Diseño Scape-Room

Introducción

A continuación, describiremos el diseño de nuestra aplicación incluyendo el diagrama de clases de la misma, el modelo jerárquico del objeto articulado con movimiento continuo, que en nuestro caso es el flexo de la mesa.

Por último, describiremos los algoritmos más importantes que hemos utilizado durante el desarrollo de la habitación, que serán el picking y el sistema de colisiones.

La funcionalidad y jugabilidad del juego quedará descrita en el documento “Manual Usuario”, donde detallaremos los pasos a seguir para lograr salir de la habitación.

Modelo Jerárquico

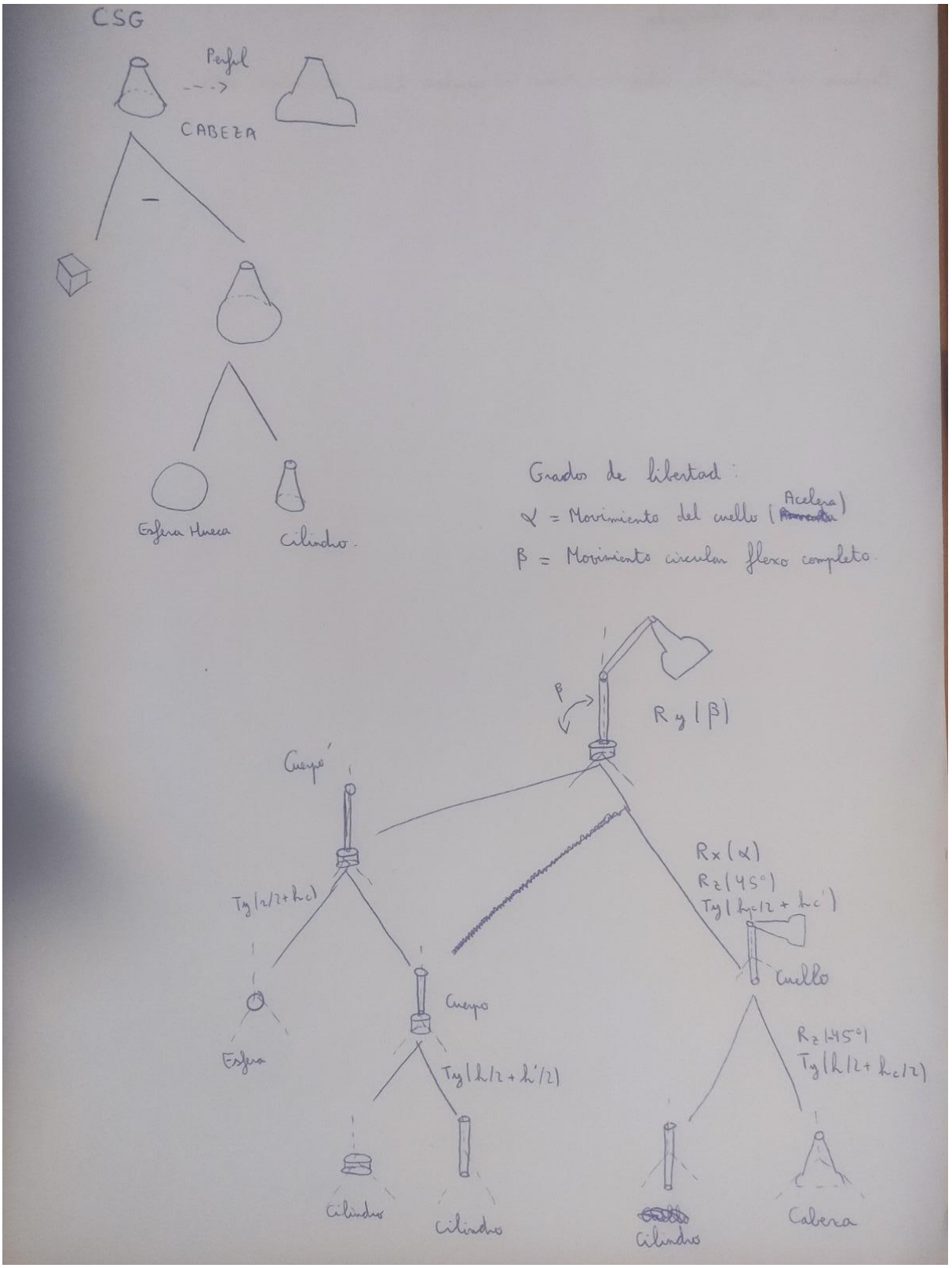
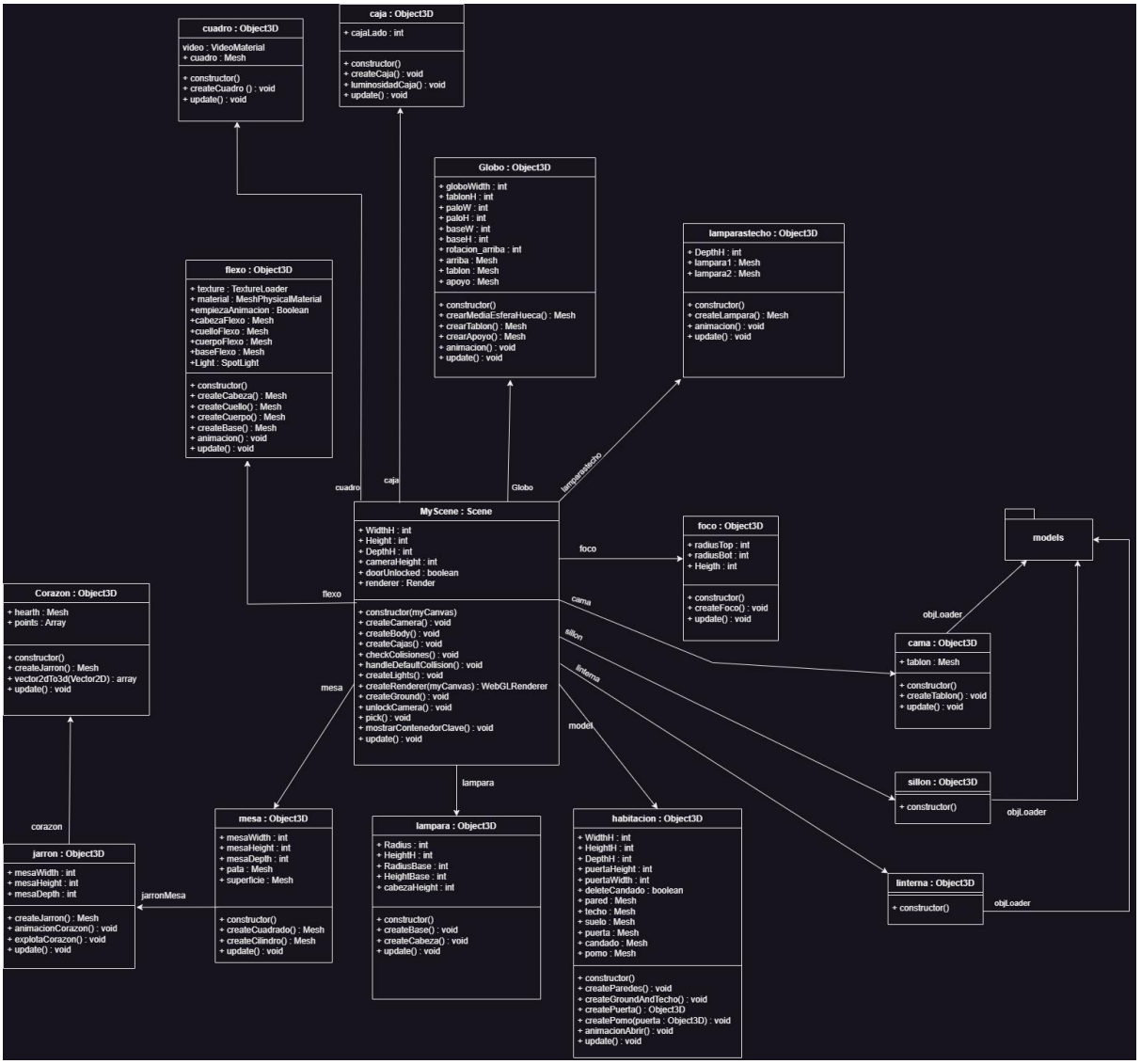


Diagrama de clases



Para una mejor visualización, mirar el archivo “diagrama_clases.jpg”

Explicación Algoritmos

Picking

En la aplicación, se utiliza un algoritmo de selección de objetos para permitir a los usuarios interactuar con elementos específicos en la pantalla. Dado que la cámara utilizada en la aplicación no muestra el cursor del ratón, se lanza un rayo desde la mitad de la pantalla para simular un punto de mira. Este punto de mira se utiliza como referencia para determinar qué objetos se encuentran dentro del área seleccionada.

El algoritmo utiliza un array de elementos llamado "pickeableObjects" que contiene todos los objetos que se pueden seleccionar en la aplicación. Cuando se realiza un clic en cualquier parte de la pantalla, el algoritmo verifica si el puntero se encuentra sobre alguno de los objetos utilizando la función "ray.intersectObjects(this.pickeableObjects, true)". Esta función realiza una intersección entre el rayo lanzado desde el punto de mira y los objetos en el array "pickeableObjects". Si se detecta una intersección con al menos uno de los objetos, se selecciona el objeto más cercano y se realiza un tratamiento adecuado.

Una vez que se ha seleccionado un objeto, el algoritmo comprueba el nombre del objeto seleccionado para determinar qué acción se debe realizar. Dependiendo del nombre del objeto, se pueden realizar diversas acciones, como encender una luz, activar una animación u otras acciones específicas para cada objeto. Esta lógica basada en el nombre del objeto seleccionado permite personalizar las interacciones.

Colisiones

Para el método de las colisiones hemos utilizado la técnica de cajas englobantes es decir, para cada objeto con el que queríamos colisionar creamos una caja que engloba a todo el objeto y la añadimos a un array de candidatos (*candidates* en la clase *MyScene)*.

Usamos esta técnica y no la de *ray tracing* ya que si la cámara está más alta que algún objeto no colisionará, entonces adicionalmente hemos añadido un “cuerpo” a la camara, le hemos añadido una caja englobante que llegue hasta el suelo.

Lo que hacemos es comprobar si cada candidato colisiona con el cuerpo añadido a la cámara y en caso de hacerlo manejamos la colisión.

Si el cuerpo colisiona contra algún objeto lo que hacemos es mover al personaje en la dirección contraría, para esto calculamos la diferencia del vector de la posición anterior y la actual, a la que ha avanzado con esto evitamos que independientemente de si colisiona de frente, hacia atrás o de lado no atraviese el objeto.

chat

Para implementar el método de colisiones en nuestro proyecto, hemos utilizado la técnica de cajas englobantes. Esto implica que, para cada objeto con el que queremos detectar colisiones, creamos una caja que englobe todo el objeto y la añadimos a un array de candidatos denominado "candidates" en nuestra clase MyScene.

Elegimos esta técnica en lugar del ray tracing debido a una limitación específica: si la cámara se encuentra en una posición elevada con respecto a algún objeto, es posible que no se detecte la colisión correctamente. Para solucionar este problema, hemos agregado un "cuerpo" a la cámara al cual le hemos añadido una caja englobante que se extiende hasta el suelo.

El proceso de detección de colisiones se lleva a cabo comprobando si cada candidato colisiona con el cuerpo agregado a la cámara. En caso de que ocurra una colisión, gestionamos el evento correspondiente.

Cuando el cuerpo de la cámara colisiona con algún objeto, aplicamos un movimiento al personaje en dirección opuesta a la colisión. Para lograr esto, calculamos la diferencia entre el vector de posición anterior y el vector de posición actual, que representa el avance del personaje. Al hacer esto, evitamos que, independientemente de si la colisión ocurre de frente, desde atrás o de lado, el personaje atraviese el objeto.

Este enfoque nos permite simular las colisiones de manera efectiva y realista, asegurando que el personaje y la cámara interactúen adecuadamente con los objetos del entorno. Al utilizar cajas englobantes y el "cuerpo" de la cámara, hemos mejorado la precisión de la detección de colisiones y hemos evitado problemas comunes asociados con la ubicación relativa de la cámara.

Bibliografía

Los modelos .obj (linterna, cama y sofá) han sido sacados de:

- Linterna: https://free3d.com/es/modelo-3d/simple-flashlight-70024.html
- Cama: https://free3d.com/es/modelo-3d/old-metal-bed-96580.html
- Sillón: https://free3d.com/es/modelo-3d/armchair-haifa-465303.html

Hemos utilizado la clase PointerLockControls para la cámara, poniendo los movimiento a mano. La clase ha sido sacada del sitio oficial:
https://threejs.org/docs/#examples/en/controls/PointerLockControls
o bien de Prado (proporcionada por el profesor).