

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

**Directores**: Maximiliano Paredes

Índice

[Resumen 5](#_Toc8750485)

[Introducción 7](#_Toc8750486)

[Objetivos 7](#_Toc8750487)

[Fundamentos 9](#_Toc8750488)

[Tecnologías 9](#_Toc8750489)

[Desarrollo 14](#_Toc8750490)

# 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 mas directa y visual con los conceptos que se explican en las aulas.

La aplicación está pensada para los cursos iniciales 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 LeapMotion se da la opción al usuario de usar el ratón

# Introducción

## Objetivos

# Fundamentos

## Tecnologías

### Unity

Unity[1] 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.

### 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.

### 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.

### 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.

# Desarrollo

## Resumen

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 esta 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 interaccion gestual.

## 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 es de objetos y variables en la escena, esta parte se ha solventado colocando los objetos en una cuadricula, permitiendo interactuar con todos 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.

Por último ha habido una parte del desarrollo que ha afectado a todas las partes del desarrollo y del diseño de la interfaz y es la interacción gestual basada en LeapMotion, este punto ha afectado a todo el desarrollo pues el enfoque es muy distinto de una aplicación tradicional como puede ser BlueJ. Ha habido que enfocar 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 e 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.

## Scripts

### Manager

Se ha creado un Script Manager que es el encargado de contener todos los datos de interés para la aplicación, para la implementación de este script se ha 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.

Para esta tarea se han creado 4 clases que representan los objetos variables y clases 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 métodos, y tener varios botones de selección. Para la introducción del nombre se ha usado el paquete TextMeshPro que viene con incluido en Unity que 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. Ademas de estas opciones tanto los métodos como las clases 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. 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. Ademas 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) con todos sus botones e información.
* 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 explorarlos, 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 copias de la clase y copias de la variables, 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.