

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN DISEÑO Y DESARROLLO DE VIDEOJUEGOS

Curso Académico 2018/2019

Trabajo Fin de Grado

INTERACCION GESTUAL PARA EL APRENDIZAJE DE POO CON LEAP-MOTION

**Autor**: Pablo Rodríguez Vicente

**Director**: Maximiliano Paredes

# Índice

[Índice 3](#_Toc9252421)

[Resumen 5](#_Toc9252422)

[Introducción 7](#_Toc9252423)

[Objetivos 7](#_Toc9252424)

[Fundamentos 9](#_Toc9252425)

[Tecnologías 9](#_Toc9252426)

[Unity 9](#_Toc9252427)

[LeapMotion 10](#_Toc9252428)

[C# 11](#_Toc9252429)

[Blender 12](#_Toc9252430)

[Visual Studio Code 12](#_Toc9252431)

[GitHub 13](#_Toc9252432)

[Descripción de la Aplicación 15](#_Toc9252433)

[Metodología de trabajo 15](#_Toc9252434)

[Desarrollo 15](#_Toc9252435)

[Inicios del Proyecto 16](#_Toc9252436)

[Manager 17](#_Toc9252437)

[Sistema de menús 18](#_Toc9252438)

[Interacción 21](#_Toc9252439)

[Transiciones 22](#_Toc9252440)

[Problemas en el Desarrollo de los Scripts 22](#_Toc9252441)

[Diseño de Interfaz 23](#_Toc9252442)

[Optimizacion de Renderizado 24](#_Toc9252443)

[Accesibilidad 24](#_Toc9252444)

[Multilenguaje 24](#_Toc9252445)

[Modelado 25](#_Toc9252446)

[Problemas en el Desarrollo de la Interfaz 26](#_Toc9252447)

[Conclusiones 29](#_Toc9252448)

[Bibliografia 31](#_Toc9252449)

# Resumen

En este trabajo se expondrá el desarrollo de la aplicación para el aprendizaje de Programación Orientada a Objetos que se ha realizado haciendo uso de tecnologías gestuales, en este caso Leap Motion.

La aplicación se desarrolla con el objetivo de servir de herramienta de apoyo para la explicación de los conceptos de POO, se busca de esta manera hacer que este concepto quede más claro mediante una interacción más directa y visual con los conceptos que se explican en las aulas.

La aplicación está pensada para los cursos iníciales sobre POO no profundizando en exceso en estos. La interfaz se ha creado pensando en el material que posen las aulas, por tanto además del uso de Leap Motion se da la opción al usuario de usar el ratón

# Introducción

## Objetivos

# Fundamentos

## Tecnologías

### Unity

Unity es un motor de renderizado de gráficos en tiempo real que aparece en 2005, creado por la empresa Unity Tecnologies[2], su uso se ha extendido mucho en los últimos años. Aunque su principal enfoque eran los videojuegos ha demostrado que sirve para la creación de otro tipo de proyectos como por ejemplo animaciones[3]. Unity ofrece el motor de manera gratuita para todas aquellas aplicaciones que no superen uno beneficios de 100000€ anuales con la aplicación realizada, si se superan se debe comprar una licencia que cuestan 125€/mes, esto lo convierte en un motor ideal para realizar proyectos académicos.

En las versiones más antiguas de Unity no estaba disponible toda la funcionalidad de manera gratuita pero esto cambio en 2016 con la versión 5.6 del motor convirtiéndolo así en un motor gratuito muy potente. Unity se complementa muy bien con otras aplicaciones como por ejemplo Blender, esto ha facilitado el desarrollo de toda la interfaz 3D. También cabe destacar que Unity posee Una serie de paquetes que permiten añadir funcionalidades extra como por ejemplo textos Textos 3D para la interfaz o postprocesos para la iluminación.

Una de las principales ventajas de Unity es que permite crear proyectos para más de 25 plataformas y el motor puede ser usado tanto en Windows como en Linux y Mac, esto lo vuelve un motor muy versátil a la hora de trabajar con él. Además de varias plataformas Unity también es compatible con varios motores gráficos como OpenGL, Direct3D e interfaces propietarias como por ejemplo la Wii. El lenguaje de programación de Shaders es ShaderLab.

Unity está basado en scripting que viene dado mediante Mono, una implementación de código abierto de .NET Framework [4]. El lenguaje ofrecido para la realización de los scripts es C#. Unity tiene una estructura de motor de juego, todos los objetos que se encuentran en la escena se llaman GameObjects, a estos GameObjects se les añade funcionalidad mediante componentes.

Se ha elegido Unity frente a su principal competidor Unreal porque este último no ofrece funcionalidad como la que ofrece Unity para la creación de aplicaciones y se centra más en el desarrollo de videojuegos. La programación en Unreal se realiza principalmente mediante su sistema de Blueprints, que es un editor por nodos para la creación de comportamientos en Unreal.

La versión de Unity usada para el proyecto es la 2018.3, esta versión incluye cambios en el sistema de prefabs [5] que incluye un nuevo flujo de trabajo y la posibilidad de anidar prefabs facilitando así la creación de elementos duplicados, esta característica ha sido de mucha importancia para la interfaz permitiendo crear los botones de manera sencilla y pudiendo modificarlos todos a la vez.

Una funcionalidad muy destacable de Unity que lo hace muy útil a la hora de desarrollar este tipo de proyectos 3D es sus sistema de escenas, este sistema nos permite hacer pruebas sin tener que cambiar cosas en las escenas principales volviéndolo muy útil para probar funcionalidades antes de implementarlas.

### LeapMotion

LeapMotion es un sensor de seguimiento de manos desarrollado por la empresa con el mismo nombre. Este dispositivo es la base de este trabajo por el potencial que tiene para el aprendizaje.

Es un dispositivo de pequeño tamaño que se conecta mediante USB, para realizar el seguimiento hace uso de 2 cámaras infrarrojas monocromáticas y 3 leds infrarrojos y observa un área parecida a una semiesfera con alrededor de 1m de rango.

LeapMotion está pensado principalmente para la RV por las interacciones naturales que permite con los objetos virtuales frente al uso de controladores tradicionales, además en su web indican que los mejores usos para el Leap Motion son la rehabilitación y la educación.

Además del hardware la empresa LeapMotion provee a los desarrolladores con una API escrita en C, cuyo uso no se recomienda directamente, y SDK para los principales motores de juegos, Unreal y Unity.

Además del SDK que permite el uso, sin problemas para el desarrollador, dentro del motor LeapMotion también provee un modulo de interacción para la creación de interfaces 3D propias de la Realidad Virtual. Este motor hace uso del motor de físicas para la detección de colisiones entre las manos y los objetos interactivos.

Se ha elegido esta tecnología para el trabajo por el potencial que posee para el desarrollo de aplicaciones de aprendizaje y por los retos que plantea al ser una tecnología con la que no se ha trabajado antes.

La conexión entre Unity y LeapMotion es muy sencilla basta con bajarse el el paquete de Unity de LeapMotion de su página web, esto es un archivo que contiene assets y puede ser descomprimido e importado por Unity. A partir de ese momento basta con crear un objeto en la escena y añadirle los Scripts encargados de trackear las manos, a partir de ese momento si el LeapMotion esta conectado al ordenador al ejecutar la aplicación esta mostrara unas manos

(foto de manos en Unity).

Ademas de esta funcionalidad básica LeapMotion para Unity ofrece un conjunto de Scripts para posiciones de la mano llamados detectores que lanzaran un evento al producirse.

(fotos de detectores)

Tambien existe un paquete que ofrece interacciones con los objetos de las escena simplemente añadiéndoles el script del comportamiento que queremos

(fotos de componentes de paquetes de la pagina web)

Otra característica adicional que se obtiene a través de otro paquete es el poder crear manos personalizadas mediante una herramienta de autorig que nos permitirá usar modelos propios si es necesario.

(foto de manos rigeadas)

### C#

C# es un lenguaje de alto nivel orientado a objetos creado por Microsoft para su plataforma .NET. Es el lenguaje de scripting usado en Unity y posee muchas similitudes con Java pero es más cercano a C++ es su diseño. Con el tiempo se ha ido diferenciando aun más de Java.

Fue creado por [Anders Hejlsberg](https://es.wikipedia.org/wiki/Anders_Hejlsberg) en el año 2000 como un nuevo lenguaje de programación orientada a objetos, se basa en tipos y métodos, al principio se iba a llamar Cool(C-like Object Oriented Language), pero por razones de registro tuvieron que cambiarlo.

En su versión 3.0 C# incluye la extensión Linq, esta librería usa sintaxis muy parecida a la de SQL para extraer elementos de bases de datos, también se puede usar para buscar en colecciones. También incluye expresiones lambdas muy útiles a la hora de trabajar con eventos en Unity.

### Blender

Blender es una herramienta de código abierto para la creación de contenido 3D desarrollada por Blender Foundation. Ofrece pues, toda la funcionalidad de un programa de modelado como puede ser 3dsMax pero de manera gratuita. Al igual que Unity es multiplataforma lo que lo convierte en una herramienta ideal para trabajar en paralelo a Unity.

Blender aparece en 1998 por parte de la compañía NaN, creada por Ton Roosendaal, que buscaba crear una herramienta profesional de contenido 3D pero de manera gratuita para el usuario frente a las aplicaciones del momento que costaban miles de dólares. La aplicación fue un existo, pero económicamente no aportaba lo suficiente a los inversores que decidieron cerrar NaN, a pesar de esto la comunidad y Ton no dejaron que Blender cayera en el olvido, para ello Ton creo la organización no lucrativa Blender Foundation que se encargar junto a los usuarios de promocionar y desarrollar Blender.

Actualmente Blender ofrece a parte de herramientas de desarrollo 3D infinidad de funcionalidades creadas por los usuarios. En la actualidad Blender se ha convertido en una opción muy importante a la hora de considerar que software de modelado usar en un proyecto por su potencia y sobre todo por su coste.

Se ha elegido para este proyecto la versión 2.8 de blender, esta versión se encuentra en beta pero ofrece muchas mejoras en la interfaz que se han visto necesarias para agilizar el proyecto en su fase de modelado .

### Visual Studio Code

Visual Studio Code es un editor de código multiplataforma de código abierto desarrollado por Microsoft, aparece en 2015 bajo una licencia MIT. Está basado en Electron, un framework para la creación de aplicaciones que utiliza Javascript como lenguaje. Es compatible con multitud de lenguaje gracias a una de sus principales características, las extensiones. Estas extensiones son desarrolladas por la comunidad y permiten mejorar el editor para que se adapte a las necesidades de cada momento.

En las últimas versiones de Unity se ha incluido compatibilidad total con Visual Studio Code mediante un par de extensiones, una para la detección de clases de Unity y otra para debuguear el código a través de Visual Studio Code.

(foto de entorno y tienda de extensiones)

### GitHub

GitHub es una plataforma para alojar proyectos software que hace uso del sistema de control de versiones Git. GitHub aparece el 8 de Febrero de 2008 por parte de la compañía GitHub Inc y el 4 de junio de 2018 es comprado por Microsoft.

Las herramientas de control de versiones son muy importantes en proyectos software, para tener siempre una copia del código en distintas versiones del proyecto para poder enmendar errores que aparezcan durante el desarrollo al añadir nuevas funcionalidades. En este caso se ha elegido GitHub por la sencillez para crear un repositorio de manera gratuita, se ha creado un repositorio privado durante el desarrollo que ha pasado a ser público al acabar este.

Para el uso del repositorio se ha descargado la herramienta GitExtansion, esta herramienta ofrece una interfaz intuitiva para trabajar con el repositorio de todas las maneras necesarias.

(foto de git extension)

# Descripción de la Aplicación

## Metodología de trabajo

El desarrollo de la aplicación ha sido dividido en 2 grandes partes, por un lado el desarrollo de la funcionalidad, es decir, los scripts necesarios para que todo el sistema de botones, menus e interacciones funcionara. Por el otro lado está el diseño de la interfaz que se ha centrado principalmente en usar una paleta para daltónicos, reducir el uso de textos en botones al mínimo y organizarse de manera que sea fácil de usar con la interacción gestual.

La metodología seguida en este proyecto ha consistido en la repetición de varias 3 fases. La primera es la prueba de la funcionalidad que se desea implementar en una escena a parte, explorando cual es la mejor manera para implementarla con todas las posibilidades que nos ofrecen tanto Unity como LeapMotion.

La segunda fase consiste en la implementación de la funcionalidad junto con todo el resto de funcionalidad corrigiendo los posibles errores que surjan.

La última fase consiste en la validación de las funcionalidades añadidas, para esto se han realizado reuniones con el profesor para que diera el visto bueno o indicase que cambios habría que realizar.

Este proceso se ha seguido de manera iterativa aumentando en cada iteración las funcionalidades de la aplicación y mejorando las funcionalidades ya existentes.

Esta metodología se ha apoyado sobre un repositorio, herramienta muy útil para este tipo de desarrollo permitiéndonos llevar un control sobre los cambios de cada iteración.

## Desarrollo

El desarrollo se divide en varios Scripts, algunos de ellos creados desde 0 usando la clase base de los GameObjects, llamada Monobehaviour, y otros heredando de los Scripts de LeapMotion, también se han creado clases auxiliares para la persistencia que se ha desarrollado mediante un archivo binario que guarda la información.

El desarrollo se ha dividido a su vez en tres partes, una parte ha consistido en la creación de los menús de creación de clases. Estos menús eran necesarios ya que el usuario no puede escribir código como haría en una aplicación de aprendizaje al uso. Por lo tanto se ha decidido dotar a la aplicación con un creador de clases que da al usuario las opciones necesarias para la creación de clases, que cumplan con las explicaciones del concepto de Programación Orientada a Objetos. El usuario puede crear atributos de 3 tipos (int, boolean, float) y 6 métodos que comprenden las operaciones básicas y operaciones de entrada y salida.

La segunda parte consiste en todo el sistema de creación y almacenamiento de representación de objetos, variables, atributos y métodos en la escena, esta parte se ha solventado usando mallas simples con distintos colores que las identifiquen, permitiendo interactuar con todos ellos haciendo uso de los scripts de interacción de LeapMotion o creando clases que hereden de ellos, y creando un menú que centraliza las clases donde se dan varias opciones sobre ellas que van desde inspeccionar el código java que se corresponde a la clase hasta modificar clases ya creadas.

La tercera y última parte consiste en la inspección de objetos y variables para ver su contenido y la posibilidad de poder ejecutar los métodos que contengan los objetos, esta parte guarda relación con la segunda pero tiene funcionalidades totalmente distintas que requerían la creación de scripts nuevos.

Por último ha habido una parte que ha afectado a todas las partes del desarrollo y del diseño de la interfaz y es la interacción gestual basada en LeapMotion, Ha afectado a todo el desarrollo pues el enfoque es muy distinto de una aplicación tradicional como puede ser BlueJ. Se ha enfocado la aplicación a una pseudo realidad virtual en la que toda la interfaz funciona por físicas y es tridimensional pero la cámara esta fija en una zona como una aplicación tradicional, esto ha generado una cantidad muy grande comportamientos no deseados en los inicios del desarrollo que han sido solventados a medida que progresaba el proyecto.

Cabe destacar que la aplicación se ha basado en su mayoría en eventos producidos por interacciones, intentando asi reducir el consumo por la ejecución de scripts, el único uso que se le ha dado a una función ejecutada periódicamente, en este caso el Update de Unity, ha sido para el movimiento de los objetos con el raton, esta posibilidad se explicara mas adelante en el apartado de interfaz.

## Inicios del Proyecto

En los inicios del desarrollo se valoro la posibilidad de permitir al usuario introducir sus propias clases Java externas a la aplicación, esta característica se abandono tras un tiempo de investigación en el que se encontraron excesivas dificultadas a la hora de juntar esta funcionalidad con la funcionalidad de ejecutar métodos. La principal dificultad pasaba por la imposibilidad de ejecutar el código java desde Unity.

Otra dificultad que se encontraron con este planteamiento fue las dificultades por las que pasaría el usuario a la hora de añadir sus clases a la aplicación para que esta las leyera. Finalmente se decidió optar por dotar a la aplicación con su propio creador de clases que aporta las opciones necesarias para las explicaciones de los conceptos que se tratan.

El resto de objetivos no cambiaron a lo largo del desarrollo y fueron desarrollados son problemas mayores.

### Manager

Este Script merece un punto aparte pues no se parece a ningún otro de la aplicación. Es el encargado de contener todos los datos de interés para la aplicación y se ha implementado usado un patrón singleton para mantener una única instancia en la escena.

El Manager almacena distintos tipos de datos:

* Instancias de las representaciones de objetos, variables, atributos y métodos que luego aparecerán en las escena. De esta manera se reduce el número de instanciaciones de objetos de memoria que se hace.
* Un diccionario con todos los menus de la aplicación, de tal manera que el acceso y búsqueda de estos se puede realizar con una clave que será el nombre del menú. De esta manera los accesos a estos menus serán muy poco costosos.
* Un diccionario con los colores generados aleatoriamente, la razón de esto es tener una comprobación rápida a la hora de comprobar si un color ya ha sido creado con anterioridad
* Una lista con todos los objetos creados, otra con todas la variables creadas y una lista con todos las clases creadas, estas listas se han colocado en este scripts pues es el que se encarga de guardarlas para la persistencia.

Además de los datos el Manager cuenta con unas funciones auxiliares para generar colores y guardar y cargar partida. El guardado se ha realizado usando el sistema de archivos binarios de Unity que permite crear archivos binarios serializando clases con tipos primitivos, el Script posee una función de carga y guardado, la función de guardado es llamada cada vez que se vuelve al menú de inicio y la de carga cuando se pulsa el botón de carga. Como se ha mencionado antes se han creado 4 clases para realizar este guardado y carga, las clases son serializadas por un BinaryFormatter que viene incluido en el lenguaje C#. Para el guardado se extraen los datos necesarios de la aplicación que se guardan en un objeto de la clase principal de guardado y esta se serializa en un archivo binario. Para la carga se realiza el proceso inverso, creando un objeto de la clase principal y creando de nuevo todos los objetos, variables, atributos y métodos a partir de los datos guardados.

Las clases creadas para el sistema de persistencia son las siguientes:

* Atributo: esta clase contiene los datos mas relevantes de un atributo para poder ser creado de nuevo al cargar, esos datos son el nombre y el nivel de acceso, ambos strings.
* Metodo: al igual que el atributo contiene los datos que identifican el método, en este caso es simplemente el nombre del método que es un string.
* Clase: para identificar una clase es necesario guardar dos listas uan para métodos y otra para atributos y además el nombre de la clase que es un string.
* Variable: para identificar la variable se guardan el nombre de esta como string y el nombre de la clase de la variable.
* Save: Este es el archivo principal, guarda una lista de variables y de clases además de una lista de enteros indicando que variables estaban referenciando que objetos, una lista indicando cuantos objetos hay y de que clase son usando un entero y una lista de strings que guarda la información de la consola. Esta es la clase que se serializa en un archivo binario.

Para esta tarea se han creado 4 clases que representan los objetos, variables, clases y demás información util usando únicamente tipos primitivos y colecciones de estos.

### Sistema de menús

Para la creación de los distintos menús se ha creado un script padre que se encarga de la funcionalidad compartida por todos los menús, esta funcionalidad consiste en la activación, desactivación y almacenamiento de los botones. Esta funcionalidad es muy necesaria para que no se produzcan pulsaciones no deseadas con los botones al realizar las transiciones entre menús.

Para el almacenamiento se han guardado los botones en un diccionario de tal manera que cualquier menú pueda acceder a sus botones con el nombre de estos de manera muy rápida.

Heredando de este primer script llamado CustomMenu se han creado el resto de scripts que dan la funcionalidad propia a cada menú, aunque todos tienen sus diferencias en la funcionalidad se pueden dividir los scripts en varios grupos:

**Menús de Creación**

Estos a menús se encuentran dentro de la zona de la aplicación de creación de clases. Comparten la característica de poder elegir un nombre, excepto el menú de creación de métodos, y tener varios botones de selección.

Para la introducción del nombre se ha usado el paquete TextMeshPro que viene incluido en Unity y permite la creación de textos con muy buena resolución independientemente de la distancia. Este paquete incluye un sistema de entrada por teclado que se ha usado para esta tarea.

Además de estas opciones tanto los métodos como los atributos pueden ser modificados antes de terminar de crear la clase. En el caso de los métodos la modificación consiste en elegir otro de los métodos dados al usuario, en el caso de los atributos se puede cambiar tanto el nombre como el tipo y el nivel de acceso.

(fotos de menus de creacion)

Finalmente el menú para crear clases contiene 2 listas que guardan los atributos y los métodos para generar una representación completa una vez que el usuario este de acuerdo. Además de estos 3 menus existen otros 2 menus de creación fuera de esta zona, el primer menú que aparece tras el inicio es usado para crear objetos, explorar el código, abrir el menú de creación de clases y abrir el menú de creación de variables. Este menú se organiza utilizando unas scripts auxiliares que conforman las filas(foto de filas), estas filas guardan toda la información necesaria para los eventos de los botones y la clase que representan.

**Menús de Exposición**

Este tipo de menú se encuentra en el resto de apartados de la aplicación y conforman la parte más interactiva de la aplicación. En el primero de estos menús el usuario puede interactuar con los objetos y variables creados a partir de las clases, puede cogerlos y si acerca la mano puede observar un cartel indicando que objeto o variable es.

Además, puede arrastrar estos objetos para inspeccionarlos, en esta parte de la aplicación al usuario se le muestra el contenido del objeto en atributos y métodos, pudiendo ejecutar estos últimos de la misma manera que inspecciona el objeto.

Para la implementación del menú que muestra objetos y variables se ha creado un script que se encarga de la creación de objetos y variables a partir de las clases creadas, también se encarga de eliminarlas, todas estas copias se almacenan en listas.

En segundo lugar el menú de inspección de objetos y variables está compuesto por otro script que recibe la variable u objeto a inspeccionar y los coloca en el centro del menú para que el usuario pueda interaccionar con ellos.

(foto de menúgrid e inspeccion)

**Menús de Ejecución**

Esta parte de la aplicación está compuesta de unos menús especiales que no heredan de la clase principal de los menús y han sido llamados SubMenus, todos ellos están almacenados en una lista dentro de un menú mas grande que se encarga de recibir el método a ejecutar y hacer visible el submenú correspondiente.

Estos submenús contienen las entradas necesarios para cada método además de la cabecera del método, también se encargan de mostrar el resultado de la ejecución del método.

Los sub menús se encargan únicamente de recoger las entradas, mostrar el resultado y enviar las entradas a los los scripts de los métodos y estos últimos son los que realizan al ejecución del método como tal.

(foto de un submenu)

**Menús Auxiliares**

En esta clasificación entran todos aquellos menús se encargan de apoyar otros menús, existen 2 se estos menús. El primero se encuentra en la zona de creación de la aplicación y es el encargado de mostrar los atributos y métodos que se han elegido para la clase que se va a crear, utiliza una organización en filas similar a la que se utiliza para representar las clases. Este menú existe para que el usuario pueda ver que atributos y métodos ha creado y pueda modificarlos o eliminarlos.

El segundo menú auxiliar aparece en la zona de exploración y ejecución y es una consola que se encarga de representar en forma de texto las operaciones que realiza el usuario con los objetos variables. Este menú sirve como relación entre las interacciones que se realizan y los conceptos que luego se aplicaran en un editor de código.

### Interacción

Este es el apartado con más importancia en el desarrollo de la aplicación pues es la principal diferencia que ofrece con respecto a otras aplicaciones clásicas de aprendizaje sobre la programación.

Para este apartado se ha hecho uso de los Scripts de Interacción que ofrece LeapMotion para Unity, creando clases que heredan o hacen uso de la funcionalidad que se brinda al desarrollador. Se puede dividir la interacción en dos partes, la interacción con objetos y la interacción con la interfaz.

La interaccion con objetos, representados por cubos en la escena, se ha implementado haciendo uso de 2 scripts de interaccion, el primero se llama InteractionBehaviour y permite a un objeto se tocado y cogido con las manos, este script nos permite lanzar un evento para cada tipo de interaccion además de cambiar diversas configuración o bloquear algunas de esas interacciones

(foto del script)

El segundo script que se ha usado se llama AnchorableBehaviour y es el encargado que anclar los objetos a puntos en la escena, de esta manera cuando un objeto no ha sido cogido por el usuario no queda flotando en la escena o perdido en algún punto no accesible. para hacer uso de este comportamiento se han colocado en los distintos menus objetos con un script de LeapMotion que se encarga de convertirlo en puntos de anclaje permitiendo anclar a ellos los objetos, todos estos anclajes son gestionados por un script que poseen todos los objetos físicos interactuadles.

Este script se encarga de recoger los anclajes y scripts de interacción del objeto y realizar operaciones sobre ellos para que el objeto este anclado o no cuando le corresponde.

(foto de anclaje y objeto) (buscar donde meter lo del ratón)

La interaccion con los botones hace uso de un script propio, llamado CustomButton, que hereda del script de LeapMotion InteractionButton y añade métodos para poder bloquear los botones y evitar asi interacciones no deseadas después de una pulsación o durante las transiciones de menus.

Las interacciones se realizan sobre los objetos y botones de manera natural, los botones pueden ser presionados usando las manos y los objetos cogidos acercando la mano a estos y cerrándola. Además, aprovechando los eventos lanzados por las interacciones, se ha añadido un pequeño cartel a todos los objetos interactivos que indica que representa ese objeto, de esta manera se ha podido ahorrar espacio y se aprovecha al máximo la interaccion.

(foto de cartel)

Ambos scripts, el que se ha creado para los objetos y el que se ha creado para los botones, añaden funcionalidad para el ratón, de tal manera que el usuario pueda realizar las mismas acciones con un dispositivo LeapMotion que con un raton. ---Esta decisión se tomo pronto en el desarrollo pues de esta manera la aplicación se podía hacer llegar a usuarios que no pudieran permitirse un dispositivo LeapMotion ofreciendo la misma interaccion. No obstante, se pierde libertad de movimiento e inmersión en los conceptos???---(no se si meter esto).

### Transiciones

El sistema de menús se apoya en un sistema de transiciones entre estos, para la creación de este sistema se ha hecho uso de un script de LeapMotion que permite, dándole 3 objetos, hacer la transición de posición rotación y escala de uno de ellos entre la posición escala y rotación de los otros 2.

Este Script tiene otra peculiaridad muy importante, el script lanza un evento siempre que se llega o abandona la posición inicial o final de la interpolación, estos eventos han sido muy importantes a la hora de solucionar problemas con interacciones no deseadas.

(foto de script tween)

### Problemas en el Desarrollo de los Scripts

En el proceso para desarrollar los scripts han surgido diversidad de problemas sobre todo al añadir nuevas funcionalidades, estos problemas no estaban fuera de los cálculos del proyecto y no han supuesto un retraso considerable. Sin embargo, hay un problema que se ha llevado a lo largo de todo el proyecto hasta su completa solución, este problema tiene que ver con la interacción con los botones.

Si bien esta interacción funcionaba de forma correcta, es decir, los botones respondían a todo contacto, esta no era la interacción que se buscaba siempre. En ocasiones al cambiar de menú o pulsar sobre un botón se producían dobles pulsaciones o se pulsaba un botón de otro menú nada mas aparecer en pantalla, esto era poco intuitivo para el usuario y producía problemas.

La solución a la que se ha llegado para solucionar esto consiste en varios métodos en el script antes mencionado que se encarga de las interacciones con los botones. Al funcionar estos con físicas en primera instancia se pensó en bloquear la función que se encargaba de actualizar todo el sistema de detección y movimiento del botón, esta aproximación solucionaba algunos problemas pero se generaban otros. Otra aproximación que se valoro fue desactivar las físicas del botón unos segundos tras cambiar de menú o pulsar un botón, esta solución funcionaba muy bien al pulsar botones pero no tenia ningún efecto al cambiar de menu. Finalmente se llego a una solución hibrida que bloquea la función de actualización en los cambios de menú, cuando no da problemas con las interacciones, y desactiva las físicas cuando se pulsa un botón. De esta forma se ha conseguido que la interfaz responda de manera correcta e intuitiva.

## Diseño de Interfaz

El diseño de la interfaz ha ocupado una gran parte del desarrollo, esta interfaz ha seguido el mismo proceso iterativo que el resto de elementos de la aplicación. El principal elemento ha tener en cuenta para la creación de la interfaz ha sido la limitación de espacio. La aplicación se localiza en un espacio limitado al no poder moverse la cámara pero tiene que seguir un paradigma de RV donde todos los objetos deben tener cierto tamaño para ser interactivos.

Se comenzó prototipando una primera interfaz con cubos y adaptándola, al final se tomo la decisión de organizar la interfaz sobre tres paredes en frente del usuario una central totalmente paralela a la cámara y otras 2 a los lados giradas 30º cada una, esta decisión se ha tomado por la forma en que el ser humano mueve sus brazos, estos forman un arco alrededor del usuario por lo tanto la interfaz no puede ser totalmente plana y debe estar colocada en un arco.

(foto de las paredes)

Otra característica de la interfaz es que la mayoría de elementos se centran en el centro y a una altura intermedia, esta decisión se ha tomado por las limitaciones del dispositivo LeapMotion, este dispositivo aunque detecta las manos en un área cercana al de una semiesfera de un radio del alrededor de 1m, pierde mucha calidad en la detección cuando las manos están muy abajo o muy cerca de los extremos.

(Foto de interfaz con dibujo)

### Optimizacion de Renderizado

En cuanto a las transiciones de los menus, antes de llegar a la solución arriba descrita se realizaron pruebas desactivando los menus para reducir el consumo de la tarjeta grafica al pintar todos los botones. Esta solución no funciono como se esperaba pues daba grandes problemas con el comportamiento de los botones al reactivar los menus. Finalemente se llego a la solución de las transiciones por movimiento arriba descrita y para reducir el consumo del pintado de las mallas se ha hecho uso de una herramienta de Unity que se llama Occlusion Culling.

El Occlusion Culling consiste en la división de la escena en cuadrantes de tamaño definido por el desarrollador, usando estos cuadrantes el motor se encarga de renderizar todas aquellas mallas que estén en los cuadrantes que ve la camara y en los cuadrantes adyacentes al cuadrante en el que se encuantra la camra, de esta manera se ahorra mucho computo en renderizar mallas que no son visibles

(foto del culling)

### Accesibilidad

La ultima consideración que se ha tenido con la interfaz es la accesibilidad, se ha elaborado la interfaz de tal manera que sea accesible para personas con deficiencia visual de colores. Para esta tarea se ha dado otro canal para transmitir la información a través de formas o textos en los botones y carteles informativos sobre todos los objetos interactivos.

(rueda de colores)

### Multilenguaje

Tambien se ha dotado a la aplicación de dos idiomas, para ello se ha hecho uso de un asset gratuito descargado de la Asset Store de Unity, el asset se llama Polyglot Tool. Este asset nos permite definir una lista de idiomas y para todos estos idiomas una lista de claves y su correspondiente valor, este valor será distinto para cada idioma y será la traducción de un elemento para cada idioma.

(fotos del menu)

Para poder cambiar el idioma en ejecución tiene que existir un script con la clave en cada texto que se desee cambiar.

(foto de script)

Tambien tiene que existir en la escena un objeto con un script que será el encargado de cambiar el idioma de todos los textos cuando se le ordene, se guarda una referencia a este script en el Manager.

(Script de idioma.)

Esta solución no se ha podido aplicar a algunos de los textos que se escriben en ejecución dependiendo de las acciones del usuario, para estos textos se ha hecho uso de un booleano guardado en el Manager que nos indica en que idoma estamos.

### Modelado

Al ser una interfaz para realidad virtual ha hecho falta crear los elementos de 0 en un programa de modelado, en este caso se ha elegido Blender. Para este proyecto se han modelado varios elementos que se pueden dividir en botones y otros elementos.

En el modelado se ha buscado reducir al mínimo el numero de polígonos pues van a exisitir varios elementos por menú, esto dispararía el consumo al pintar la mallas. Esta reducción a su vez se apoya en las herramientas del motor para reducir el numero de llamadas de pintado al compartir los botones de un mismo tipo malla y material.

**Botones**

El objetivo de los botones es producir una reacción a las acciones del usuario, se ha intentado usar formas que se diferencien y cuando no era posible se han usado textos, de esta manera los colores no son el único estimulo que recibe el usuario consiguiendo asi accesibilidad para usuarios con deficiencia visual de colores.

(fotos de botones en blender)

Siguiendo un diseño parecido a los botones bidimensionales se han elaborado modelos modulares para algunos botones de tal forma que no se deformen la laterales y esquinas al escalarlos. Para agilizar la colocación de las piezas de estos botones modulares en Unity se han elaborado algunos scripts que se encargan de colocar todos los elementos de un botón en las posiciones que le corresponden además del redimensionar el texto que contiene el botón

(foto de script y piezas del botón en blender)

**Otros modelos**

A parte de los botones se han creado otros modelos. El primero consiste en un fondo con un marco, este objeto se ha usado en el menú que representa la consola, en el menú que muestra el código java y en el menú de la guía de uso. Este marco se ha modelado modularmente al igual que los botones, de manera que los marcos mantienen su grosor aprovechando asi el máximo de espacio. Para su uso en Unity se ha elaborado un script muy parecido al que se usa para el botón.

(foto del marco en blender)

El segundo modelo que se ha creado ha sido una papelera, esta papelera se usa para indicar la eliminación de objetos y va siempre acompañada de un texto superior como apoyo.

(foto de papelera en blender y Unity)

**Implementacion en Unity.**

Finalmente para acabar con la parte de modelado se han exportado los modelos en FBX, un formato compatible con la mayoría de motores y en este caso Unity, una vez exportados basta con introducir los archivos en cualquier carpeta del proyecto y Unity será capaz de reconocerlos y renderizar la malla correspondiente si necesidad de realizar ningún cambio o trabajo.

### Problemas en el Desarrollo de la Interfaz

El principal problema que se ha encontrado al desarrollar la interfaz ha consistido en el espacio limitado que se tenía para la colocación de todos los elementos, esto ha supuesto la necesidad de separar la interfaz en varios menús. Se ha conseguido encontrar un equilibrio entre el tamaño de los elementos, la cantidad de elementos y el numero de menus de manera que se pueda acceder a todos ellos de manera sencilla sin tener que pasar por una cantidad excesiva de menus y sin que los elementos estén muy pegados.

Para eliminar elementos excesivos muchas veces se ha buscado, tras varias iteraciones, una solución que se acercara a la interaccion con los objetos en vez del uso de botones, este es el caso de la acción de eliminar que en toda la aplicación se ejecuta arrastrando el objeto a una papelera.

De esta manera se ha conseguido llegar a una interfaz que aprovecha al máximo el LeapMotion sin usar un dispositivo de realidad virtual.

# Conclusiones

El desarrollo de la aplicación ha sido un proceso enriquecedor por las tecnologías usadas, princpalmente el LeapMotion. Comenzare hablando de los puntos más personales.

Estas tecnología era desconocida para mí antes de comenzar el desarrollo de este proyecto y ha conllevado un proceso de investigación previo a la implementación bastante extenso, pero una vez superada esta barrera la sensación de añadir elementos que funcionaran con el LeapMotion era gratificante.

Ademas del aprendizaje de esta nueva tecnología el trabajo también ha conllevado el aprendizaje del desarrollo de una interfaz completa y desde 0, cosa con la que antes no se había trabajado, además esta interfaz sigue un paradigma de realidad virtual que es aun mas raro de encontrar.

Por parte de la aplicación, se han logrado todos los objetivos. En cuanto a la creación de clases se ha logrado crear un sistema satisfactorio para el usuario que le da todas las opciones que necesita para el uso correcto de la aplicación sin necesidad de escribir código.

Otro objetivo que se ha cumplido ha sido el de permitir al usuario la modificación de las clases, este objetivo va de la mano del anterior y se ha conseguido realizar de manera intuitiva y sencilla para el usuario.

En cuanto a la interacción esta principalmente basada en botones para los menus, esta decisión fue tomada por las características de la interfaz y el mundo virtual y por las limitaciones de la del LeapMotion. Se considera que se cumple este objetivo de manera satisfactoria para el usuario.

La segunda parte de la interaccion consistía en los objetos que representabas las variables, atributos, métodos y objetos, esta interacción se ha conseguido realizar de una manera natural pudiendo coger los objetos mirarlos, explorarlos y eliminarlos solo con las manos.

Por último, un objetivo muy importante que se ha logrado ha sido el de la ejecución de métodos, este objetivo nace como una funcionalidad extraida de otra aplicación llamada BlueJ y añade mucho valor a la interaccion de la aplicación completando asi todo el proceso de creación y uso de una clase.

Finalmente me gustaría comentar algunos trabajos futuros que se han valorado para esta aplicación:

* Realidad virtual: Un cambio que resultaría increíblemente positivo para este tipo de aplicación seria el uso de unos cascos de realidad virtual. Esta mejora tendría consecuencias muy grandes en el desarrollo pues habría que cambiar toda la interfaz, pero supondría una mejoría en interaccion del usuario con el mundo y potenciaría asi ese factor de interaccion en el que la aplicación se apoya para conseguir sus objetivos.
* Mas gestos: Por las limitaciones del hardware se ha recurrido a una interfaz basada en botones, si la aplicación pasase a una realidad virtual completa, la colocación del LeapMotion sobre el casco de realidad virtual mejoraría enormemente la detección y permitiría la implmentacion de mas gestos que podrían sustituir a botones
* Mejor interfaz: como se ha mencionado antes la interfaz tendría que ser rehecha si se cambiara a una realidad virtual completa usando un casco de realidad virtual. Se apuesta por la creación de un mundo virtual donde el usuario pueda moverse, en el que se usen objetos cotidianos para la explicación de los conceptos. Un ejemplo seria un aula con una mesa en la que estén los objetos y puedas interactuar con ellos totalmente, si hubiera necesidad de algún menú estos podrían estar a tu alrededor y no fijos en una pared.

# Bibliografia