta apunta al cuadro central. Por defecto está apagada y se encenderá al activar el interruptor.

La luz azul, al igual que la luz amarilla, es otra spotLight que apunta al cuadro de la derecha. Esta también está apagada a la espera de ser encendida con el interruptor.

La luz roja, esta luz, aunque sea una spotLight como el resto, no hace exactamente lo mismo. Es cierto que ilumina a su respectivo cuadro, el izquierdo; y que por defecto está apagada hasta que sea encendida con el interruptor. Sin embargo, esta luz es especial, ya que en ella hemos introducido una textura. Dicha textura es el puzle de la segunda pista, la combinación de letras unidas con líneas.

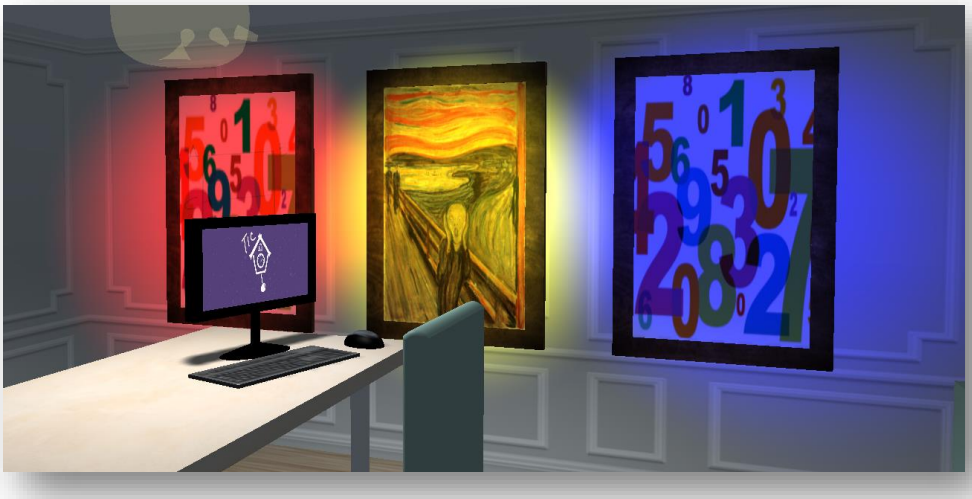
Las dos primeras luces proyectan sombras gracias al atributo “castShadow = true”.

Los objetos que reciben y, a su vez, proyectan sombras son:

La estructura del ordenador, la pantalla, el teclado, el ratón, la mesa, las patas de la mesa, las sillas, la alfombra, la caja grande, cada uno de los botones, la caja pequeña y la calavera.

No se ha hecho que todos los objetos de la escena generasen sombras porque esto reducía drásticamente el rendimiento del juego. Si se desease que todos los objetos generasen sombras, debería de activarse el atributo de cada mesh de la siguiente forma: “atributo.receiveShadow = true” para recibir sombras y “atributo.castShadow = true” para generar sombras.

Además, en el renderer debemos activar su atributo renderer.shadowMap.enabled a true y renderer.shadowMap.type a THREE.PCFSoftShadowMap para suavizar las sombras.



4. Interacción con la escena.

Una vez tenemos toda la escena físicamente creada, podemos interactuar con ella. Hay tres formas de hacerlo: moviéndonos por la sala, clicando en objetos de la sala y notas emergentes y/o permanentes de la sala. Vamos a explicar cómo hemos conseguido cada una de estas tres cosas en sus respectivos puntos.

4.1. Cámara y movimiento.

Para crear la cámara, creamos una nueva PerspectiveCamera, la cual se adapta a la resolución de la ventana. Tras esto, la colocamos en la posición (0, 160, 200). 160 va a ser la altura con la que vamos a jugar (aunque también podremos agacharnos como veremos más adelante).

Junto a esta, creamos un vector al que llamaremos look, que es hacia dónde mira la cámara.Gracias a un pointerLockControls al que le pasasmos por parámetros la camara y el look, podremos controlar la cámara.

Tendremos dos booleanos, uno para fijar o desfijar el control de la cámara con el ratón, y el otro para alternar entre agacharnos y estar de pie.

El control de los eventos para modificar la cámara los haremos a través de las teclas del teclado, y distinguiremos los eventos en función de si son “tecla presionada” o “tecla soltada”.

Para fijar la cámara: Deberemos de pulsar una única vez el “Shift izquierdo” o el “Shift derecho”. Alternaremos entre modo cámara bloqueada y desbloqueada pulsando alguna de estas dos teclas. Esto se consigue gracias a las funciones lock y unlock del objeto PointerLockControls.

Para agacharnos: Deberemos de pulsar una única vez el “Control izquierdo” o el “Control derecho”. Alternaremos entre modo agachado y de pie pulsando alguna de estas dos teclas. Esto se consigue gracias a que en el update actualiza la altura (Y) de la cámara.

Para avanzar: Deberemos de mantener pulsado “w” o “W” o “Flecha hacia arriba”. El movimiento se producirá mientras esté presionada la tecla, y se detendrá cuando la levantemos. Esto se consigue gracias a que cambiamos el valor de un booleano al pulsar la tecla a true, lo que se evalúa en el update, moviendo la cámara en dos valores hacia delante si no genera colisiones (las colisiones se explicarán en el siguiente apartado).

Para retroceder: Deberemos de mantener pulsado “s” o “S” o “Flecha hacia abajo”. El movimiento se producirá mientras esté presionada la tecla, y se detendrá cuando la levantemos.

Para movernos a la derecha: Deberemos de mantener pulsado “d” o “D” o “Flecha hacia la derecha”. El movimiento se producirá mientras esté presionada la tecla, y se detendrá cuando la levantemos.

Para movernos a la izquierda: Deberemos de mantener pulsado “a” o “A” o “Flecha hacia la izquierda”. El movimiento se producirá mientras esté presionada la tecla, y se detendrá cuando la levantemos.

4.2. Colisiones.

Una vez explicado el movimiento, debemos impedir que nuestro jugador atraviese todo tipo de objetos. Pues bien, esto se consigue gracias a generar y evaluar las llamadas colisiones.

Para ello, tenemos la función testColision(dondeEstoy), que realiza todo lo siguiente:

Primero, englobamos a nuestra cámara en una caja colisionable de 1x1x1. Si esta caja se interseca con alguna de las cajas colisionables, pondrá el bool “colisión” a true.

Le hemos añadido cajas colisionables a los siguientes objetos: Pared1, Pared2, Pared3, Pared4, Mesa, Silla1, Silla2, Estanteria, CajaFuerte, Marco1, Marco2, Marco3. Es decir, cualquier objeto con el que podríamos llegar a cruzarnos.

Para evaluar si nos colisionamos ejecutamos un bucle for que recorre todas las cajas colsionables que acabamos de mencionar ejecutando la función intersectsBox con la caja de nuestra cámara. Esta función devuelve true si se produjese alguna colisión.

Para el tema del movimiento, como hemos mencionado antes, en vez de movernos y evaluar la colisión, primero "desplazamos virtualmente la cámara" para comprobar si el movimiento que quiere realizar el usuario produciría colisión. Si no produce colisión, se le permite realizarlo; en caso, contrario, no.

4.3. Picking.

Una parte fundamental de nuestro juego es poder interactuar con los objetos de la escena haciendo clic con el ratón. Pues bien, esto lo conseguimos gracias al picking. Para introducir esta técnica al código, hemos creado la función "onDocumentMouseDown(event)".

Esta función recoge las coordenadas x e y del ratón a la hora de clicar, lo que proyecta un rayo que interseca todos los mesh que hay en su camino, interactuándo con el primero de ellos.

Para definir los objetos a los que podemos clicar, los añadimos al vector "pickableObjects". Estos objetos son: libros, ordendor, cajaFuerte, caja, interruptores, puerta, pendrive, nota y fin.

Una vez hagamos clic, los objetos que interseque se introducirán al vector pickedObjects. Si dicho vector es mayor que 0, se entrará en un switch-case, que realizará una función u otra en base del objeto que haya clicado.

Aquí es donde entran en juego todas las funciones de las clases anteriormente descritas, las cuales son invocadas desde aquí. Algunos ejemplos son activar los interruptores, llamar a las animaciones de las notas para que vengan a la cámara o la pulsación de los botones.

Cabe resaltar que el hecho de clicar el pomo, si no se ha completado el juego, no hace nada. Esto se consigue evaluando dentro de su case si el bool "ganar" se encuentra a true.

4.4. Mensajes permanentes.

Hay un último elemento que interactúa con el jugador, sin embargo, a diferencia de los otros, el jugador no podrá modificarlo o interactuar con ello a su antojo.

Se trata de las notas permanentes, unas notas que aparecen en la esquina superior derecha y cambian en función del momento del juego en el que nos encontremos. Nos sirven para orientarnos sobre qué está pasando y para almacenar información como las combinaciones de las letras obtenidas por el libro y no obligar al jugador a memorizarlas o apuntarlas en algún lugar externo al juego.

Estas notas se consiguen gracias a que desde esta clase podemos actualizar un apartado en el html, concretamente el que tiene id="Mensajes".

Con la función setMensagge(str) podemos ir modificando el mensaje que aparecerá en ese apartado. Llamamos a esta función cada vez que se supera un puzle o sucede algo extraño en el juego.

5. MyScene.js.

Aunque ya se han introducido muchas de las funciones de esta clase, en este apartado se va a comentar cuál es su función general en el juego; así como funciones básicas que no se han comentado aún o funciones introducidas por nosotros para llevar a cabo algunos de los puzles.

Esta clase es la clase principal del juego, en ella se instancian todos los objetos previamente creados, todas las luces, las funcionalidades, el renderer, el update, etc. En general, es la clase base para el ecosistema de nuestro juego.

5.1. Componentes básicos no descritos anteriormente.

El constructor. En el constructor inicializamos todos los objetos o, en su lugar, llamamos a las funciones que inicializan objetos como el renderer, la cámara, las luces o el suelo. Además, tenemos booleanos generales como en qué punto del juego estamos. Junto a esto, tenemos la cadena de la caja fuerte y los respectivos booleanos para poder ir evaluando si se va introduciendo correctamente.

Tras esto, inicializamos los objetos de las clases descritas anteriormente, y las añadimos a la escena. Definimos los objetos pickables y colocamos en la escena el primer mensaje permanente, “Has iniciado una partida. ¡Consigue escapar antes de perder la cordura!"

createGround(). Crea la geometría del suelo y le activa el atributo para poder recibir sombras. Se desplaza en y –(la mitad de su altura) para colorse justo debajo del origen. También se crea el suelo del pasillo.

createRenderer(). Crea el renderer, el cual es el encargaddo de renderizar la imagen que se muestra por pantalla.

getCamera(). Devuelve el objeto de la cámara.

setCameraAspect(ratio). Cada vez que el usuario modifica el tamaño de la ventana desde el gestor de ventanas de su sistema operativo se actualiza el ratio de aspecto de la cámara, y la matriz de proyección de la cámara.

onWindowResize(). Cada vez que el usuario modifica el tamaño de la ventana de la aplicación se actualiza el ratio de aspecto de la cámara y también el tamaño del renderer.

update(). Constantemente se actualiza el renderer, los vectores dondeEstoy y dondeMiro. Junto a esto, como se ha comentado anteriormente, las evaluaciones de avanzar, retroceder, moverse hacia la derecha y moverse hacia la izquierda.

Comprueba si el bool de agacharse está true o false para alternar entre y=160 e y=110.

Si el bool abirCaja está a true (debido a que se haya metido correctamente la combinación en la caja fuerte), se abre la caja y se cambia el mensaje permanente de la esquina superior derecha.

Ejecuta el update() de la clase calavera para que esté constantemente girando. Así como el del ordenador, para que el vídeo se esté ejecutando correctamente durante todo el tiempo.

Modifica la intensidad de la luz para que parpadee como se explicó en el apartado de luces.

requestAnimationFrame(() => this.update()). Este método debe ser llamado cada vez que queremos visualizar la escena de nuevo, ya que si no se llamara a este método update() solo se ejecutaría la primera vez.

$(function (){}. La función main, la cual contiene:

var scene = new MyScene("#WebGL-output"): Se instancia la escena pasándole el div que se ha creado en el html para visualizar.

window.addEventListener ("resize", () => scene.onWindowResize()): Listener que comprueba cuándo se modifica el tamaño de la ventana de la aplicaión.

window.addEventListener("click", (event) => scene.onDocumentMouseDown(event)): Listener que comprueba cuándo se ha hecho clic en la ventana de la apliación.

scene.update(): La primera visualización.

5.2. Funciones auxiliares utilizadas.

En este apartado se describen las funciones que hemos creado nosotros para la correcta ejecución de nuestro juego y los puzles. Estas funciones son: setMessage(str), testColision(dondeEstoy), evaluarCadena(cadena) y onDocumentMouseDown(event). Sin embargo, tres de estos cuatro ya han sido explicados previamente en apartados anteriores, por lo que en este apartado solo procederemos a explicar la función evaluarCadena(cadena).

Esta función obtiene una cadena como argumento (la cadena que se le pasa es la que había previamente a pulsar uno de los botones más el número del botón que se ha pulsado), es decir, si cadena era [2, 3] y se pulsa el botón 8, se pasará la cadena [2, 3, 8].

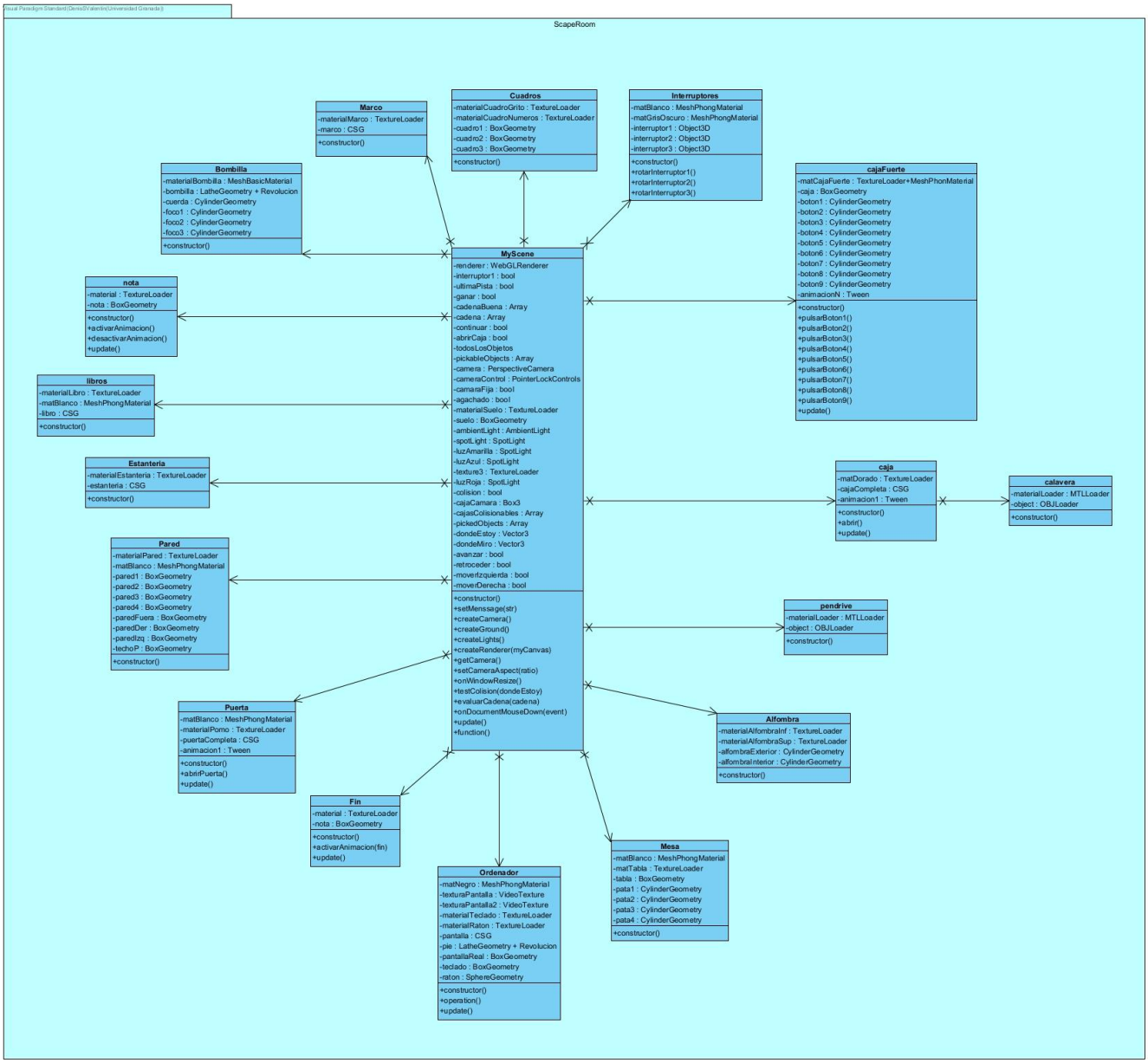
Como si hemos llegado hasta el punto de enviar 3 números significa que los dos anteriores ya han sido evaluados y son correctos, es decir, que coinciden de momento con la cadena correcta; se evalúa solo el último elemento introducido.

Esto se consigue obteniendo la longitud de la cadena pasada-1 y evaluando ese valor con el valor que se encuentra en esa misma posición en la cadena correcta.

Si el valor coincide, el bool continuar se mantiene a true y permite seguir introduciendo números en esa cadena, en caso contrario, la cadena se reiniciaría dejandola completamente vacía, teniendo que rellenarse de nuevo.

Si el valor que se va a evaluar hace que la cadena sea de la misma longitud que la cadena buena, se accede a otra evaluación, la cual, en caso de salir correcta, pone abrirCaja a true, lo que abre la caja que contiene el pendrive.

6. Diagrama de clases.



7. Referencias a modelos 3D y sus respectivas librerías.

Para buscar modelos 3D en internet hemos utilizado la página: https://free3d.com/es/

De esta página hemos importado los dos modelos 3D que tenemos, los cuales son calavera y pendrive.

Para poder utilizar dichos modelos en nuestro juego hemos tenido que utilizar las librerías MTLLoader y OBJLoader.

La librería MTLLoader en Three.js permite cargar y aplicar los materiales del archivo MTL a los objetos 3D en la escena. El archivo MTL contiene información sobre los diferentes materiales que se utilizan en el modelo 3D, como el color, la textura, el brillo, la opacidad, entre otros.

La librería OBJLoader analiza el archivo OBJ y crea un objeto Three.js que representa el modelo, con su geometría y otras propiedades asociadas.

8. Vídeo del juego.

Hemos realizado un vídeo mostrando los aspectos más importantes del juego: https://drive.google.com/file/d/1KNzuDkeSo4QgbgYyf5wWHk-zb7Xv2lmS/view?usp=sharing