Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Mejora del aprendizaje mediante entorno virtual en Unity

Autor: Javier Fernández-Coto Córdoba

Tutor/es: Julia Clemente Párraga

2021

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado
Mejora del aprendizaje mediante entorno virtual en Unity

Autor: Javier Fernández-Coto Córdoba

Tutor/es: Julia Clemente Párraga

TRIBUNAL:

Presidente: María del Mar Lendínez Chica

Vocal 1º: Concepción Batanero Ochaíta

Vocal 2º: Julia Clemente Párraga

FECHA: 03 de Noviembre de 2021

Agradecimientos

Comienzo este documento tan importante con una sección que, aunque no es obligatoria, sí es una parte esencial, pues ha sido, en cierta manera, el motor principal de este trabajo y de todo el proceso que ha conllevado. Para mí, esta es una de las secciones más difíciles de redactar, ya que es imposible plasmar en palabras el apoyo que he recibido y el agradecimiento y el cariño que he sentido hacia estas personas con las que cierro una etapa muy importante de mi vida.

En primer lugar, quiero agradecer a mi familia su incansable apoyo durante mi formación en la universidad. Desde un principio han sabido ayudarme en los momentos que lo necesitaba y esto es algo que quiero dejar reflejado en este documento que da carpetazo a esta etapa de mi vida. Han valorado todo el esfuerzo que es necesario realizar para poder llegar a este nivel de formación.

También, quiero hacer saber a mi tutora de este trabajo, Julia Clemente Párraga, mi eterno agradecimiento por estar conmigo durante todo este proceso, y por haberme guiado en lo que ha sido mi cierre de etapa universitaria. Sus consejos y su interés han hecho posible que este trabajo haya llegado a tan buen puerto.

Por supuesto, no puedo olvidar mencionar a Beatriz B. porque sin su estimado apoyo y consejo no hubiera llegado a escribir todo este trabajo. La energía que me ha aportado ha sido clave para la culminación de este trabajo.

Finalmente, he de mencionar a mi grupo de amigos que han llegado a desear este momento casi más que yo, y que a su manera también han sido un gran apoyo moral.

Índice general

Índice de figuras

Índice de tablas

Resumen

En el ámbito educativo se dispone de múltiples técnicas y herramientas para facilitar al alumno el aprendizaje. Debido a las dificultades que presenta el aprendizaje de conceptos abstractos y algoritmos de una asignatura técnica perteneciente a una titulación superior como es Sistemas Operativos, se plantea el objetivo de mejorar su aprendizaje mediante una simulación en un entorno virtual basado en la tecnología de Unity. Para su desarrollo se aplicará la metodología XP. La mejora del aprendizaje va ligada a la reducción de costes de los recursos docentes, la eliminación de restricciones geográficas y mejora de la adquisición de conocimiento del alumno.

Palabras clave

Unity, Entorno virtual, Script, Aprendizaje, Sistemas Operativos.

Summary

In the educational field, multiple techniques and tools are available to help students through their learning. Due to the difficulties in learning abstract concepts and algorithms in a technical subject belonging to a higher degree, such as Operating Systems. The aim is improving learning by simulations in a virtual environment based on Unity technology. The XP methodology will be applied for its development. The improvement of learning is linked to the reduction of the cost in teaching resources, the elimination of geographical restrictions and the improvement of the student's knowledge acquisition.

Keywords

Unity, Virtual environment, Script, Learning, Operating Systems.

Parte I

Resumen extendido

En el aprendizaje se usan una variedad de métodos, técnicas e instrumentos diferentes, realizados y utilizados para transmitir un conocimiento.

Con la aparición y aplicación de las nuevas tecnologías multimedia, se ha revolucionado el ámbito de la enseñanza de tal manera que ahora, conceptos prácticos cuya realización era inasequible como para usarlos en la enseñanza ahora son usados por estudiantes de todo el mundo para su formación.

Como ejemplo de estas nuevas técnicas podemos destacar la gamificación, técnica consistente en la aplicación de la mecánica de los juegos para implementar conceptos educativos de manera lúdica. El objetivo del proyecto consiste en el perfeccionamiento del aprendizaje de los diferentes conceptos abstractos de una disciplina referente a unos estudios superiores realizando una simulación en un entorno virtual ejecutado en Unity.

En el marco teórico serán expuestos varios conceptos clave que nos serán útiles para comprender el proyecto. El progreso de la enseñanza está estrechamente relacionado con las denominadas plataformas de enseñanza. Esta tecnología ayuda a organizar y valorar el progreso del aprendizaje de los estudiantes.

Ya que el aprendizaje es algo muy presente a diario y evoluciona con nosotros, la tecnología nos ha brindado una serie de infinitas posibilidades y ventajas con las que podemos trabajar. Debido a que en asignaturas técnicas, donde surgen conceptos abstractos y algoritmos cuyo aprendizaje puede resultar más complejo, surge la necesidad de buscar soluciones que faciliten este proceso de adquisición de conocimientos. En el presente trabajo se diseñará y se aplicará una herramienta basada en un entorno virtual para simplificar y facilitar el progreso de aprendizaje del estudiante.

La arquitectura de la herramienta desarrollada es planteada, con los diferentes componentes que la conforman responsables de llevar a cabo cada una de las tareas. Este proyecto se desarrollará principalmente usando la tecnología de Unity lo que permitirá crear los elementos y recursos para lograr los objetivos establecidos. Unity es una plataforma de desarrollo de contenido interactivo en tiempo real, tanto 2D como 3D y permite la creación de juegos, aplicaciones y entornos virtuales.

Para realizar un proyecto de estas dimensiones, que requiere de una gran cantidad de recursos y acciones a realizar, es aconsejable el uso de metodologías de desarrollo de software para finalizar el trabajo satisfactoriamente y tomando así las ventajas y beneficios de las mismas, entre otras, crear un producto de calidad y aumentar la productividad.

Como resultado final, la implementación de este método de enseñanza virtual va a complementar la metodología de enseñanza tradicional, sobre todo en asignaturas con contenidos técnicos y

procedimentales, simulando un aula en la que el estudiante puede interactuar por medio de un avatar.

Adicionalmente, habría que tener en cuenta posibles trabajos a futuro dedicados a la optimización de recursos, la carga de cursos y temario, ampliación de actividades, internacionalización de la interfaz, etc.

Parte II

Memoria del trabajo

Capítulo 1

Introducción

1.1. Motivación del trabajo

Este primer capítulo contiene una breve descripción de los motivos por los que se desarrolla este trabajo con el fin de presentar al lector los objetivos a alcanzar y la estructura del mismo.

En el ámbito del aprendizaje se utilizan distintas técnicas e instrumentos para facilitar al alumno la comprensión de los diversos conceptos subyacentes a la materia que se está estudiando. Sin embargo, en numerosos escenarios el alumno tiene dificultades para asimilar ciertos aspectos académicos. Por ello, se requiere utilizar diferentes recursos más allá de un libro, una pizarra y una clase magistral.

Los mecanismos pedagógicos con los que se cuenta habitualmente transmiten los conocimientos a través de los distintos canales (auditivo, visual y sensitivo). Estos canales de transmisión permiten diferentes combinaciones para hacer énfasis en la idea o concepto que se desee presentar al alumno, como puede ser el paradigma de la Programación Orientada a Objetos (POO), en el cual se utilizan modelos basados en conceptos del mundo real para el desarrollo de aplicaciones [Rodríguez García, 2006]. Generalmente, estos conceptos presenta una complejidad evaluada para el alumno que está adquiriendo estos conocimientos.

Actualmente, con la inclusión de las nuevas tecnologías multimedia en el ámbito educativo, se dispone de nuevos métodos y técnicas que facilitan al docente la labor de presentación de nuevos conceptos a sus alumnos. Algunas de ellas, permiten aumentar el dinamismo de las explicaciones fomentando la capacidad de interacción del alumno en el proceso de aprendizaje. Entre estas nuevas técnicas docentes, cabe destacar la simulación basada en entornos virtuales de aprendizaje que permiten a los alumnos enfrentarse a experiencias de aprendizaje de difícil recreación en el mundo real o de alto coste. Un ejemplo de estas nuevas técnicas que se utilizan en el proceso de aprendizaje es

la aplicación del concepto de gamificación. Esta técnica aplica la mecánica de los juegos para explicar los conceptos abstractos de una manera lúdica, generando una experiencia eficaz, que refuerza (de forma positiva) el aprendizaje o mejora de los conocimientos objetivos que se plantean.

En el desarrollo de entornos virtuales se puede optar por una interfaz realista que simule un mundo virtual a semejanza del mundo tangible. Para este fin, se hace uso de motores gráficos, que son herramientas que permiten generar todo lo relacionado con la física del juego y la parte visual. Estos, a su vez, tienen capacidades para engendrar un entorno basado en 2D o en 3D. La situación actual en el mundo del desarrollo de videojuegos (o animaciones) está marcada por los dos principales entornos que son accesibles por cualquier desarrollador. Estos son el motor gráfico Unreal Engine, creado por la compañía *Epic Games* en 1998, y el motor gráfico multiplataforma Unity, creado por *Unity Technologies* en 2005. Unity es uno de los motores gráficos más utilizados en los desarrollos actuales de videojuegos, animaciones y/o aplicaciones. Esto se debe a las principales ventajas que diferencian a Unity de la competencia. La primera característica es la facilidad de uso en el diseño e implementación; cuenta con una interfaz sencilla con diferentes paneles en los que muestra la jerarquía de objetos e incluso podemos obtener una previsualización del entorno, muy útil a la hora de diseñar. También ofrece la posibilidad de realizar los *scripts* necesarios en tres lenguajes diferentes: *C#* (de uso mayoritario), *Javascript* o *Boo*. Cabe destacar también la gran potencia que proporciona tanto en entornos 2D y 3D junto con la posibilidad de exportar los desarrollos a más de 20 plataformas distintas. Adicionalmente, Unity cuenta con una gran comunidad de desarrolladores que utilizan este motor, lo que facilita la resolución de dudas y problemas que se puedan plantear durante todas las fases de desarrollo. Finalmente, para aportar flexibilidad, es interesante poder integrar Unity en las diferentes plataformas de enseñanza (del inglés, *Learning Management System*, LMS) que se usan actualmente, como son *Moodle*, *Blackboard*, *Evolvampus*, etc. En este sentido, la mayoría de LMS, tanto *Open Source* como SaaS, se orientan al mundo web. Por todo lo anterior, Unity ofrece gran facilidad para ser exportado a sistemas web.

1.2. Objetivos y tareas

El principal objetivo del proyecto consiste en la mejora del aprendizaje de conceptos abstractos de una asignatura técnica perteneciente a una titulación superior llevando a cabo la simulación en un entorno virtual bajo el motor Unity. A continuación, se enumeran los objetivos específicos del trabajo:

1. Estudio y análisis de los entornos virtuales de aprendizaje; definición exacta del objetivo final al que vamos a llegar.

2. Estudio en profundidad del motor gráfico Unity, que va a ser fundamental para el desarrollo del proyecto.

3. Selección de una asignatura técnica, como puede ser la disciplina de Sistemas Operativos, y sus conceptos abstractos/algoritmos candidatos a ser virtualizados en el trabajo.

4. Análisis de requisitos en el desarrollo del entorno virtual para el aprendizaje mediante la definición, entre otras, de las funcionalidades necesarias para mejorar la enseñanza.

5. Diseño del entorno virtual según los requisitos funcionales.

6. Desarrollo del entorno virtual mediante las herramientas de desarrollo de Unity, así como librerías de objetos(*assets*) disponibles.

7. Redacción de la memoria del proyecto, detallando todo el proceso de diseño y desarrollo del entorno virtual de aprendizaje.

Las tareas que van a permitir alcanzar estos objetivos son variadas y se describen a continuación:

- Tarea 1: Realizar un estudio previo de las tecnologías disponibles y su funcionamiento.
  - Tarea 1.1: Entornos virtuales (2.3)
  - Tarea 1.2: Gamificación (2.1)
  - Tarea 1.3: Sistema LMS (2.2)
  - Tarea 1.4: Unity (4.2)
- Tarea 2: Realizar un estudio de la metodología más apropiada para el proyecto.
  - Tarea 2.1: Modelo de Prototipo
  - Tarea 2.2: Metodologías Ágiles
  - Tarea 2.3: Metodología XP (*Extreme Programming*)(4.3)
- Tarea 3: Realizar los diagramas necesarios para obtener un diseño funcional e implementable.
  - Tarea 3.1: Diagramas de casos de uso
  - Tarea 3.2: Diagrama de secuencia
- Tarea 4: Implementar del entorno virtual de aprendizaje y los objetos de aprendizaje.
- Tarea 5: Realizar pruebas del entorno virtual desarrollado.
- Tarea 6: Analizar resultados del proyecto.

1.3. Estructura del trabajo

Este Trabajo Final de Grado se ha estructurado en diferentes partes, capítulos, secciones y subsecciones. En la primera parte, se ha desarrollado el cuerpo del proyecto, la memoria del trabajo. La segunda parte del proyecto contiene los apéndices que aportan otra información esencial al proyecto como son el presupuesto o el pliego de condiciones entre otros. Finalmente, la última parte del proyecto contiene la bibliográfica, sección del documento donde se sitúan todas las referencias usadas a lo largo del proyecto.

A continuación se pasa a detallar los capítulos que conforman la memoria del trabajo:

1. El primer capítulo, desarrolla la introducción al proyecto donde se incluye la motivación del trabajo, los objetivos y tareas del mismo, y la propia estructura del mismo.
2. Este capítulo se centra en el marco teórico sobre el que se centra el proyecto, que comprende conceptos como la gamificación, la plataforma de enseñanza (LMS), entorno virtual de aprendizaje y mundo virtual y la tecnología actual. Todos ellos necesarios para la correcta comprensión del proyecto.
3. En el siguiente capítulo se expone el planteamiento del problema, que comprende el ámbito de aplicación del proyecto, objetivos, restricciones y supuestos del trabajo que se asumen antes de empezar el proyecto. Todo esto es fundamental para situar el proyecto, conocer cuál es el origen y cuál es la meta que se pretende alcanzar.
4. El cuarto capítulo es donde se sitúa la solución adoptada, en el cual se detalla la arquitectura utilizada, la tecnología usada, metodología de desarrollo del proyecto además del diseño de la solución. También se detalla la implementación de la solución así como las pruebas realizadas y el análisis de resultados.
5. El quinto capítulo expone las conclusiones desarrolladas a lo largo del proyecto.
6. En el sexto y último capítulo se analizan las posibles líneas de trabajo a futuro que puede seguir el proyecto basándose en la experiencia obtenida en el desarrollo, a las futuras tendencias y nuevas tecnologías que puedan ofrecer una mejorar al sistema.

El final del documento contiene los apéndices en los que se exponen el pliego de condiciones y el presupuesto para el desarrollo del proyecto, además de la bibliografía que cuenta con todas las referencias usadas para el desarrollo del proyecto.

Capítulo 2

Marco teórico

En el marco teórico se van a exponer los diferentes conceptos clave que fundamentan las bases del proyecto. Estos conceptos son necesarios para comprender de forma satisfactoria el proyecto, cuáles son sus bases y el contexto en el que se enmarca el mismo.

2.1. Gamificación

Definición

Desde hace 3000 años existe el concepto de los juegos entre nosotros como método de entrenamiento para diversas habilidades como pueden ser las instintivas, las cognitivas o aquellas relacionadas con la locomoción. No obstante, las primeras manifestaciones de los videojuegos como los conocemos hoy en día a través de una pantalla y mediante un sistema HID (*Human Interface Device*) aparecieron en los años 70. Un claro ejemplo son las máquinas *Pong* de los Pubs de aquella época. Posteriormente, en la década de los 80 y 90, la empresa japonesa Nintendo consigue posicionarse como referente en el mercado y en el mundo de los videojuegos. Gracias a la capacidad que tuvo a la hora de ofrecer a todos los perfiles de clientes una gran variedad de productos que permitían acercar los videojuegos al público general. Además, se puede denominar al siglo XXI como la Edad de Oro de los videojuegos, debido tanto al gran desarrollo que estos han llevado a cabo como a su evolución. Sin embargo, el detonante de su gran éxito es, sin duda, el uso de internet, que ha sido su herramienta principal de expansión e interconexión de los usuarios.

Aquí es donde toma partido la denominación de **gamificación**. Este término se entiende como técnica de aprendizaje que traslada la mecánica de los juegos al ámbito educativo-profesional, con el fin de conseguir mejores resultados, ya sea para facilitar la asimilación de algunos conocimientos, mejorar alguna habilidad, o bien recompensar acciones concretas, entre otros muchos objetivos.

El principal objetivo de esta técnica es influir en la conducta psicológica y social de los usuarios. Para alcanzar dicha finalidad, hay que aplicar las mecánicas de juego para así poder incentivar al usuario, ya sea para aprender un nuevo concepto o para recompensar a un cliente que adquiere un producto determinado a una empresa. Este tipo de aprendizaje va ganando terreno en las metodologías de formación teniendo en cuenta su carácter lúdico, que facilita la interiorización de conocimientos de una manera más positiva en el usuario, generando una experiencia de gran valor y por tanto de mayor persistencia.

Aquellos mecanismos, que se han mencionado con anterioridad, de los que se vale la gamificación, se engloban dentro de la teoría del **Modelo de Fogg**. Este modelo permite un cambio de comportamiento de los individuos fundamentado en tres conceptos (véase figura 2.1):

- **Disparador (*Trigger*):** El primer concepto se conoce como *Trigger*, es decir, el detonante o la acción que desencadena un determinado comportamiento.
- **Habilidad:** El segundo concepto hace referencia a la habilidad, que se interpreta como un valor que indica la capacidad técnica que un individuo pueda demostrar a la hora de ejecutar el comportamiento deseado.
- **Motivación:** El tercer concepto, motivación, define la posibilidad que tiene un individuo de alcanzar el objetivo deseado mediante la superación de sus limitaciones.

En el camino hacia la gamificación, el Modelo de Fogg es una herramienta más junto con el diseño de los elementos de juego, teniendo en cuenta sus mecánicas y sus dinámicas.

Este modelo de juego puede probar que realmente funciona, ya que consigue motivar a los alumnos haciendo que estos desarrollen un mayor compromiso hacia las personas y que incentiven su ánimo de superación, motivándoles. Para alcanzar a cabo dicho objetivo, se utilizan una serie de técnicas mecánicas y dinámicas extrapoladas de los juegos.

Características

La metodología de aprendizaje que propone la gamificación se basa principalmente en aplicar diferentes técnicas provenientes de los juegos, para que se pueda facilitar y potenciar tanto la enseñanza como el aprendizaje. Por ello, cabe que destacar las siguientes características relacionadas con la gamificación:

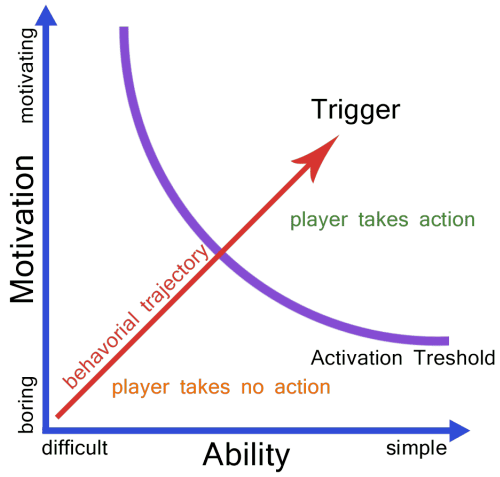

Figura 2.1: Diagrama de Fogg [MaxSchlutter, 2013]

a) Mediante el uso de mecánicas de juego para las recompensas, basadas en el condicionamiento clásico[1], un estímulo condicionado produce una respuesta condicionada.

b) Crea sensación de autonomía en la persona favoreciendo un cambio de comportamiento en el usuario.

c) Fomenta la participación mediante un espacio que será atractivo a ojos del usuario.

Cómo diseñar un sistema de gamificación

De acuerdo con [Teixes, 2014], al comenzar un proyecto de gamificación hay que tener en cuenta que el 80 % de los proyectos que vayan a hacer uso de un entorno virtual de aprendizaje no van a tener éxito. La mayor causa de este tipo de situaciones se debe al mal diseño de los sistemas. Como resultado es importante tener en cuenta un modelo de éxito para el proyecto que se desee llevar a cabo. Hay que destacar cuatro modelos: el primer modelo está basado en las 6D y lo propuso Werbach (véase figura 2.2). En dicho modelo, se tienen en cuenta cuáles son las metas del proyecto desde un primer modelo y la evolución del proyecto dependerá de estas. Las conductas objetivas, la descripción de los jugadores y los bucles de actividad van a estar sujetos a los objetivos del proyecto, teniendo en cuenta dos aspectos muy importantes que todo entorno virtual de aprendizaje debe tener: la diversión y las herramientas indicadas para llevarla a cabo.

El segundo modelo lo propuso Víctor Manrique (véase figura 2.3), el célebre creador del blog *Epic Win*. El principal motor es buscar la razón por la cual se va a llevar a cabo el proyecto usando

[1]Condicionamiento Pavlovliano denominado así por Iván Pávlov, Premio Nobel en 1904

Figura 2.2: Modelo Werbach [Teixes, 2014]

la gamificación, cómo es posible realizar averiguando qué acciones se deben realizar. También se tienen en cuenta los jugadores destinados, pues se puede buscar un perfil concreto de jugador. Por último, hay que idear cómo va a ser el sistema y cómo este puede mejorar los tiempos de implicación y visita, hacerse viral e incrementar la calidad de los resultados.

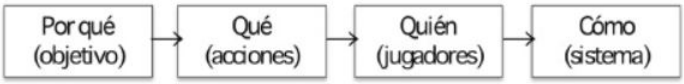

Figura 2.3: Modelo Víctor Manrique [Teixes, 2014]

El tercer modelo lo elabora Jamaki Kumar (véase figura 2.4). Aquí el diseño del entorno virtual de aprendizaje va a basar su diseño enfocándose en el usuario. Primero se tiene en cuenta un perfil concreto del participante y a partir de ahí se comienza la elaboración del proyecto de gamificación: se especifica el objetivo que se quiere alcanzar con el proyecto y cómo se va a poder motivar al usuario. Una vez se tienen claros los puntos anteriores, se pasa a la aplicación de las mecánicas necesarias para gestionar el entorno y controlarlo, y a la vez medir sus resultados.

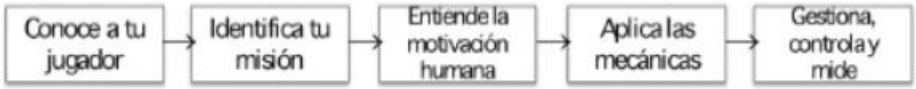

Figura 2.4: Modelo Jamaki Kumar [Teixes, 2014]

El cuarto y último modelo tomado en cuenta por [Teixes, 2014], se elabora gracias a una serie de comunicaciones con la consultora, Craig Ferrara, que trabajaba para la empresa norteamericana Gygia. Se comienza definiendo los objetivos del proyecto de gamificación que se quiere llevar a cabo (véase figura 2.5); sobre todo hay que especificar a quiénes va dirigido. Una vez se tienen todos estos puntos clave, se procede a la construcción del sistema de gamificación, se lleva a cabo su implementación en el entorno virtual, y finalmente hay que mantener y llevar un seguimiento del proyecto, pudiendo corregir los diferentes fallos que tenga y qué mejoras se pueden realizar.

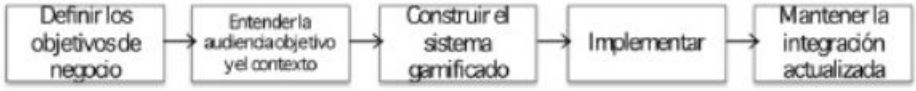

Figura 2.5: Modelo a Gygia, Craig Ferrar [Teixes, 2014]

Resumiendo estos cuatro modelos, Teixes elabora un quinto que resulta de los anteriores (véase figura 2.6). Sin embargo, hay un aspecto que lo diferencia del resto, y es el comienzo del proyecto. Se deben tener en cuenta cuáles son los objetivos y por qué se va a realizar con la ayuda de la gamificación. Una vez se obtienen las respuestas a tales preguntas, se tendrá claro cuál será la audiencia, el objetivo y el contexto del proyecto de gamificación. A continuación, se pasa a la delineación de las conductas que se desea transmitir a los usuarios que nos ayudarán con el desarrollo del entorno virtual del aprendizaje, y a los elementos de gamificación de los que se hará uso y pasar a su implementación.

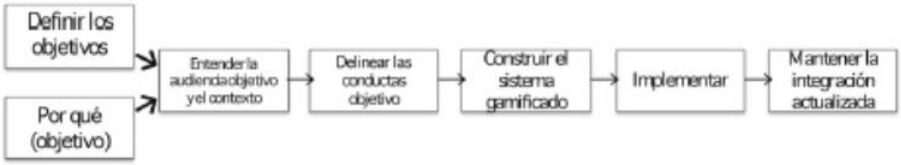

Figura 2.6: Modelo propuesto por [Teixes, 2014]

Por último, es muy importante actualizar el sistema de gamificación y llevar un mantenimiento de este, pues se podrá ver qué errores hay y qué soluciones los pueden contrarrestar y, a su vez, verificar los resultados que se van obteniendo en el futuro.

Ejemplos de éxito

En la mayoría de los casos en los que se aplica la gamificación, el éxito de la misma se basa en el cumplimiento de una serie de métricas que puedan evaluar la obtención de un mencionado logro o medalla.

Los sistemas gamificados poseen varios objetivos principales y secundarios entre los que cabe destacar la implicación del usuario en el proceso gamificado y la fidelidad, cualidad por la que el usuario permanece en la actividad. A todo esto hay que sumar objetivos relativamente secundarios como son la proactividad. El proceso de implementación de la gamificación siempre se ha puesto en duda debido al uso de la táctica del "palo y la zanahoria". Esta estrategia se basa en que la zanahoria simboliza la motivación y el palo es el equivalente al refuerzo negativo o a un castigo. El uso del condicionamiento clásico aplicado al refuerzo negativo crea un dilema entre la educación y la modificación de la conducta y el comportamiento. A lo largo de los años, la gamificación se ha limitado al uso de diferentes mecánicas de sistemas de puntuación y de insignias. En pro de la evolución, se han incorporado nuevas mecánicas como recompensa, logro, auto-expresión, altruismo, competición y estatus. El aspecto competitivo de los juegos es un atributo que se achaca a algunas implementaciones de la gamificación.

El primer ejemplo de caso de éxito en la ludificación relacionada con la educación es *Classcraft* [Classcraft, 2021]. Este sistema se basa tanto en la tecnología y en los roles de los alumnos, como en juegos e historias narradas para reinventar el concepto del aula. La clave del éxito de las mecánicas de un juego se utiliza para hacer atractivo el aprendizaje, ya que cuando una persona cumple veintiún años tendrá una experiencia de en torno a 10.000 horas, en videojuegos. Por este motivo, se relaciona el material del curso con las experiencias que vive el alumno.

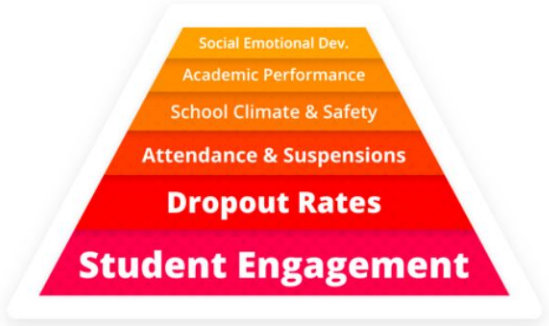

Figura 2.7: Gráfico del compromiso [Classcraft, 2021]

Por todo lo anterior, en la web de ClassCraft podemos ver que se define como un Sistema de Gestión de Compromiso (del inglés *Engagement Management System*, EMS), ya que gracias a las diferentes herramientas que se ponen a disposición de los docentes genera un alto impacto en el rendimiento académico, el comportamiento en el aula y la motivación del alumno entre otros. Logrando de este modo que el alumno obtenga unos buenos resultados educativos y su etapa educativa sea superada con éxito.

El segundo ejemplo de caso de éxito se trata de *Kahoot!*[Kahoot!, 2019], una herramienta para el refuerzo del aprendizaje basada en juegos. Su principal ventaja es fomentar la participación y el aprendizaje mediante el uso de cuestionarios. Estos cuestionarios se afrontan al alumno como un concurso, en el cual se puede participar en equipos o de forma individual. Generalmente, el cuestionario se basa en preguntas de tipo test. La puntuación de cada alumno o grupo de alumnos depende del número de preguntas acertadas y si están en racha, se aumenta esta puntuación.

La mecánica utilizada se basa en preguntas de tipo trivial que el docente presenta a sus alumnos en las que la respuesta puede ser de tipo:

- **Prueba:** pregunta con múltiples opciones.
- **Verdadero / falso:** preguntas simples para llamar la atención al alumno.

- **Rompecabezas:** prueba de compresión en la que hay que ordenar las respuestas en el orden correcto.
- **Prueba escrita:** prueba de retención de contenidos en la que se debe escribir una respuesta corta.
- **Diapositiva:** prueba de percepción en la que se deben identificar y señalar características relevantes al enunciado.

Además de estos casos de éxito para el refuerzo de aprendizaje en distintas áreas, existe soluciones gamificadas para aprender programación. Vamos a presentar brevemente algunas de ellas:

- **Code Combat**[2]**:** Se trata de un juego de rol (RPG[3]) HTML5 que enseña conceptos fundamentales de programación, como pueden ser variables, condicionales, bucles, etc. a los alumnos. Mediante el código que escribe el alumno en cualquiera de los lenguajes soportados (*Javascript*, *C++* y *Python*), que es interpretado por su correspondiente interprete, se produce una acción (o movimiento) del avatar por el mundo virtual en dos dimensiones.
- **CodinGame**[4]**:** Es una gamificación literal para el aprendizaje de la codificación. Permite al alumno elegir entre diferentes lenguajes (*Bash*, *C*, *C++*, *C#*, etc.). Se plantea un enunciado que si se cumple mediante el código, se representa una acción de forma correcta.
- **code.game**[5]**:** Sistema de aprendizaje basado en cursos en los que el alumno controla un avatar mediante la codificación en *Python* o un lenguaje de programación visual (*scratch*), muy útil para la enseñanza en niños.
- **Robocode**[6]**:** Entorno de simulación de guerra de tanques robot desarrollado por *Alphaworks* de IBM en que se aprende programando en *Java* y se realizan combates por equipos o individuales.
- **Screeps**[7]**:** Un juego tipo *sandbox*[8] que da libertad al jugador para completar tareas con el fin de alcanzar un objetivo. Es de carácter online, multijugador y masivo (en inglés, *Massively Multiplayer Online, MMO*), donde el objetivo principal es programar la IA de las unidades mediante *JavaScript*.

[2] https://codecombat.com/
[3] del inglés, *role-playing game*
[4] https://www.codingame.com/
[5] https://code.game/home
[6] https://robocode.sourceforge.io/
[7] https://screeps.com/
[8] Estilo de videojuego que se caracteriza por dar al jugador un alto grado libertad, tanto de movimiento como para ser creativo a la hora de completar tareas hacia un objetivo dentro del juego.

- **CheckIO**[9]**:** Plataforma interactiva que permite mejorar la habilidad de codificación de principiantes y avanzados mediante desafíos utilizando *Python* y *TypeScript*.

- **Flexbox Froggy**[10]**:** Una web donde el usuario puede desplazar un avatar, *Froggy*, mediante las instrucciones correctas en código CSS. El objetivo es situar el avatar sobre la posición correcta, marcada por una hoja de lirio verde.

- **Code monkey**[11]**:** Sistema de aprendizaje de conceptos de programación para niños en edad preescolar (de 4 a 6 años) con el que se desarrolla el pensamiento lógico sobre elementos como secuencias, algoritmos, bucles, etc., completando los diferentes niveles. En cada nivel se debe mover el avatar, un mono, salvando los diferentes obstáculos para alcanzar su objetivo, un plátano. Los alumnos deben arrastrar los bloques que representan acciones al área de codificación y el avatar los ejecutará.

La mayoría de los ejemplos anteriores además de contar con distintas actividades gamificadas, ofrecen a los docentes y centro formativos diferentes herramientas para realizar el seguimiento de los alumnos y analizar las dificultades que tengan con relación a la materia.

2.2. Plataforma de enseñanza LMS

Definición

El desarrollo de la enseñanza va ligado a las denominadas plataformas de enseñanza. En esta sección vamos a hablar en mayor profundidad sobre las plataformas LMS, un concepto que actualmente puede causar confusión. El término LMS es una tecnología basada en la web que ayuda a planificar, alertar y evaluar cualquier proceso de aprendizaje [Alshammari1 et al., 2018]. Según [Zah et al., 2010] es una aplicación de software de entorno diseñada para la gestión de las interacciones de los alumnos, así como para la entrega de recursos de aprendizaje a los usuarios/alumnos.

Estas plataformas son aquellas aplicaciones software o tecnologías basadas en web destinadas a planear, implementar y evaluar un proceso de aprendizaje específico en un entorno virtual. La plataforma de enseñanza se conoce como sistema de gestión del aprendizaje o LMS [Levinson, 2013]. Dicho objeto de estudio es un software cuya meta consiste en proporcionarlo y ponerlo a disposición, tanto del alumno como de los instructores. Se trata de un espacio virtual donde el instructor pueda compartir diferentes contenidos con el alumno, como pueden ser temarios de la asignatura en formatos PDF o Word. El alumno, a su vez, podrá exponer preguntas y dudas en un foro o

[9] https://checkio.org/
[10] https://flexboxfroggy.com/
[11] https://www.codemonkey.com/

a través de un mensaje directo y privado que le llega a otros usuarios, otros alumnos o el mismo instructor entre otras funciones. Por otro lado, también se pueden elaborar exámenes tipo test o pruebas de evaluación que el alumno tendrá que realizar en esta plataforma, donde el instructor tendrá la posibilidad de corregir las diferentes tareas y añadir los comentarios pertinentes para la mejora del aprendizaje del alumno.

Como consecuencia de lo mencionado con anterioridad, se mostrará al instructor el acceso a una sección privada donde solo pueda acceder él o ella y donde se registrarán, con los resultados y las notas de las evaluaciones, las métricas generales que se han obtenido a lo largo del curso virtual sobre cada alumno. En dichas métricas se tendrá en cuenta toda la actividad que ha desempeñado cada alumno a nivel individual en la plataforma a la hora de acceder a los diferentes recursos. Estas métricas pueden ser: la hora de acceso, por parte del usuario, al recurso, duración del visionado del recurso, actividad en el foro de la clase, etc.

Características

Una vez analizada la definición de una plataforma LMS, se van a estudiar los motivos por los que muchos usuarios se han decantado por el uso de este sistema y qué beneficios proporciona. Para ello, es importante averiguar qué características tiene y ver en profundidad cada una de ellas:

- **Facilidad de uso:** Esta plataforma destaca por tener un método de uso muy intuitivo, por lo que los usuarios que quieran acceder a la plataforma lo harán de una manera simple y sencilla. Como resultado, podrán hacer uso de la misma usuarios de diferentes niveles, sin importar los conocimientos que se hayan adquirido hasta el momento de su uso.

- **Comunicación bidireccional:** Como su nombre indica, la plataforma tiene una conversación bidireccional, es decir, habiendo solo un canal los mensajes pueden ir del profesor al alumno y viceversa, ya que son los actores principales.

- **Disponibilidad de acceso:** En este punto toma importancia la educación telemática, en otras palabras, aquella que se lleva a cabo por medios electrónicos. Aquí se permite la deslocalización de la relación, anteriormente cita, profesor-alumno, que permitirá el acceso a este último, sin restricciones temporales, a los recursos que precise.

- **Garantía de inclusión:** No se tendrán en cuenta los límites de capacidad o acceso, se podrá hacer uso de una plataforma LMS y todos los contenidos del curso, así como el sistema en sí mismo. Para ello, se debe facilitar el *e-learning* a cualquier usuario sin importar su condición o capacidad.

Las plataformas LMS proporcionan funciones interactivas con las cuales los alumnos pueden participar y sociabilizar en videoconferencias y foros de debate. Para la normalización de los sistemas de gestión de enseñanza, se ha definido un conjunto de especificaciones para poder compartir contenido denominado SCORM (*Shareable Content Object Reference Model*, en castellano Modelo Referenciado de Objetos de Contenido Compatible) con el que reutilizar contenido fácilmente gracias al estándar definido.

2.3. Entornos Virtuales de Aprendizaje y Mundos Virtuales

Definición de Entorno Virtual de Aprendizaje

Un entorno virtual de aprendizaje (EVA) [Poveda Criado et al., 2014] es un espacio de acceso restringido, implementado en una aplicación software, especialmente diseñado para el desarrollo de habilidades, conocimientos o capacidades.

El principal objetivo de un EVA es facilitar la comunicación como medio en el proceso de la actividad pedagógica. Esto permite que el proceso educativo pueda ser presencial, a distancia o una mezcla entre las dos anteriores llamada semi-presencial. Este entorno hace posible que al alumno acceda y desarrolle las mismas acciones que se podrían realizar en la enseñanza presencial (leer documentos, realizar actividades, comunicarse con el resto de alumnos de la clase, plantear preguntas al docente, resolver exámenes, etc.) sin necesidad de una interacción física entre los diferentes actores que intervienen en el proceso educativo.

Además de lo anteriormente enumerado, un EVA permite al alumno adquirir una mayor autonomía frente a la enseñanza presencial, ya que la forma en la que los contenidos se presentan disponibles en el aula virtual, incentiva la autogestión de los tiempos de trabajo así como la planificación de los tiempos de estudio.

Características

Las características de un sistema EVA pueden ser de varios tipos. Para [Boneu, 2007] hay cuatro características básicas, e imprescindibles, que cualquier plataforma *e-learning* debería tener:

- **Flexibilidad:** la docencia es una tarea en la que se necesita elasticidad para adaptar los diferentes métodos docentes, estructuras institucionales, contenidos y planes de estudios.
- **Escalabilidad:** capacidad de la plataforma de *e-learning* de no variar su funcionamiento con un número pequeño o grande de usuarios.

- **Estandarización:** es clave para poder crear diferentes cursos y usar diferentes plataformas, permitiendo importar y exportar contenido mediante el formato estándar SCORM[12].

- **Interactividad:** se fomenta la participación del alumno transmitiendo la sensación de protagonismo.

Además de estas cuatro características básicas expuestas anteriormente, [Robles and Ángel Gallardo Vigil, 2013] definen una serie de características igualmente relevantes que todo EVA debe cumplir:

- **Usabilidad:** los conocimientos necesarios para el uso de la plataforma tienen que ser mínimos (se dice que debe ser intuitivo). Esto reduce el tiempo de aprendizaje sobre el uso de la plataforma y centra al alumno en usar la herramienta para cumplir el objetivo principal de la misma, el aprendizaje.

- **Multiplataforma:** estos entornos deben ser capaces de operar en diferentes plataformas, tanto software como hardware, permitiendo así que su usabilidad y funcionalidad no se resuma a un solo programa o dispositivo concreto.

- **Fiabilidad:** el acceso a la plataforma se realiza mediante sistemas legítimos autenticando al usuario restringiendo el acceso a usuarios ajenos.

- **Transparencia:** el tratamiento de la información recogida y procesada por la plataforma es conocida por los actores, alumno y docente, al realizar el uso de la plataforma.

- **Adaptabilidad:** la interfaz gráfica del entorno virtual de aprendizaje debe ser capaz de ajustarse correctamente a los diferentes dispositivos desde los que se acceda a la misma.

Diseño EVA

Para diseñar un EVA con el claro objetivo de mejorar el aprendizaje existen diferentes factores, de acuerdo con [Quiroz and Salvat, 2011] hay que tener varios aspectos en cuenta a la hora de diseñar un EVA a la hora de destacar y mejorar en las prácticas educativas.

Primero, hay que desarrollar y llevar a cabo un plan de formación del profesorado completo que abarque enseñanzas ligadas a la construcción social del conocimiento y el avance de las diferentes capacidades informáticas relacionadas con este ámbito y con el papel que desempeña el profesor/a como creador de contenidos de aprendizaje virtual. El apoyo institucional es fundamental para

[12]SCORM (del inglés, *Sharable Content Object Reference Model*) representa una forma "de cómo un Sistema de Gestión de Aprendizaje (LMS) de aprendizaje (LMS) puede ofrecer a los alumnos contenidos de aprendizaje basados en la web de una manera estándar" [Baeini, 2004]

contar con todos los recursos tecnológicos y humanos necesarios para llevar a cabo la experiencia formativa virtual.

Adicionalmente, a todo lo anterior siempre es bueno que los docentes que realicen tareas de diseño, administración e implementación del EVA puedan contar con la ayuda, la experiencia y el apoyo de un equipo versado en el campo técnico, gráfico y pedagógico.

Por otro lado, un aspecto fundamental es la creación de comunidades de práctica, ya que en estas los docentes pueden conversar sobre su experiencia, recibir un consejo o proporcionar un consejo, preguntar y resolver dudas sobre un EVA, y a su vez, van a adquirir ayuda relacionada con el ámbito educativo y técnico.

Finalmente, es esencial que el estudiante se identifique como una figura fundamental y activa en esta experiencia educativa, por ello, establecer propuestas que cumplan con este requisito son fundamentales para alcanzar el objetivo final, un correcto aprendizaje.

Definición de Mundo virtual

Un **mundo virtual** se define como un entorno simulado con una representación en dos o tres dimensiones. Esta representación virtual se basa en el mundo real y crea una copia virtual. El usuario accede al mundo virtual mediante un dispositivo tecnológico con capacidad de cómputo suficiente como puede ser un ordenador, *tablet* o *smartphone*.

Mundo virtual al igual que realidad virtual, puede tener un significado bastante completo y a la vez complejo, pues no se trata solo de una creación irreal o ficticia en 3D para el participante. Desde el punto de vista informático, se trata de una simulación que puede ser alterada según las decisiones del participante en tiempo real haciendo uso de una interfase de usuario implícita, siendo todo esto posible gracias a los avances de última generación. Este concepto empieza a hacerse oír en 1968 gracias a los primeros trabajos realizados por pate de Ivan Sutherland. Sutherland fue capaz de crear un "sistema de presentación instalado en un casco ... para entrenamiento de pilotos y astronautas"[García, 1997].

Características del mundo virtual

Las principales características de un mundo virtual son:

- **Simulación visual:** La representación del entorno se realiza en dos o tres dimensiones y con un grado de detalle suficiente para permitir la inmersión del usuario.

- **Interacción:** El usuario debe poder realizar acciones sobre el mundo y explorarlo mediante un avatar.
- **Multiusuario:** La relación real entre personas se ve reflejada en el mundo virtual mediante la posibilidad de interacción entre avatares.
- **Comunicación:** Una característica clave es la comunicación entre los distintos usuarios mediante chat de texto o de voz, así como la comunicación entre el mundo y el usuario por medio del avatar.
- **Desarrollo de la habilidad estratégica:** el desarrollo de cualidades como la planificación y resolución de problemas son clave en el desarrollo de la independencia del alumno.
- **Role-playing:** mediante diversas actividades se simula situaciones del mundo real en el entorno virtual.
- **Perspectivas múltiples:** se presentan diferentes puntos de vista
- **Negociación y colaboración social:** mediante la interacción entre los usuarios se pueden preguntar dudas y resolverlas.
- **Inmersión:** el usuario vive una experiencia en una realidad alternativa ficticia que puede ser controlada en tiempo real.

2.4. Tecnología actual

El aprendizaje es un aspecto que está presente en nuestro día a día y evoluciona al mismo que lo hace el ser humano. A día de hoy, la tecnología nos ha brindado infinidad de posibilidades, y entre las numerosas ventajas que nos ha aportado también se encuentra el aprendizaje. Se puede afirmar dicho hecho por las grandes comunidades de jugadores (o *gamers*), como *Gamers.Red* [13] o *Retro Gamer* [14], que modernizan y hacen gran cantidad de usos del mundo de los videojuegos, entre otros, en entornos virtuales que no son LMS que apuestan por la educación. A continuación se van a enumerar las tecnologías más relevantes para el desarrollo de estos entornos virtuales:

- **Godot** [15] - Este motor de juego se caracteriza porque con él se pueden crear realidades en 3D y en 2D gracias a un diseño innovador que cuenta con muchos nódulos, un sistema de escenas flexibles, un editor visual, creación de contenido vistoso y también se pueden personalizar las herramientas.

[13] https://www.gamers.red/
[14] https://comunidadretrogamer.com/
[15] https://godotengine.org/

- **Unreal Engine 4** [16] - Se considera la herramienta de creación 3D en tiempo real más avanzada que se utiliza no solo para juegos sino para películas y televisión. Destaca su capacidad de producción virtual, su interfaz de máquina humana, la realidad extendida XR y la implementación multiplataforma.

- **CryEngine** [17] - Destaca sobre todo por su sistema de esqueleto paramétrico que permite y ayuda a llevar a cabo una animación en 3D muy realista. Además, cuenta con un sistema avanzado de AI para personajes NPC (*Non Player Character*), es decir, personajes que no son controlados por el usuario de ese entorno virtual o juego. Estos personajes pueden desempeñar funciones auxiliares de apoyo al usuario con distintos roles como puede ser por ejemplo, un rol de asistente virtual que realice una visita guiada por el entorno virtual para enseñar al usuario como interactuar con el mismo. Adicionalmente, este motor gráfico proporciona una calidad realista.

- **Amazon Lumberyard** [18] - Este entorno virtual cuenta con herramientas que ayudan a la interacción en tiempo real (*Twitch Chatplay*), a la personalización de las retransmisiones mediante componentes HTML que se usan para crear y visualizar gráficos dinámicos en tiempo real (*Twitch Metastream*), a la compatibilidad con diferentes *mods* (modificación que añade nuevas posibilidades a un videojuego) y que permite la creación de juegos multijugador (*Twitch Joinin*).

- **Unity** [19] - Herramienta de diseño de mundos virtuales habitualmente utilizada para desarrolladores de videojuegos. Esta herramienta está basada en el motor gráfico Unity, que permite crear entornos virtuales 2D y 3D. Las funcionalidades que ofrece y calidad de gráficos está en la media de este tipo de motores gráficos.

- **GameMaker Studio** [20] - Fue creado por Mark Overmars en 1998, con el objetivo de ayudar aquellos que tienen nociones básicas de programación o conocimiento nulo. Actualmente, ha evolucionado hasta presentar un motor de juego fácil de usar e intuitivo a la hora de usar las herramientas (pincel o mosaicos automáticos) para realizar animaciones 2D y 3D, además se apoya en los cálculos basados en la física para darle realismo a la experiencia.

Análisis crítico

En esta sección vamos a analizar en mayor detalle las diferentes alternativas tecnológicas con la que podemos contar a la hora de afrontar un proyecto como este. A continuación se muestran las

[16] https://www.unrealengine.com
[17] https://www.cryengine.com
[18] https://aws.amazon.com/es/lumberyard
[19] https://unity.com/es
[20] https://www.yoyogames.com/es/gamemaker

diferentes tecnologías a comparar desde una visión general, características y técnicas (véase Tabla 2.1).

	Definición	Desarrollado por	Precio
Godot	Motor de juego en 2D y 3D multiplataforma	MIT	Gratis
Unreal Engine 4	Motor de juego disponible en el código de fuente	Epic Games	Gratis en versión básica
CryEngine	Motor de juego en 3D de gran realismo en ambientación	Crytek	Gratis en versión básica
Amazon Lumberyard	Motor de juego de plataforma cruzada	Amazon	Gratis
Unity	Motor de juego de plataforma cruzada	Unity Technologies	Gratis en versión básica
GameMaker Studio	Motor de juego creada para aprender a programar	StudioYoyoGames	Gratis en versión básica

Tabla 2.1: Comparativa de los aspectos generales.

Esta primera tabla muestra la definición de cada tecnología así como quién ha desarrollado la misma. Un aspecto muy reseñable a la hora de comenzar un proyecto de este calado es el precio de la tecnología. En este caso, existen opciones sin ningún coste y otras con un coste dividido en planes.

Otros aspectos importantes a la hora de decantarse por una tecnología u otra con las características que definen cada una de estas opciones y en concreto el tipo de gráficos que permite generar.

	Características	Gráficos
Godot	Libertad en uso de *script*, XR, creación en 2D y 3D	Muy buenos en 3D y 2D
Unreal Engine 4	Gran marco multijugador, VFX y simulación de partículas	Muy buenos
CryEngine	Relismo en efectos de clima y vegetación en 3D, y mejoras de definición	Excelentes
Amazon Lumberyard	Edición del sistema de juego en tiempo real	Muy buenos en 3D
Unity	Mejores en 2D, animación y creación de instantáneas	Buenos, en 2D
GameMaker Studio	NPC controlados por Inteligencia Artificial, y hace uso de la física	Muy buenos en 3D y 2D

Tabla 2.2: Comparativa de las características tecnologías.

En la Tabla 2.2 cabe destacar que la tecnología basada en CryEngine posee un realismo y calidad gráfica superior a sus adversarios. Amazon Lumberyard ofrece una característica de edición en tiempo real de la dinámica interna lo que ofrece una gran versatilidad. GameMaker posee unas cualidades equilibradas a la par que ofrece nuevas posibilidades en el área de NPCs y la inteligencia artificial así como la implementación de las mecánicas de físicas.

	Plataforma soportadas
Godot	Multiplataforma (PC, Android,iOS, videoconsolas y web)
Unreal Engine 4	PC, Android,iOS, videoconsolas y web
CryEngine	Videoconsolas
Amazon Lumberyard	Creación de juegos para PC, consolas y móbiles
Unity	Multiplataforma (Windows, Mac, Linux, Android, iOS, videoconsolas y web)
GameMaker Studio	Multiplataforma (Windows, Mac, Linux, Android, iOS, videoconsolas y web)

Tabla 2.3: Comparativa compatibilidad de las plataformas tecnologías.

Hoy en día, la mayoría de usuario posee diferentes dispositivos con distintas plataformas con las cuales tienen la necesidad de poder acceder a recursos software. Por este motivo las tecnologías

disponibles para la realización de este proyecto deben ser capaces de proporcionar un correcto soporte en todas las plataformas más comunes. Las distintas plataformas existentes se pueden categorizar en tres grandes grupos: teléfonos móviles (incluye *tablets*), ordenadores personales, videoconsolas y web.

En la Tabla 2.3 la mayoría de las tecnologías que se están analizando tienen un soporte multiplataforma, lo que permite poder generar el proyecto en múltiples soporte o plataformas y así poder alcanzar el mayor número de usuarios posibles. En este aspecto, tecnologías como CryEngine y Amazon Lumberyard se encuentran más limitadas al no poder englobar un soporte multiplataforma como si implementan el resto de tecnologías.

	Curva de aprendizaje	**Lenguaje de programación**	**Accesibilidad código fuente**
Godot	Fácil	C++, C# y GDScript	Código abierto
Unreal Engine 4	Difícil	C#	Código abierto
CryEngine	Complejo	C++, Lua y C#	Código propietario
Amazon Lumberyard	Fácil	C++ y Lua	Código abierto
Unity	Fácil e intuitivo	C#	Código propietario
GameMaker Studio	Fácil e intuitivo	C++ y C#	Código propietario

Tabla 2.4: Comparativa de los aspectos tecnológicos.

Antes de decantarnos por una tecnología software el lenguaje de programación usado para su desarrollo es clave, ya que esto nos limitará en cierto modo la velocidad de ejecución o capacidad de operación. Junto con el lenguaje va ligado la curva de aprendizaje de la tecnología, factor clave para determinar el coste en tiempo y dinero para crear un proyecto. La comunidad de desarrolladores que usa una determinada tecnología es importante a la hora de poder resolver diferentes problemas que puedan surgir durante el desarrollo y que al ser esta tecnología de código abierto poseerá una comunidad de usuario mayor y con más conocimientos que una tecnología propietaria que será generalmente más cerrada para resolver este tipo de cuestiones.

En este caso, vemos que en la tabla 2.4 predomina el lenguaje tipo C, concretamente su versión C++ que es un lenguaje compilado que admite paradigma de programación orientado a objetos. Al mismo muchas de estas tecnologías son compatibles con el lenguaje C#, un lenguaje creado por Microsoft basado en la familia de lenguajes C basado en el paradigma de objetos y con seguridad de Tipos.

Teniendo todo lo anterior en cuenta, a lo largo del siguiente capítulo se tomará la decisión de la tecnología a usar de acuerdo al ámbito del proyecto, los objetivos, restricciones y supuestos definidos. Estos aspectos son clave para tomar la decisión en un proyecto de este tamaño y complejidad. Tras realizar este análisis comparativo, se podrá valorar con mayor conocimiento cada tecnología para proceder a seleccionar la más adecuada.

Capítulo 3

Planteamiento del problema, restricciones y supuestos

Partiendo de la base teórica expuesta en capítulos anteriores, se van a enumerar los objetivos que debe alcanzar el proyecto, así como las restricciones y supuestos del mismo.

3.1. Ámbito de aplicación del trabajo

El trabajo se va a desarrollar en el contexto de mejorar el aprendizaje de los alumnos, principalmente en asignaturas técnicas. En este ámbito, las dificultades del aprendizaje aumentan mediante el sistema de clase teórica y pequeños laboratorios de práctica. De todas maneras, esta metodología no excluye las posibles explicaciones teóricas que puedan darse en los laboratorios.

Algunos de los conceptos abordados en las asignaturas técnicas se exponen a menudo de forma limitada. Los objetos de aprendizaje asociados al aprendizaje de algoritmos suelen incluir en gráficos estáticos en los cuales solo se muestran el estado inicial y el estado final, aunque en algunas ocasiones se añade algún estado intermedio. No obstante, esto puede llegar a ser insuficiente para que el proceso de aprendizaje de un alumno finalice satisfactoriamente.

El desarrollo presentado en este proyecto se va a basar en la utilización de recursos tecnológicos relacionados con entornos virtuales que faciliten el aprendizaje de los conceptos mencionados con anterioridad.

3.2. Objetivos del trabajo

El objetivo principal del proyecto consiste en diseñar e implementar una herramienta basada en un entorno virtual que facilite y mejore el aprendizaje del alumno en conceptos de tipo procedimental

tales como algoritmos. Partiendo de este primer objetivo se han extraído los siguientes objetivos específicos:

- **Objetivo 1:** Diseñar y desarrollar los entornos virtuales que simulen el entorno real de la Escuela Politécnica.
- **Objetivo 2:** Creación del curso mediante una presentación basada en diapositivas con contenido procedente de la asignatura de Sistemas Operativos impartida en la Escuela Politécnica.
- **Objetivo 3:** Para la interacción del usuario con el entorno virtual, se deberá crear un avatar que pueda ser controlado por el usuario.
- **Objetivo 4:** La actividad docente se basa en la adquisición por parte del alumno de conocimientos de forma práctica mediante actividades basadas en la parte teórica del curso.
- **Objetivo 5:** Desarrollo de los diferentes códigos que proporcionen la operatividad del entorno virtual que permitan cumplir el resto de objetivos.

Este diseño se enfocará en la creación de un mundo virtual en 3D interactivo en el cual el alumno va a estar representado mediante un avatar con el que pueda desenvolverse dentro de dicho entorno. Para el desarrollo del mundo virtual se utilizará la plataforma Unity.

En consecuencia, es necesario realizar un aprendizaje del método de desarrollo en Unity, así como implementar *scripts* en *C#*. Una vez adquiridos los conocimientos esenciales se tendrán todas las competencias para poder crear el mundo virtual con Unity.

3.3. Restricciones del trabajo

Una vez se ha definido el ámbito de aplicación y los objetivos relacionados con el proyecto de estudio a continuación se exponen las restricciones a las que está sujeta dicha investigación y que establecerán los límites del proyecto y, a partir de ellas, posibles líneas de trabajo que pueda tener en un futuro.

1. No se considera parte del trabajo la actividad de monitorización con la participación activa de los alumnos.
2. El diseño únicamente se centrará en la mejora del aprendizaje sin el análisis de datos mediante ontologías o un sistema basado en SCORM

3. El lenguaje de la interfaz estará limitado únicamente al lenguaje español sin posibilidad de seleccionar otra alternativa.

4. Como parte del trabajo no se contempla la posibilidad de integración del proyecto en un LMS como *Blackboard*[1].

5. La representación del entorno virtual será lo más realista que permitan los medios disponibles para su desarrollo.

6. El contenido didáctico se limitará a una única materia de aprendizaje específica de carácter técnico.

3.4. Supuestos del trabajo

Teniendo en cuenta cada una de las restricciones mencionadas con anterioridad, se plantea una serie de supuestos asumidos en este tipo de trabajo.

- Uso de software y bibliotecas *Open Source*, siempre que sea posible.
- Implementación de una herramienta multiplataforma, que sea fácil de adaptar a diferentes sistemas operativos o plataformas.
- La herramienta será accedida únicamente por los usuarios desde ordenadores con sistema operativo Windows[2], Mac OS[3] o tipo Linux[4]
- Se supone el uso de la herramienta de mejora de aprendizaje en apoyo al material didáctico habilitado por los profesores que imparten la asignatura de Sistemas Operativos.

[1] https://www.blackboard.com/es-es
[2] https://www.microsoft.com/es-es/windows
[3] https://www.apple.com/es/macos/big-sur/
[4] Definición sistema operativo con *kernel* tipo Linux: https://www.redhat.com/es/topics/linux/what-is-linux

Capítulo 4

Solución adoptada

Partiendo del planteamiento del capítulo anterior, a continuación se va a exponer el desarrollo de la solución propuesta al problema. En concreto, se presentará la arquitectura de la herramienta, tecnología utilizada, método de aplicación de Unity así como el desarrollo de la misma que engloba el diseño de la aplicación, implementación pruebas y análisis de los resultados obtenidos.

4.1. Arquitectura

La arquitectura de un proyecto define de forma abstracta los distintos componentes que constituyen un sistema software, con el fin de desempeñar una o más tareas. Estas tareas pueden ser por ejemplo, mostrar el temario del curso, en la cual no se define la forma de implementación ni los componentes que serán necesarios.

La arquitectura interna de Unity (véase figura 4.1) se basa en patrón MVC modificado, aunque en su base implementa este patrón. Los principales componentes de Unity son: el inspector, donde se permite visualizar los detalles de todos los objetos, el editor, permite el diseño del entorno gestionando los diferentes objetos y el *Game Output*, el encargado de generar el mundo virtual a base de la definición del editor y los elementos contenidos en el inspector.

De igual modo, la arquitectura determina como son las interfaces y las comunicaciones entra cada componente del sistema. Esta arquitectura debe ser técnicamente posible, es decir, implementable en algún entorno de desarrollo con un lenguaje de programación determinado.

Partiendo del punto anterior, planteamiento del problema, restricciones y supuestos, se establecen los diferentes elementos que componen el sistema y las relaciones entre ellos. Cada módulo o componente de la arquitectura puede o no relacionarse con otros módulos.

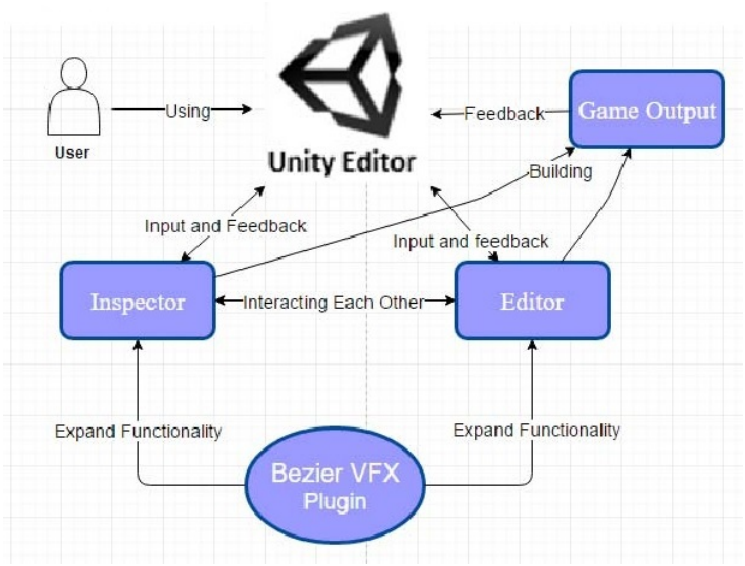

Figura 4.1: Arquitectura interna de Unity [Desmond and Bahana, 2016].

Para este proyecto no se ha implementado un patrón concreto a la hora de definir la arquitectura global del sistema. La mayor parte del desarrollo se engloba en la arquitectura proporcionada por Unity y su entorno de trabajo

Estos tres elementos que conforman la estructura base del sistema se definen de la siguiente manera:

- **Mundo virtual:** En los objetivos del proyecto queda reflejada la necesidad de crear un mundo virtual donde el alumno pueda mejorar su aprendizaje. Este componente de la arquitectura debe ser capaz de relacionarse con los otros dos componentes y poseer una interfaz de comunicación para que el usuario pueda acceder fácilmente.

- **Presentación:** Componente que debe mostrar una presentación basada en diapositivas donde se muestren los contenidos del curso a exponer.

- **Actividad:** El elemento actividad desarrollará en el alumno la capacidad de practicar el conocimiento teórico de la presentación.

- **Controlador:** Todos los elementos anteriores necesitan recibir órdenes de un componente líder. Este componente atenderá las órdenes del alumno y ejecutará las órdenes pertinentes con cada componente de la arquitectura para que el alumno cumpla los objetivos del sistema, la mejora del aprendizaje.

La funcionalidad del sistema recae en las capacidades de cada componente de la arquitectura, en las interacciones entre ellos y la gestión de los recursos como un sistema.

Atendiendo a la tecnología seleccionada para desarrollar el proyecto, vamos a definir las características de una arquitectura óptima [Cuenca and UP, 2020] que se va a aplicar en el desarrollo bajo el entorno de Unity.

- **Modular:** Crear sistemas que funcionen de manera independiente sin hacer referencias a otros sistemas, minimizando la transmisión de información entre los diferentes componentes. La gestión de eventos para la comunicación entres *scripts* es clave para un correcto funcionamiento de los diferentes módulos.

- **Escalable:** Separar los distintos componentes del modelizado en diferentes capas, separando controladores, comportamiento, etc. en diferentes capas y poder adaptar la profundidad de la arquitectura al proyecto que estamos creando.

- **Configurable:** Configuración de aspectos del proyecto sin programación. Modificar los datos de los sistemas en tiempo de ejecución para realizar un correcto ajuste.

- **Depurable:** Es necesarios crear módulos para comprobar de forma unitaria. El establecimiento de un sistema de obtención de excepciones. Realizar ejecución sin simulación.

Todas estas características que definen una arquitectura óptima se basan en los conceptos de la arquitectura interna de *Unity*. La arquitectura interna de *Unity* se fundamenta en *Monobehaviour*, la estructura de nuestro proyecto se adaptará a la estructura interna de *Unity*.

Monobehaviour es una clase serializable que puede ser visualizada en el inspector, siendo esta el padre de todos los *scripts* que se añadan como componentes que permite dar acceso a los principales métodos del ciclo de vida de Unity (*update*, *await*, etc.).

Los *scripts* en Unity, parte fundamental del desarrollo, siguen unos pasos de ejecución determinados. En la figura 4.2 podemos observar el orden de ejecución y repetición de los eventos de Unity. En [Unity, 2016] podemos obtener información más detallada y un diagrama completo en el que vemos que cada evento y método de Unity se ejecuta en un orden determinado lo que nos indica por ejemplo, que antes de ejecutar la función update Unity ejecuta una lectura de eventos del ratón. Esto es fundamental para conocer cuando se va a ejecutar los *script* que el desarrollador implemente como componentes durante el desarrollo del proyecto.

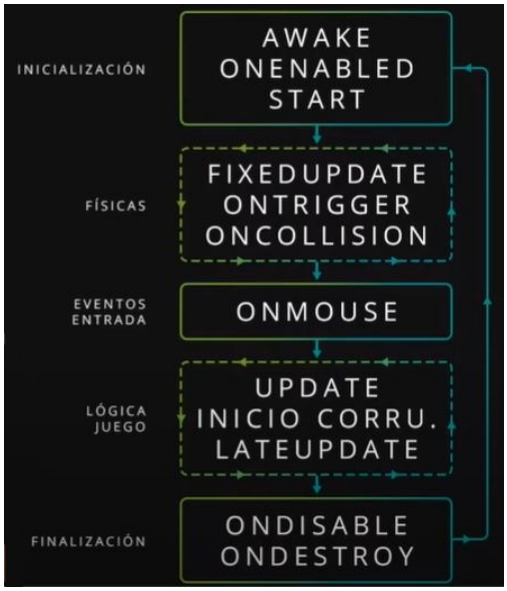

Figura 4.2: Ciclo de vida de un *script* en Unity. [Cuenca and UP, 2020]

4.2. Tecnología

El proyecto se va a desarrollar mayoritariamente usando la Tecnología de Unity que va a permitir crear todos los recursos y elementos necesarios para cumplir los objetivos establecidos en el planteamiento del problema.

¿Qué es Unity?

Unity[1] es una plataforma de desarrollo de contenido interactivo en tiempo real. La plataforma Unity permite desarrollar contenido 3D interactivo en tiempo real, proporcionando herramientas mediante un robusto ecosistema de diseño para la creación de juegos, aplicaciones y entornos virtuales. Gracias al ecosistema tan amplio con el que cuenta Unity y su capacidad multiplataforma, permite desarrollar contenido para una gran variedad de dispositivos como puede ser móvil (iOS y Android), web, PC (Windows, Linux, Mac) o videoconsola. Esto ofrece una gran versatilidad que le convierte en el entorno líder en el mercado.

Características de Unity

La plataforma Unity ofrece una serie de características que definen su comportamiento:

- **Herramientas de animación y cinemática:** Incluye herramientas de animación y para crear cinemáticas como 2D *Animation* o *Cinemachine*.

[1] https://unity.com/es

- **Audio y vídeo:** Componentes para mezclar audio y reproductor de vídeo.
- **Pila de tecnología basada en datos (DOTS):** Nuevo sistema de alto rendimiento DOTS (en inglés, *Multithreaded Data-Oriented Technology Stack*) permite mejorar la experiencia del usuario, la lectura y reutilización del código gracias a C#.
- **Editor:** El editor proporciona un gestor de paquetes, sistema para la gestión de los cálculos de físicas, procesamiento de elementos prefabricados (*prefabs*), además de la gestión del flujo de trabajo con los activos del proyecto.
- **Efectos:** Incluye sistema de simulación de partículas y fluidos.
- **Interfaz:** Herramientas para la interfaz gráfica, elementos gráficos de calidad (*TextMesh Pro*).
- **Iluminación:** Sistema para cálculo de iluminación mediante mapeo de rayos (*ray casting*) o iluminación global (GI).
- **Herramientas para programadores:** Utilidades para *scripting* en *C#*, API y control de versiones entre otras.
- **Optimización:** Herramientas de depuración como *Frame Debugger*.
- **Renderizado:** Utilidades para el renderizado como son *Shaders*, postprocesado o canal de renderizado de alta definición.
- **Construcción de mundos:** Diferentes complementos que permiten generar terrenos, como *ProBuilder* o *PolyBrush*.
- **2D:** Completa suite de herramientas para proyectos en dos dimensiones para la gestión de *Sprites* o física 2D.

Planes de Unity

Unity establece unos planes que proporcionan mayores o menores posibilidades en función del tipo de proyecto, es decir, un proyecto personal o estudiantil, si es un proyecto de una empresa pequeña o bien es un proyecto de gran escala.

Opcionalmente a los planes anteriores, existe un plan especialmente indicado para estudiantes, *Unity Student*[2], que cumplan los requisitos (inscritos en una institución educativa acreditada). Este plan cuenta con ventajas de los planes de pago similares a *Unity Pro*, permitiendo al estudiante hacer uso de flujos de trabajo y servicios profesionales para acelerar su enseñanza. Además de esto, permite la colaboración basada en la nube mediante la administración de versiones. Incluye el servicio *Cloud*

[2]https://unity.com/products/unity-student

	Personal	Plus	Pro	Empresa
Precio	Gratis			
Crear				
Plataforma básica de desarrollo en tiempo real	✓	✓	✓	✓
Herramienta Bolt para *scripting* visual	✓	✓	✓	✓
Personalización de la pantalla de inicio	✗	✓	✓	✓
Integración con las herramientas de colaboración	✓(1, a elección)	✓	✓	✓
Unity Team Advanced (3 puestos)	+	+	✓	✓
Paquete de assets de arte de alta gama	✗	✗	✓	✓
Capacidad de licencias de Build Server	✗	✗	✓	✓
Acceso al código fuente	✗	✗	+	+
Conjuntos de herramientas específicas para la industria	✗	✗	✗	+
Opera				
Cloud Diagnostics Advanced	✗	✓	✓	✓
Análisis básico	✗	✓	✓	✓
Monetizar				
Unity Ads	✓	✓	✓	✓
Complemento de compras dentro de la aplicación	✓	✓	✓	✓
Asistencia y aprendizaje				
Soporte técnico	✗	✗	+	✓
Administrador del éxito del cliente	✗	✗	✗	+
Fila con prioridad para servicio al cliente	✗	✗	✓	✓
Servicios para el éxito integrados	✗	✗	+	+

Tabla 4.1: Comparativa de planes económicos de *Unity*.

Diagnostics Advanced lo que permite rastrear fácilmente el origen de los errores. Todo esto está destinados a miembros de la comunidad educativa de forma gratuita que cuenten con un email con subdominio edu.

Para el desarrollo del proyecto se ha valorado, según la tabla 4.1, usar el plan Personal, ya que por las dimensiones del trabajo no es necesario adquirir ningún plan superior.

Los conceptos básicos de *Unity* son los siguientes:

- **Assets**
- **Interfaz de usuario**
- **Escenas**
- **GameObjects**
- **Componentes**

- **Prefabs**
- **Scripts**

Assets

Un *asset* es una representación de cualquier elemento que se puede utilizar en un proyecto de *Unity*. Estos elementos pueden ser de tipo audio, vídeo, imagen, modelo 3D, animaciones FBX, incluso *assets* creados dentro del propio *Unity*, como puede ser un *Animator Controller*.

Existen objetos 3D que pueden ser creados por un software de modelado (como *Blender*) y además hay unos tipos primitivos incorporados en *Unity* que son:

- **Cubo:** Es un objeto simple con la dimensión de una unidad de lado, que una vez escalado es muy útil para crear paredes u objetos similares.
- **Esfera:** Representa una esfera de diámetro una unidad. Utilizados para representar un proyectil o pelota.
- **Cápsula:** Elemento cilíndrico con dos tapas semiesféricas con una unidad de diámetro y dos unidades de altura. Generalmente esta forma se usa como *collider*.
- **Cilindro:** Unidad cilíndrica de una unidad de diámetro y dos unidades de altura. Normalmente se usa para generar postes o llantas.
- **Plano:** Se trata de un cuadrado plano de 10 unidades por lado orientado en plano XZ. Muy usado para crear paredes o pisos.
- **Quad:** Semejante al plano, pero su superficie está orientada en el plano XY.

Generalmente, la mayoría de *assets* se importan de forma sencilla arrastrando el elemento en cuestión sobre la carpeta *asset* en la jerarquía de archivos del proyecto. Una vez importado cada elemento, se debe configurar los parámetros de importación, ya que estos afectan a la forma de visualización y de comportamiento del *asset*. Para ver estos ajustes, primero se debe seleccionar el *asset* en la jerarquía del proyecto y posteriormente, en el inspector, podrá ver los valores de importación (*import settings*) correspondientes al tipo de elemento.

Unity pone a disposición de los usuarios otras dos formas de importar *assets*, a través de la *asset store*[3] (tienda de assets) y *asset package* (gestor de paquetes). Estas herramientas permiten a los usuarios compartir y/o vender proyectos completos o conjuntos de *assets* prefabricados. Por ejemplo,

[3]https://assetstore.unity.com/

se puede importar *Standard Asset Packages*, un paquete con colecciones de *assets* prefabricados proporcionados por *Unity*, y *Custom Packages*, los cuales son creados por usuarios utilizando Unity.

Interfaz de usuario de Unity

La interfaz de usuario del editor de Unity se compone de diferentes vistas que aportan diferente información al usuario a la hora de desarrollar un proyecto. Esta interfaz que se muestra en la figura 4.3 se organiza en diferentes ventanas.

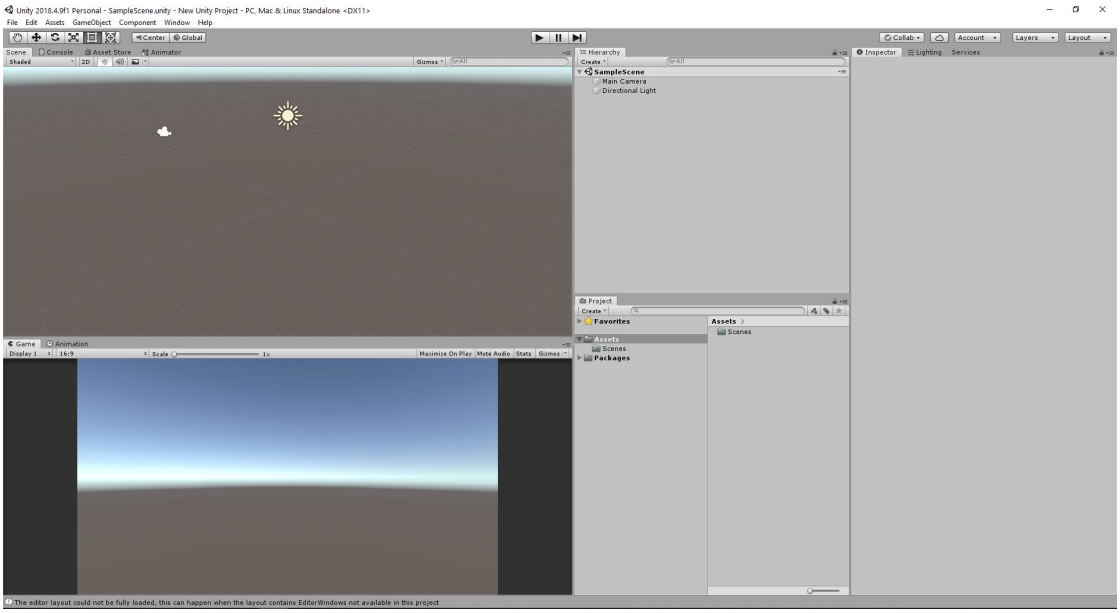

Figura 4.3: Interfaz de usuario de Unity.

A continuación se explica cada una de las ventanas que posee Unity en su interfaz por defecto:

- **Proyecto (*Project*):** En esta ventana se puede acceder y gestionar todos los *assets* del proyecto. Esta ventana se compone de dos paneles:
  - El panel izquierdo se muestra la jerarquía de ficheros del proyecto en la que se puede observar cada una de las carpetas y archivos que conforman el proyecto.
  - En la parte superior del panel derecho se encuentra una barra de búsqueda donde se podrá buscar un asset concreto.
  - Desde este panel también se pueden crear, mediante el botón *create* situado a la izquierda del buscador, nuevos *assets*, al igual que sub carpetas.
  - Ubicada en la parte superior del panel derecho se sitúa la ruta de navegación, que muestra la ruta actual de la carpeta la cual se esté mostrando su contenido en el panel derecho.

- **Vista de escena (*Scene view*):** En esta ventana podemos interactuar con los objetos de la escena a modo de editor en la que podremos seleccionar y posicionar cualquier *GameObject* que esté en la jerarquía de objetos. Las operaciones básicas, situadas en los botones de la parte superior izquierda de la interfaz de Unity, que podemos realizar son:

  - Herramienta mano: Esta herramienta se usa para la navegación dentro la escena y cambiar así el punto de vista de esta ventana. Esta herramienta puede realizar tres acciones que se describen a continuación:

    - Desplazar (*Move*): Con esta herramienta, hacemos clic con el botón izquierdo del ratón en la vista de escena y arrastramos para mover la cámara alrededor. Presionando la tecla *shift* se aumenta la velocidad de movimiento.
    - Orbitar (*Orbit*): Solo disponible en modo 3D, dejando presionado la tecla *Alt*, hacemos clic y arrastramos para orbitar la cámara alrededor del punto actual de pivote.
    - Acercarse (*Zoom*): Presionando la tecla *Alt*, hacemos clic y arrastre el ratón para acercar la escena o use la ruleta del ratón.

  - Mover (Move): Esta operación desplaza al objeto seleccionado por la escena según el eje en el que se haga clic y manteniendo presionado el indicador que se sitúa sobre el pivote del *GameObject*. Mediante la tecla de acceso rápido *W*, se activa esta función. También se pueden realizar los cambios en el inspector, con el componente *transform* del objeto que se desea desplazar. Para posicionar elementos de la interfaz de usuario (UI), del inglés *User Interface*, se utiliza la operación *RectTransform* que se activa con la tecla *T*.

  - Rotar (*Rotate*): Funcionalidad que permite rotar el objeto seleccionado mediante el *gizmo* que aparece sobre el mismo o bien, introduciendo los valores en el componente transform.

  - Escalar (*Scale*): Para aumentar el tamaño del objeto usamos esta función que se puede realizar desde el gizmo sobre el objeto o desde su componente *transform*.

- **Vista del juego (*Game View*):** Desde esta vista podemos observar la escena generada tal y como la podría ver el usuario desde el punto de vista de la cámara de la escena. Se pueden ajustar las siguientes opciones:

  - Visualización (*Display*): Esta opción permite visualizar en la vista las diferentes cámaras existentes en la escena.

  - Aspecto (*Aspect*): Con esta opción se puede configurar la relación de aspecto con la que se verá la escena.

  - Escalado(*Scale slider*): Unity permite realizar un escalado, es decir, ajustar el tamaño de la visualización de la escena mediante un control deslizable.

- Maximizar en ejecución (*Maximize on Play*): Permite ampliar la ventana hasta el 100 % del tamaño relativo a la ventana del editor, para poder observar mejor los detalles de la escena.
- Silenciar audio (*Mute audio*): En el modo ejecución, se silencia cualquier sonido procedente de la escena.
- Estadísticas (*Stats*): En modo ejecución al activar esta opción se superponen estadísticas de renderizado del audio y de los gráficos sobre la pantalla.
- Gizmos[4]: Este menú desplegable permite seleccionar qué *gizmos* se muestran en pantalla durante la ejecución.

- **Vista de la jerarquía (*Hierarchy*):** El listado de todos los *GameObject* existentes en la escena es una información muy importante en el desarrollo de Unity. En esta ventana se muestran todos estos elementos. Existe la posibilidad de situar un *GameObject* debajo de otro en la jerarquía, esto permite crear una relación de parentesco (*Parenting*).

- **Vista del inspector (*Inspector*):** El inspector es quizás la ventana más importante en *Unity*. En ella, se pueden ver todos los componentes de un *GameObject*, así como distintos atributos y valores que se pueden editar y cambiar. De igual modo se pueden ver los valores de *script*. Por último, y no menos importante, desde esta ventana se pueden consultar y modificar los valores de los ajustes de importación de los *assets* y los ajustes del proyecto.

- **Otras ventanas de interés:** En *Unity* existen una gran cantidad de ventanas que aportan información y funcionalidad durante el desarrollo. Entre estas ventanas cabe mencionar las siguientes:

  - La ventana de consola (*console*) muestra los mensajes, advertencias y errores relativos al proyecto, antes y durante la ejecución.
  - El visor de animación (*Animation View*), se utiliza para animar los objetos de la escena.
  - Para el manejo de las máquinas de estados se utiliza el visor de animador (*Animator view*) que permite crear y gestionar las transiciones entre los diferentes estados.
  - Existe una vista llamada *Profiler* que permite buscar y encontrar errores que afecten al rendimiento del sistema.
  - En el ajuste de la iluminación, Unity posee un visor del mapa de luz (*Lightmapping View*) que permite desempeñar esta tarea.

[4]Una superposición gráfica asociada a un *GameObject* en una escena y que se muestra en la vista de la escena. Puede situarse sobre el objeto y ser siempre visible o solo ser visible cuando se selecciona el objeto.[Technologies, 2021]

Escenas

Las escenas en Unity contienen los entornos del entorno virtual. En cada Escena, colocas tus entornos, obstáculos y decoraciones, esencialmente diseñando y construyendo tu juego en piezas [Technologies, 2021]. Una escena puede ser desde el menú principal de la aplicación, una estancia concreta del entorno virtual o una escena que contenga una interfaz gráfica para el usuario. Cada escena contiene diferentes tipos de elementos como son objetos 3D o 2D, decorados, diseño esencial o *scripts* que aporten funcionalidad.

La escena se almacena en el proyecto de Unity como un fichero en que se guarda los objetos que contiene la escena así como sus valores de posición.

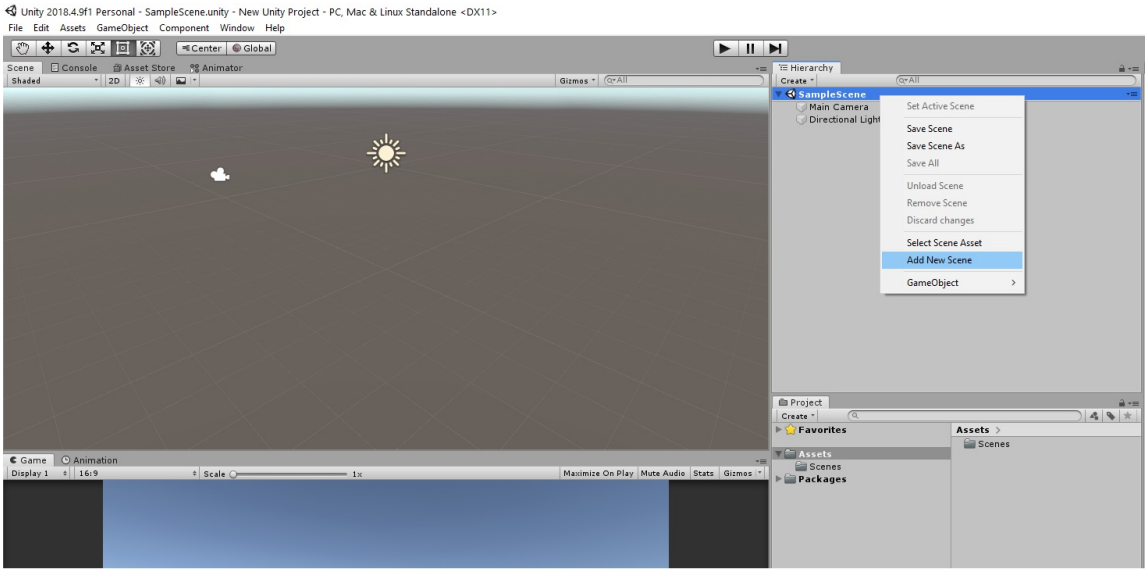

Figura 4.4: Escena vacía en Unity con los objetos 3D predeterminados.

En la figura 4.4 se puede observar como se representa una escena de Unity 3D por defecto. Los elementos por defecto son una cámara principal y una luz direccional.

GameObjects

Los *GameObjects* son parte fundamental de Unity, son todos los objetos que conforman el entorno virtual. En la figura 4.5 podemos observas unos ejemplos. Por si mismos, estos objetos no poseen ninguna funcionalidad o carácter diferenciador. Un *GemaObject* puede ser un avatar, un edificio, un vehículo o un simple cubo. Pero los *GameObjects* no son capaces de hacer nada si no se les añade los componentes necesarios. Por eso se dice que un *GameObject* actúa como un contenedor de componentes y estos implementan la verdadera funcionalidad. Por ejemplo, si creamos un objeto llamado *Punto de luz* es necesario añadir el componente luz (light) para que ese objeto puede emitir

rayos que simulen la luz emitida por una bombilla en el entorno virtual de la escena.

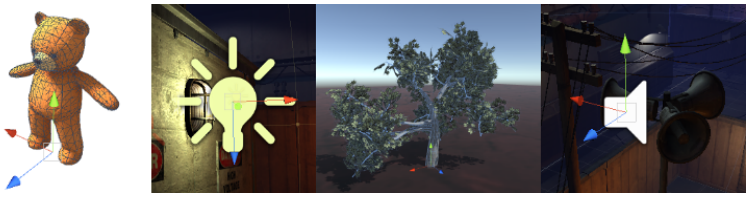

Figura 4.5: Ejemplos de *Game Objects*. [Technologies, 2016]

Un *GameObject* siempre tiene el componente *Transform* adjunto, que permite representar la posición y orientación de este objeto, y no es posible quitar este componente básico. En caso de un objeto de la interfaz gráfica (UI) este *Transform* se pasa a llamar *RectTransform* cumple la misma funcionalidad pero con más atributos.

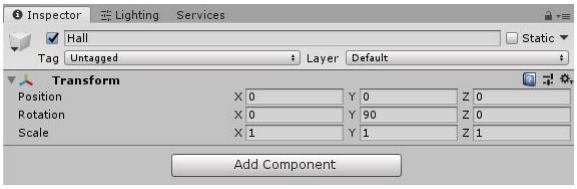

Figura 4.6: GameObject por defecto.

Si necesitamos agregar componentes para aportar funcionalidad a un objeto podemos usar la opción *Add Component* situada en el inspector como se puede observar en la figura 4.6.

Además podemos utilizar objetos pre-construidos como figuras primitivas (cubo, esfera, etc.) que son accesibles desde el menú *GameObject* o desde el menú *create* situado en la parte superior de la ventana de vista de la jerarquía.

Componentes

Los componentes (en inglés, *Components*) aportan funcionalidad a los *GameObjets* de forma que estos son contenedores de *components* que caracterizan al *GameObject* al que pertenecen.

Cada *GameObject* creado en *Unity* tiene un componente *Transform* que no se puede eliminar y que es la base de todo *GameObject*. El componente *Transform* es la base de todo en Unity, este componente aporta la funcionalidad básica de posición, rotación y escala de cada *GameObject*. Además de esto, *Transform* habilita el concepto de parentesco entre todos los *GameObject* ya que todos tiene ese componente básico.

Por ejemplo, un componente *Rigidbody* en un objeto vacío, que se puede crear mediante la opción *create* de la jerarquía del proyecto, la opción *Create Empty* del submenú (véase figura 4.7), provoca

que al ejecutar en Unity la simulación este se aleje de la vista de la cámara, ya que le afecta la gravedad y no hay ningún elemento situado por debajo de este que le pueda frenar.

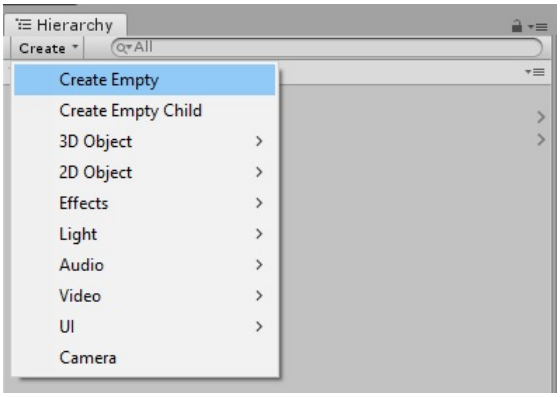

Figura 4.7: Crear *GameObject* vacío.

Desde el inspector se pueden observar los componentes del *GameObject* seleccionado y todas las propiedades de cada componente como se puede observar en la figura 4.8.

Cada componente posee diferentes valores y referencias que pueden ser editadas mientras se ejecuta el entorno virtual desde el inspector (al finalizar la ejecución no se almacenan los valores seleccionados) o mediante algún *script* accediendo a las distintas propiedades de cada componente. Para fijar unos valores por defecto al inicio de la ejecución, estos pueden ser fijados mediante un *script* o de manera estática desde el inspector.

En la parte superior de esta venta, dentro del panel *Inspector*, podemos ver y editar el nombre el objeto seleccionado. Junto a este campo se sitúa en su lado izquierdo un selector de tipo *checkbox* que en la figura actual indica que este objeto está activo y se genera en la escena con todos sus componentes y atributos. En la siguiente línea, se permite definir un *tag* para el objeto seleccionado y una capa (*layer*) que permite agrupar diferentes objetos.

En la parte inferior de esta ventana se sitúa el botón *Add Component* con la funcionalidad de añadir nuevos componentes o componentes existentes en la ventana *Project* al objeto seleccionado en el inspector.

Componente *Transform*

El componente clave de cualquier *GameObject* es el componente *Transform*, el cual fija el valor *position* que establece las coordenadas X, Y, Z. Además de la posición, los valores de *rotation* indican la orientación en torno a los ejes X, Y, Z medida en la unidad de grados. Por último, este componente se encarga de fijar la escala (*scale*) del objeto a lo largo de los ejes X, Y, Z. El valor 1

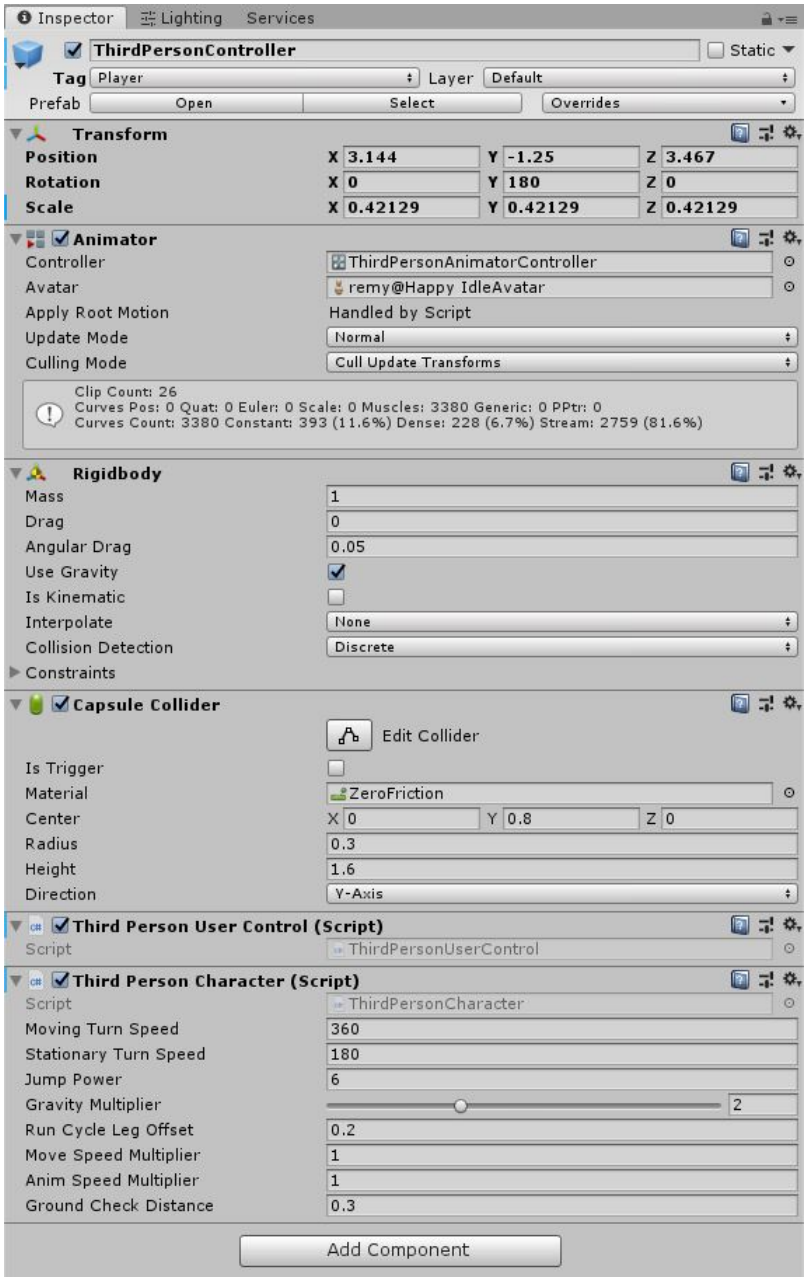

Figura 4.8: Componentes en *GameObject*.

representa el tamaño real, es decir, una unidad en el plano equivale una unidad en la realidad. Estos valores se obtienen tomando como referencia el espacio virtual si el componente *transform* no tiene un componente padre.

Componente Script

Como se ha mencionado anteriormente, los componentes son elementos que *Unity* nos permite integrar en los objetos que deseemos y que podremos ajustar basándonos en las opciones permitidas.

Cuando se crea un componente que se puede añadir a cualquier objeto, solo existe una opción; mediante *scripts*.

Unity permite que el usuario cree componentes y pueda ajustar las opciones que desee que sean accesibles desde el inspector. Además, estos componentes incluirán la funcionalidad necesaria para cada objeto.

Scripts

Uno de los componentes clave en el desarrollo de cualquier entorno virtual basado en la tecnología de *Unity* es el *script*. Las funciones que desempeña un *script* son muy variadas: responder a entradas del usuario (teclado, ratón o *gamepad*), procesar el comportamiento físico de objetos, gestionar que los eventos del entorno virtual se ejecuten en el momento adecuado, implementar mecanismos de presentación de datos, etc. Otras funcionalidades pueden ser la creación de efectos gráficos o un sistema de inteligencia artificial para los personajes no jugador (NPC, del inglés *Non Playable Character*), es decir, personajes avatar que el alumno no puede controlar de un entorno virtual.

Mediante los componentes (*Components*) se modela el comportamiento de los *GameObjects* usados en el desarrollo virtual dentro del entorno de *Unity*. Utilizando los componentes integrados de *Unity*, se pueden añadir características interesantes para el proyecto. Así mismo, *Unity* da la posibilidad de crear componentes propios mediante los *scripts*. Los *scripts* permiten activar o desactivar eventos del entorno virtual (*trigger*), detectar una orden del usuario mediante las entradas (*inputs*) que proporciona *Unity* y modificar el valor de las propiedades de un componente o de un objeto en tiempo de ejecución. Por ejemplo, un *script* puede mover un objeto en tiempo de ejecución o cambiar el valor de la altura al detectar una colisión.

Los *scripts* de *Unity* se pueden codificar en dos lenguajes, *C#* y *UnityScript*:

- **C#:** Este lenguaje es un referente en la industria similar a lenguajes como *Java* o *C++*.
- **UnityScript:** Lenguaje específico diseñado para su uso en *Unity* similar a *JavaScript*.

Los *assets* de tipo *script* se puede crear desde la propia interfaz de *Unity*, desde el menú situado en la ventana *project* haciendo clic en la opción *Create* desplegando el submenú y seleccionando nuevo *Script* o desde el menú *Asset*. De esta forma, el nuevo *script* se situará en la carpeta que esté seleccionada en el panel del proyecto. Es conveniente modificar el nombre del *script* en este momento para evitar posibles conflictos al modificarlo más adelante. También es interesante situar todos los

scripts bajo una misma capeta dentro de la estructura de carpetas del proyecto, para facilitar el desarrollo.

Para editar un *script*, se debe hacer doble clic sobre el mismo o desde el inspector seleccionar la opción *Edit script*. Al realizar esta acción, se abrirá una nueva ventana con el editor configurado en *Unity*. Para seleccionar qué editor se desea utilizar para editar *scripts*, se debe acceder al panel *External Tools* en las preferencias de *Unity* donde se elegirá entre los editores compatibles disponibles en nuestro equipo.

Un *script* funciona gracias a las conexiones que posee con el núcleo de *Unity*. Esta conexión se realiza mediante la herencia de una clase que deriva de la clase *MonoBehaviour*. Cuando se adjunta un *script* a un *GameObject*, este crea una nueva instancia de objeto en el plano. El nombre de la clase se obtiene del archivo del *script* cuando este fue creado. Este nombre debe ser el mismo para permitir que el *script* sea añadido al *GameObject*.

A continuación podemos observar la estructura básica de un archivo *script* codificado en C# de *Unity*.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class MainPlayer : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11    // Update is called once per frame
12    void Update () {
13
14    }
15 }
```

Código 4.1: Estructura básica script C#.

La estructura principal de un *script* básico se componen de dos funciones que son *Update* y *Start*.

- *Update* : Esta función ejecuta el código que haya en su interior por cada *frame*, es decir, por cada fotograma que se genera. Generalmente, en esta función se sitúa código relativo al movimiento de un objeto, acciones de *trigger* o responder a la entrada del usuario. Son acciones relativas al *Gameplay* del entorno virtual.

- *Start* : Como se puede observar en los comentarios del fragmento de código 4.1, esta función se ejecuta cuando sé instancia el objeto al que está asociado un componente *script*, únicamente una vez. En ella se inicializan valores de variables, se realizan lectura de preferencias y se llevan a cabo conexiones con otros *GameObject*. No es necesario definir un constructor para las clases, ya que es *Unity* el encargado de crear estas clases.

La implementación de *Mono* además incluye más funciones como pueden ser *Awake*, *FixedUpdate*, o para los eventos de cálculo de física de objetos *OnCollisionEnter*, *OnTriggerEnter*, etc. A todo esto hay que sumar que se pueden ejecutar funciones definidas por el desarrollador.

Esta implementación basada en *Mono* no impide que desde un *script* se pueda acceder al motor de *Unity* sin operar bajo el estándar que fija *Mono*.

Prefabs

En el desarrollo de un proyecto es muy probable que sea necesario volver a utilizar un objeto en múltiples ocasiones. Para esto, *Unity* posee diferentes mecanismos que permiten reutilizar objetos de tal forma que no sea necesario volver a crear uno a uno estos objetos y volver a añadir sus componentes o ajustar sus propiedades desde el inspector.

Los *Prefabs* son el mecanismo que proporciona *Unity* al desarrollador para poder reutilizar objetos. Este mecanismo es un tipo de *asset* que almacena un objeto de tipo *GameObject* con sus componentes y propiedades. Este *asset* actúa como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en cada escena o lugar necesario.

La mecánica de trabajo con un *prefab* es la siguiente:

1. Se crea el *GameObject* completo, con sus componentes y sus propiedades correctamente ajustadas.

2. El objeto se selecciona y se arrastra a una carpeta en la jerarquía del proyecto. Esta carpeta es común renombrarla con el nombre *Prefab* por conveniencia para tener todos los *assets* de este tipo ubicados en una misma capeta.

3. En ese momento ya se ha creado el *Prefab* y cada vez que se desee instancia en la escena únicamente hay que arrastrar el objeto a la escena y se creara una nueva instancia idéntica al objeto original.

Como se puede observar en la figura 4.9 el icono del objeto *Prefab* (en este caso Columna) cambia del color gris que representa los objetos activos al color azul que distingue a los objetos dependientes de un *Prefab*.

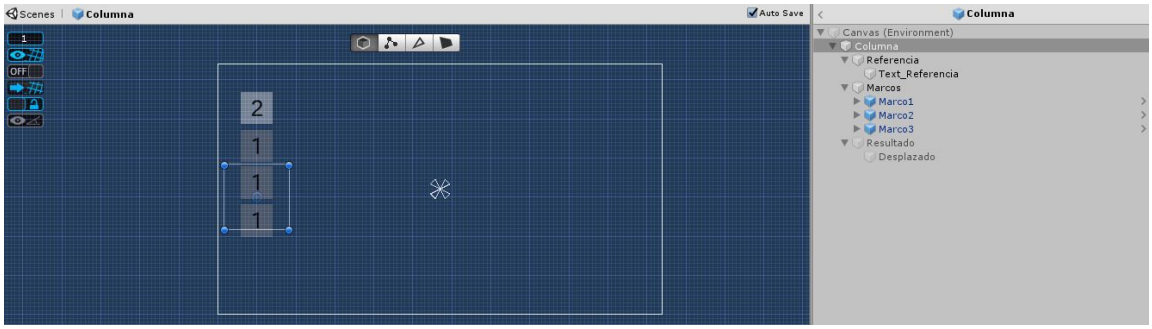

Figura 4.9: Ejemplo de un *asset* tipo *Prefab*.

Mediante este procedimiento, se pueden crear todas las instancias que necesitemos del objeto *prefab* en nuestro proyecto, todas ellas idénticas a la original. Si en algún momento se desea editar una instancia o todas, desde el inspector se puede acceder al *Prefab*. Se permite sobrescribir valores de tal modo que se puede modificar una instancia sin afectar al resto. En caso de querer editar todas las instancias el procedimiento sería desde una instancia, en la vista del inspector, hacer clic en *open prefab* y realizar los cambios que se guardaran en el objeto principal. Del mismo modo podemos modificar una instancia y aplicar todos o ciertos cambios al *Prefab* desde las opciones del inspector (situadas bajo la propiedad *Tag*). Otra opción para crear un *Prefab* consiste en acceder al menú *Asset* a la opción *Create Prefab* y arrastrar el objeto a la escena con el *Prefab* vacío.

Aplicación del SCORM en Unity

La integración de un LMS se basa en la interconexión entre el *SCORM* y el sistema *Unity*. Se debe exportar el proyecto *Unity* como *WebGL* para poder ser ejecutado por un LMS como Moodle[5].

Existen *assets* que permiten realizar una integración entre *Unity* y un LMS mediante *SCORM*. Un ejemplo concreto es el paquete *Unity-SCORM-Integration-Kit*[rstals, 2021] el cual implementa los mecanismos necesarios para comunicar nuestro proyecto en Unity con el SCORM del LMS donde se va a integrar el proyecto. Esta integración se basa en el acceso a las funcionalidades que incluye el *Scorm Manager* para cargar y guardar datos desde y hacia el LMS mediante el *SCORM*.

El *Scorm Manager* es el encargado de proveer una interfaz entre el *SCORM* del LMS y el proyecto de *Unity* mediante un archivo en *JavaScript* llamado *scorm.js* el cual establece la comunicación con

[5]https://forum.unity.com/threads/released-moodle-api.510201/

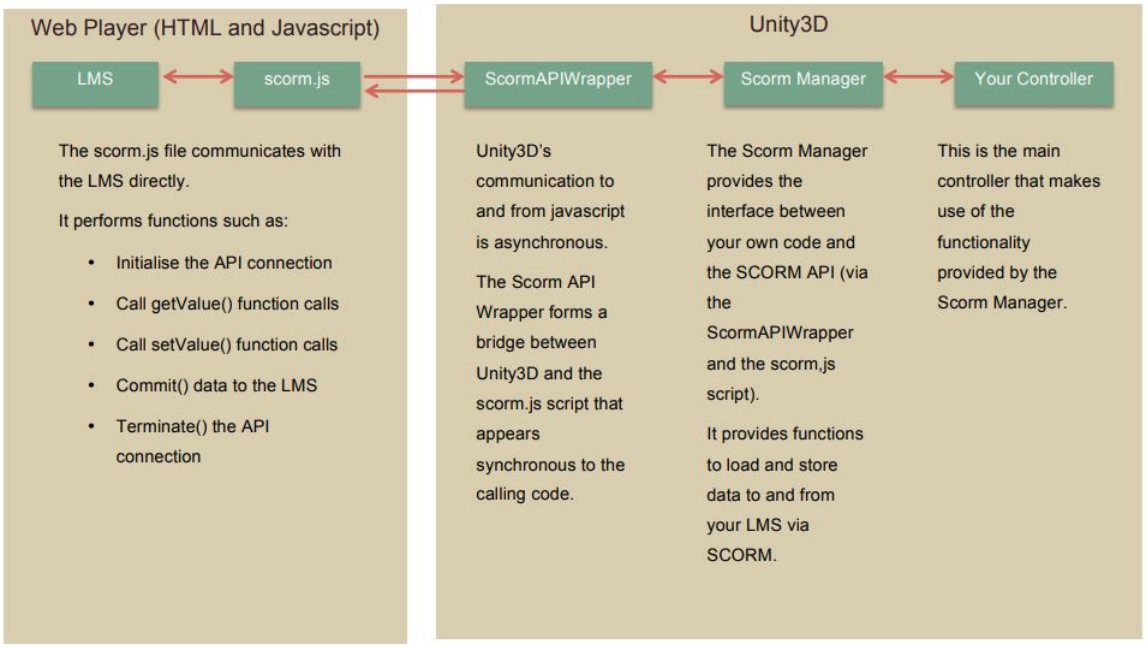

Figura 4.10: Arquitectura de Unity-SCORM-Integration-Kit [rstals, 2021]

el LMS, implementando la función de inicializar, obtener/almacenar valores al LMS y terminar la conexión.

Una restricción de este paquete es la *API SCORM*. Concretamente, para aplicar este proyecto en un LMS *Blackboard*[6], se debe usar como *API SCORM* el estándar *SCORM* 2004 o superior para que funcione con el paquete.

4.3. Metodología aplicada

Para la realización de un proyecto, si además este es de un extenso uso de recursos (personales, temporales y técnicos), es aconsejable la utilización de metodologías de desarrollo de software que nos permitan finalizar con éxito el trabajo. Así, una vez se empleen las metodologías de software, podremos obtener ciertos beneficios, entre los cuales destacan la creación de un producto de calidad, la reducción de riesgos propios del mismo desarrollo del software y un aumento de la productividad.

Hay que conocer en profundidad las metodologías que se encuentran a nuestra disposición, para elegir la adecuada a nuestro proyecto, teniendo presente que estas están divididas en dos grandes grupos, a saber:

[6]http://unity3d.stals.com.au/scorm-integration-kit/

- Metodologías tradicionales.
- Metodologías ágiles.

Metodologías tradicionales

En los inicios de la década de los años 60, se produjo una gran crisis, surgida de las dificultades de realización de grandes proyectos, momento en el que se acuñó el término de "ingeniería de software", que utilizaba las metodologías tradicionales.

Para definir la ingeniería de software, con sus características correspondientes, utilizamos varios términos, destacando entre ellos: La cuantificación es el proceso de convertir un objeto a un grupo de valores discretos, como por ejemplo un número entero. La disciplina, nos indica que todas las actividades están siendo supervisadas según un procedimiento preestablecido de estándares. La sistematicidad se refiere a la forma ordenada de como debe avanzar el proceso investigativo, conforme a un plan previamente concebido.

Destacamos de entre las metodologías tradicionales las siguientes [Pressman, 2010]:

- **Modelo de Desarrollo Rápido de Aplicaciones (RAD):** El Desarrollo Rápido de Aplicaciones. Este método abarca el desarrollo interactivo, la creación de prototipos y el empleo de utilidades CASE (*Computer Aided Software Engineering*). Además, la metodología RAD suele englobar también la usabilidad, utilidad y la rapidez de ejecución.

  Se compone de cinco fases, cuya definición se expone a continuación:

  - **Modelado de datos:** un modelo de datos es un lenguaje orientado a hablar de una base de datos. Típicamente un modelo de datos permite describir las estructuras de datos de la base; el tipo de los datos que hay en la base y la forma en que estos se relacionan.
  - **Modelado de gestión:** el flujo de información entre las funciones de gestión se modela de forma que responda a las preguntas a dónde conduce el proceso de gestión, qué información se genera, quién la genera, a dónde se dirige la información y quién la procesó.
  - **Modelado de proceso:** cuando un proceso es modelado, con ayuda de una representación gráfica (diagrama de proceso), pueden apreciarse con facilidad las interrelaciones existentes entre distintas actividades, analizar cada actividad, definir los puntos de contacto con otros procesos, así como identificar los subprocesos que los componen.
  - **Generador de aplicaciones:** en el proceso RAD se trabaja para reutilizar, siempre que es posible, componentes de programas ya creados o, cuando sea necesario, crear

componentes reutilizables. Para ello, se utilizan herramientas automáticas que facilitan la construcción del software.

- **Pruebas de entrega:** los procesos RAD, acentúan la reutilización de componentes que ya se han usado y, por ende, comprobados. De este modo se logra reducir el tiempo empleado en probar el programa, aunque esto no exime de probar los componentes nuevos, ni todas las interfaces.

- **Modelo en espiral:** las actividades a desarrollar se conforman en una espiral en la que cada bucle representa un conjunto de acciones (véase la figura 4.11). Cada bucle se compone de cuatro acciones, *Planificar*, *Análisis de riesgo*, *Desarrollar y probar* y *Determinar objetivos.*

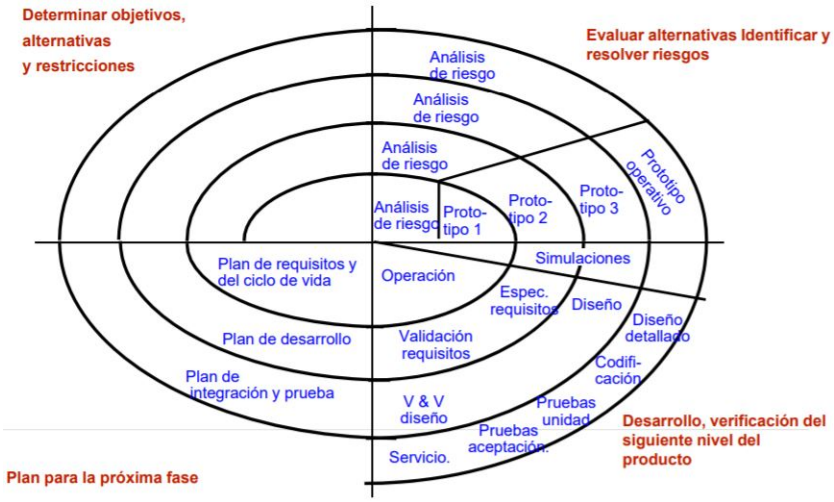

Figura 4.11: Diagrama Metodología Espiral. [García, 2009]

El modelo de espiral está compuesto de seis acciones, a saber:

- **Comunicación con el cliente:** tarea fundamental para conectar al cliente con el desarrollador y poder conocer sus necesidades.
- Planificación: tarea inherente a los requerimientos necesarios para llevar adelante el proyecto, (definición de recursos, tiempo, etc.)
- Análisis de riesgos: tarea de evaluación de información del proyecto y de los riesgos técnicos del mismo.
- Ingeniería: tarea de construir una o más representaciones de la aplicación.
- Evaluación del cliente: tarea requerida para la obtención de las reacciones del cliente en las evaluaciones del software creadas en la etapa de ingeniería. Esta tarea se implementa en la etapa de instalación.

- Construcción y entrega: tareas que se requieren para construir, probar e instalar el software desarrollado. Estas tareas prestan apoyo técnico a los usuarios.

- **Modelo incremental:** Se van desarrollando continuamente etapas en cascada para una pequeña parte de los sistemas. Este ciclo se va repitiendo hasta finalizar el proyecto (véase figura 4.12). Tal vez haya una necesidad imperiosa de dar rápidamente cierta funcionalidad limitada de software a los usuarios y aumentarla en las entregas posteriores de software. En tales casos, es apropiado utilizar este tipo de modelo.

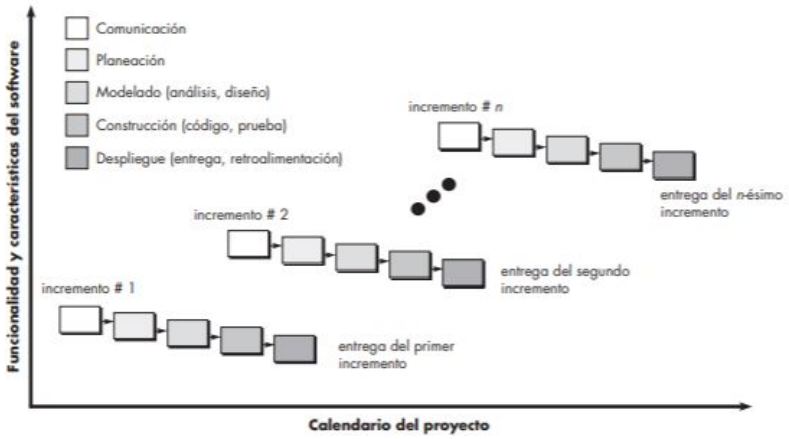

Figura 4.12: Diagrama Metodología Incremental. [Pressman, 2010]

- **Modelo en cascada:** En este modelo básicamente el software se desarrolla linealmente por etapas, en la que una vez completada una etapa no se puede volver a la anterior. Consta de las siguientes cinco etapas (véase figura 4.13):

  - **Análisis de requisitos:** Etapa en la que se analizan los requerimientos que debe cumplir el proyecto.
  - **Diseño del sistema:** Se descompone el sistema en elementos más pequeños, que posteriormente serán desarrollados por el equipo.
  - **Implementación:** En esta fase se implementa el código fuente del sistema.
  - **Pruebas o verificación:** Momento en el desarrollo donde se evalúa el funcionamiento y el cumplimiento de los requisitos.
  - **Mantenimiento:** Tal vez no cumpla los requisitos solicitados y sea necesario modificar su estructura.

  Como podemos observar, algunos de los métodos tradicionales, basados en estándares y rígidas normas, no permiten en la mayoría de los casos, que el cliente incorpore nuevas ideas al proyecto o modifique las existentes, por lo que estos métodos no son ni rápidos ni satisfactorios.

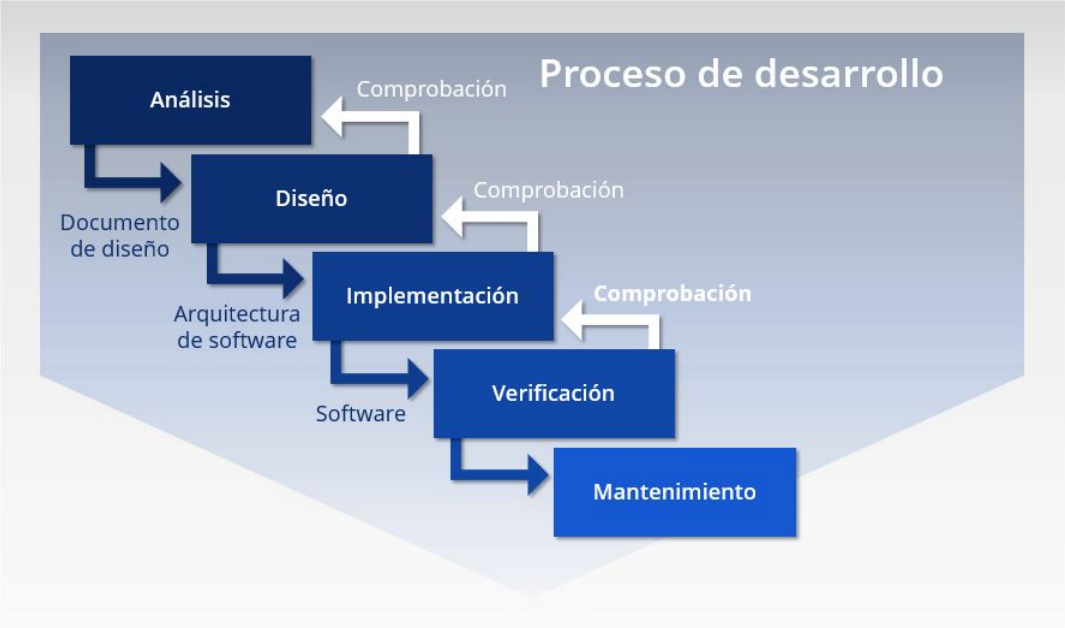

Figura 4.13: Diagrama Modelo Cascada. [SANABRIA, 2021]

Por ello y al no responder satisfactoriamente a las necesidades del cliente, surgieron las metodologías ágiles, intentando mejorar aquellos aspectos en los que las tradicionales no cumplían expectativas y fracasaban. En primer lugar, es necesario entender y explicar el término *Agile* en este contexto, que va mucho más allá del simple concepto de rápido. *Agile* es un conjunto de métodos y metodologías que ayudan a un equipo a pensar de manera más efectiva, trabajar eficientemente y tomar mejores decisiones en el entorno de trabajo. Por lo tanto, ágil también se considera una forma de pensar. *Agile* es, además de todo lo anterior, un movimiento que revolucionó la forma de desarrollar software, incluyendo un conjunto de ideas, valores y principios que dan forma a esta forma de pensar [Stellman and Greene, 2014].

Existen cuatro grupos de valores, que deben ser puestos en práctica por los desarrolladores en los proyectos:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

El manifiesto Ágil consta de doce principios:

1. **Satisfacción del cliente:** Es la base de todo. Se alcanza a través de la entrega de productos de valor que cubran una necesidad.

2. **Nuevos requisitos:** Cambiar sobre la marcha no es dar un paso atrás. Cualquier sugerencia o solución es bienvenida si se trata de mejorar el producto.
3. **Entregas por semanas:** La división del trabajo en fases productivas es la base de la metodología. En lo posible, ejecutar una cada semana.
4. **Medir el progreso:** La evolución de los procesos no es un elemento subjetivo. Se puede medir con indicadores concretos.
5. **Desarrollo sostenible:** La forma de ejecutar los proyectos debe garantizar en sí misma su continuidad. No es una cuestión de hacer por hacer.
6. **Trabajo cercano:** Los líderes de los proyectos deben ejercer su labor en el mismo terreno donde tienen lugar las tareas y no desde los despachos.
7. **Conversación cara a cara:** El gestor responsable debe comunicar de forma eficaz sus mensajes, mejor si se hace de forma presencial. Se recomiendan reuniones periódicas tanto con el cliente como con sus colaboradores.
8. **Motivación y confianza:** Los procesos solo tendrán éxito si quienes los llevan a cabo son personas motivadas y que interactúan en climas de confianza y solidaridad.
9. **Excelencia técnica y buen diseño:** Las formas nunca deben perderse, así como tampoco la calidad del trabajo.
10. **Simplicidad de las tareas:** Las tareas han de ser lo más sencillas posible. Si alguna no puede ser ejecutada en esos términos, debe ser dividida en iteraciones hasta que se reduzca su nivel de complejidad.
11. **Autogestión de los equipos:** Si bien debe existir una figura que monitorice los equipos de trabajo, estos deben ser capaces de organizarse por sí mismos. El exceso de jerarquías crea dependencia entre los colaboradores.
12. **Adaptarse a las circunstancias:** Los proyectos no suelen terminar de la misma forma en que empezaron. Es indispensable que quienes los ejecutan puedan adaptarse a las distintas circunstancias que puedan surgir.

Todo lo anteriormente expuesto, se fue integrando en las metodologías ágiles, que fueron apareciendo, de las cuales destacamos, Scrum, Kan-Ban, eXtreme Programming y Lean.

Metodologías ágiles

Dentro de la categoría de metodologías ágiles podemos encontrar metodología Scrum, eXtreme Programming entre las más destacadas.

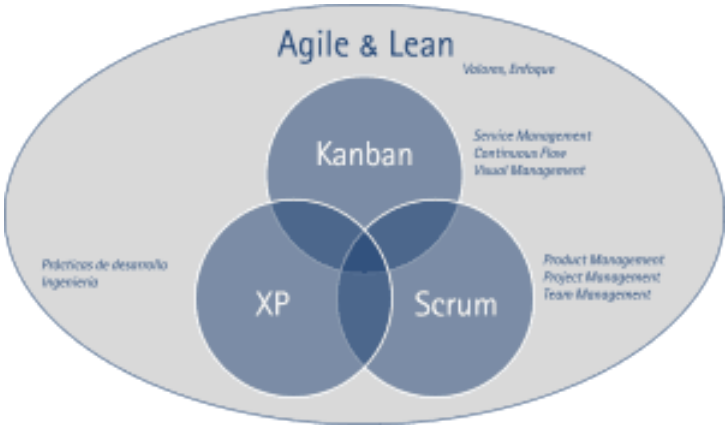

Figura 4.14: Diagrama metodologías ágiles [Rodríguez, 2014].

Metodología Scrum

La metodología Scrum se comenzó a desarrollar en la década de 1980 debido a las necesidades de procesos de reingeniería por parte de Goldratt, Takeuchi y Nonaka. En esta época surgió un cambio en los procesos de desarrollo utilizados en productos exitosos en Japón y Estados Unidos. Estos procesos partían de requisitos muy generales y debían salir al mercado en periodo de tiempo inferior al tradicional.

Esta metodología que surgió como respuesta fue Scrum, que se basaba en el trabajo de un equipo multidisciplinar y que implementa diferentes mecanismos para controlar la incertidumbre y la flexibilidad.

En Scrum, se trabaja realizando entregas parciales de una forma regular del producto final. Estas entregas priorizan los requisitos con base en las necesidades del cliente final. Esta característica determina que Scrum sea la metodología ideal para proyectos en un entorno complejo con unos requisitos de plazo de entrega muy ajustados y además estos requisitos pueden sufrir ligeras variaciones a lo largo del desarrollo del proyecto, ya que lo importante es la innovación y la productividad fundamentalmente. De este modo, Scrum permite resolver situaciones de retrasos de entregas al cliente final, así como controlar los costes de desarrollo del proyecto, aumentar la calidad del proyecto para mejorar la eficacia del equipo.

La metodología Scrum se fundamenta en una serie de puntos principales que determinan la operatividad de esta metodología:

- **Desarrollo incremental de los requisitos del proyecto:** Se realiza un desarrollo incrementa por bloques del proyecto. Esto permite que durante el periodo de la iteración que

puede ser desde dos semanas hasta un mes, se debe realizar todas las tareas necesarias para completar el desarrollo (análisis, desarrollo, pruebas y documentación) y preparar todo para la entrega al cliente final. Este método evita retrasar ciertas tareas críticas relacionadas con la entrega de requisitos.

- **Priorización de requisitos:** En el desarrollo óptimo del proyecto, es necesario priorizar los requisitos en cada iteración según el valor que proporcionan al cliente y el coste estimado. Estos requisitos se almacenan en la lista priorizada llamada **Product Backlog**.
- **Control del proyecto:** Diariamente el equipo se reúne y se coordina para realizar las adaptaciones necesarias **(Daily Scrum)**. Igualmente, el cliente puede tomar decisiones en función de las entregas que reciba tras cada iteración **(Sprint)**.
- **Responsabilidad de equipo:** Cada miembro del equipo se compromete a entregar unos requisitos y por ello se permite que planifique su trabajo libremente.
- **Colaboración y comunicación:** Es necesario para poder coordinarse correctamente entre los miembros del equipo y con el cliente.
- **Fijar objetivos:** Un factor clave para la realización correcta de un proyecto es la planificación temporal de los objetivos, fijar plazo determinados para poder priorizar objetivos y/o tareas favoreciendo la toma de decisiones.

La metodología Scrum necesita cumplir ciertos requisitos que permitan que el grupo o empresa donde se aplique pueda funcionar:

- **Cultura de empresa:** El trabajo en equipo, la creatividad, el saber delegar son fundamentales para trabajar en la búsqueda de la mejora continua.
- **Compromiso del cliente:** Gestión del aspecto económico y disponibilidad para poder colaborar.
- **Compromiso dirección:** Fundamental para la resolución de problemas frecuentes y realizar cambios organizativos. Se basa en una cultura de la colaboración, autogestión de los equipos multidisciplinares facilitada por los líderes del equipo.
- **Tamaño del equipo:** Es necesario que el equipo tenga un tamaño que permita que se coordinen con facilidad, y en caso de intervención de más de un equipo al mismo tiempo, aplicando técnicas de coordinación para maximizar la comunicación entre todos los miembros.

Todo esto define a la metodología Scrum, una de las metodologías más utilizadas hoy en día.

Metodología XP: *Agile* a nivel de software

La metodología de programación extrema (XP) marca un punto de inflexión dentro de la ingeniería del software. Esta nueva metodología de trabajo fue creada por Kent Beck[Beck, 2004], Ward Cunninghamn y Ron Jeffries [Jeffries, 2004] a finales de la década de los noventa. Los creadores sentaron las bases de un cambio disruptivo de pensamiento que pone énfasis en la adaptabilidad y no en la previsibilidad de las metodologías tradicionales. Esto le permite poder adaptar los requisitos del proyecto a los cambios que surjan en cualquier momento, en lugar de centrarse en definir los requisitos al comienzo del proyecto e intentar controlar los cambios del proyecto para cumplir los requisitos.

Esta metodología se centra en la potenciación de las relaciones interpersonales como clave para el éxito del desarrollo del software. El trabajo en equipo genera un buen clima de trabajo en el cual los desarrolladores puedan aprender y relacionarse entre ellos. La comunicación entre los miembros del equipo es fundamental al igual que la realimentación continua entre el cliente y el equipo es clave para el éxito.

La metodología XP se define especialmente para proyectos con requisitos cambiantes y ciertamente imprecisos, con un alto riesgo técnico. Para lograr los objetivos, se categoriza la metodología en tres capas que agrupan doce prácticas básicas (véase figura 4.15).

Figura 4.15: Diagrama de capas XP. [Fernández González, 2012].

1. **Metodología de programación:** diseño sencillo, *testing*, refactorización[7] y codificación con estándares.

[7]Se denomina *refactoring* al proceso sistemático por el cual se mejora un código sin añadir nuevas funcionalidades.

2. **Metodología de equipo:** propiedad colectiva del código, programación en parejas, integración continua, entregas semanales e integridad con el cliente.

3. **Metodología de procesos:** cliente in situ, entregas frecuentes y planificación.

- **Diseño sencillo:** Para facilitar la lectura, un código sin redundancia ni duplicidades y que supere todos los test de funcionalidad, integración y aceptación.

- **Prueba (*Test*):** La base fundamental de esta metodología son los test, que no pretender resolver errores, sino evitarlos. Los test han de ser de tres tipos y siempre automatizados:

  - Test de aceptación: Realizado junto al cliente, se definen las necesidades funcionales primero.
  - Test unitario: Creado por el programador como comprobación del correcto funcionamiento de todos los métodos de una clase.
  - Test de integridad: Este test se encarga de evaluar el correcto funcionamiento del software en desarrollo en todo su conjunto.

- **Estándares de codificación:** Los desarrolladores deben seguir unas normas de codificación común que facilite que todo el equipo pueda entender el código. La nomenclatura común es clave a la hora de facilitar la comprensión, modificación y refactorización del código.

- **Propiedad colectiva del código:** La propiedad del código no es individual, es decir, no es el código de quien lo escribió sino que es el código del grupo. De esta forma la refactorización y la comprensión de todo el código no tiene límites.

- **Programación por parejas (*Pair programming*):** Permite que el trabajo de codificación llegue a buen puerto, pues la organización es la base y con la ayuda de otra persona siempre pueden aparecer errores que no se han detectado previamente. Dicha organización requiere una buena técnica y una buena estrategia, por ello, establecer pasos a seguir, ver cómo abordarlos y llevarlos a cabo, hará que la programación se realice de la manera adecuada. Además, hará que el exceso de trabajo y el estrés que puede producir este trabajo decrezca si se realiza entre dos.

- **Integración continua:** Esta es una de las principales ventajas que tienen los equipos de XP, pues mientras se van desarrollando las primeras funcionalidades estas ya están trabajando en el sistema. También, si hay un error se puede identificar y ponerle remedio con rapidez. Y finalmente, si se quiere trabajar en el código se puede modificar en la última versión, no hace falta retroceder.

- **Dedicación plena:** No es para nada aconsejable trabajar más de las horas establecidas y en franjas proporcionales, pues decrecerá la motivación y la calidad del trabajo.
- **Metáfora del negocio:** Los conocimientos y el vocabulario que tenga el programador no serán los mismos que el cliente, por eso hay que adaptar el mensaje final al nivel del receptor para que pueda entender los conceptos del tema que se va a tratar.
- **Cliente in situ:** Para trabajar con los equipos de XP es esencial que haya una buena comunicación con el cliente, pues si surge alguna duda es el cliente quién tiene que dar especificaciones sobre lo que quiere. Por ello, el cliente llega a formar parte del propio equipo, ayudando a la agilización del proceso.
- **Entregas frecuentes:** Crear demos del sistema es una buena herramienta a largo plazo, aunque no tenga todas las funcionalidades, pues el equipo de programación podrá ver si es funcional y el cliente irá familiarizándose con el manejo.
- **Planificación incremental:** La planificación no va a ser siempre la que se ha pensado desde un principio, pues irán surgiendo obstáculos durante el proceso y habrá que adaptarse a la situación volviendo a trabajar sobre las cuatro variables de la metodología XP. Esta capacidad y la revisión frecuente del proyecto hará que el proceso se agilice.

Las variables que definen cualquier proyecto software con una metodología XP son las siguientes:

- **Coste**
- **Tiempo**
- **Calidad**
- **Alcance**

El objetivo de esto es intentar alcanzar un equilibrio entre las cuatro variables, que ninguna destaque sobre otra. De estas cuatro variables, tres pueden ser fijadas por actores externos al equipo de desarrollo (clientes, responsables de proyecto, etc.) y la variable restante debe ser establecida por el quipo de desarrollo basándose en las otras tres variables fijadas previamente.

Elección de la metodología de desarrollo

Para finalizar este punto sobre metodologías de desarrollo existentes, después de haber repasado las metodologías tradicionales y las metodologías ágiles, se ha elegido desarrollar el presente proyecto utilizando la metodología ágil XP (*eXtreme Programming*). Esta elección de metodología se debe

a las características del proyecto: se espera realizar modificaciones constantes posteriores a nuevos requisitos y se presenta una necesidad de ciclo de trabajo flexible. Por todo lo anterior, se ha seleccionado la metodología XP, siendo esta además, una de las metodologías más utilizadas. El proceso de aplicación de esta metodología se va a llevar a cabo en el próximo capítulo.

4.4. Diseño de la aplicación

El diseño del sistema se compone de dos partes: el diseño del software y el diseño instruccional del curso.

Diseño del software

Para el desarrollo del software se ha aplicado la metodología XP de tipo *Agile* al proyecto frente a otras alternativas por los siguientes motivos:

- Esta metodología permite un desarrollo iterativo sobre el que se van realizando mejoras. A lo largo del proyecto se van a realizar reuniones con el cliente (este rol será representado por la tutora de este proyecto) en las que se irán definiendo los requisitos del proyecto.
- La capacidad de adaptación y flexibilidad de esta metodología son claves para un desarrollo de estas dimensiones.
- El buen código y fácil de entender, es una de las características de esta metodología implementada mediante el mecanismo de refactorizar (*refactoring*).
- En esta metodología no es necesario tantos roles en el equipo ni existe el concepto de *sprint* que favorece la previsibilidad.

Una vez establecida la metodología de desarrollo elegida y los motivos por los cuales se ha decidido optar por ella se expone el proceso de desarrollo. Como ya se describió en la sección 4.3, la metodología XP se basa en la adaptación del proyecto a medida que este avanza. Esta metodología establece una serie de iteraciones en las que se producen reuniones entre los miembros del equipo para fijar los objetivos de la iteración y los requisitos sobre los que se va a trabajar.

La fase de diseño del proyecto se centrará en la creación de los entornos virtuales y su funcionamiento para realizar una simulación del comportamiento del entorno real asociado. Unity permite diseñar entornos virtuales utilizando todos los objetos disponibles dentro del entorno para esta función.

El desarrollo en Unity está definido por el uso de entornos en dos dimensiones (2D) y entornos en tres dimensiones (3D). Para cumplir las restricciones y objetivos anteriormente establecidos, se va a desarrollar el proyecto en un entorno 3D de aspecto más realista.

La metodología de desarrollo en Unity se basa en el uso de componentes, objetos y escenas como base de utilización de la plataforma de desarrollo. Por este motivo, se comenzará por el diseño de las estancias del mundo virtual de las que constará el proyecto, que serán contenidas en las diferentes escenas que se van a diseñar.

Reuniones Scrum

El seguimiento del proyecto se ha realizado, siguiendo la metodología elegida, mediante reuniones periódicas entre el alumno y la tutora de este Trabajo Final de Grado.

En estas reuniones se han analizado los requisitos, propuestas de diseño y todas las ideas de implementación que han ido surgiendo a lo largo de desarrollo del proyecto, del mismo modo que se realizaran las correcciones necesarias sobre la base de estas reuniones.

Se han realizado 3 reuniones en las cuales se ha definido el diseño, basado en los componentes que proporciona el entorno de desarrollo de Unity, y la funcionalidad a implementar, adecuada al diseño basado en los objetivos y requisitos necesarios.

Primera reunión

En esta primera reunión se fijan las bases del diseño tomando como punto de partida la sección 3 en donde se sitúan los objetivos del proyecto. En el punto 3.2 se define el *objetivo 1* del proyecto en el cual se indica "Diseñar y desarrollar los entornos virtuales que simulen el entorno real de la Escuela Politécnica", con lo cual se toma como base para establecer el diseño básico del entorno virtual.

En esta primera reunión, se han establecido como estancias a diseñar las siguientes:

1. Entrada al edificio (*hall*)
2. Aula de teoría
3. Laboratorio

Estas tres estancias permiten plantear el entorno virtual de aprendizaje como una simulación realista de la Escuela Politécnica de la Universidad de Alcalá, que se compone de un edifico con

múltiples estancias como son: el *hall* de entrada, una serie de pasillos y escaleras, unos despachos y unas clases de teoría y de laboratorios de diferentes tamaños. Por simplicidad y la envergadura de este trabajo se ha decidido simplificar el entorno.

A continuación, se expone el diseño establecido durante la reunión de cada una de estas estancias que por simplicidad se han creado en diferentes escenas (véase el punto 4.2) lo que permite así independizar el diseño y el desarrollo de cada una de ellas reduciendo las dependencias entre escenas.

Entrada al edificio (*Hall*)

La estancia de entrada al edificio (véase figura 4.16), contendrá la puerta principal y las puertas a las diferentes clases y/o estancias que componen el mundo virtual, sirviendo como punto de entrada y de interconexión entre las diferentes secciones del mundo virtual. Las posibles acciones que puede realizar el usuario son las siguientes:

1. **Acceder al laboratorio:** Cuando el alumno sitúa el avatar cerca de la puerta, a la izquierda de la imagen, con el letrero que indica **Acceso a laboratorio**, se realizará una animación y se producirá un cambio de escena en la que se situará al usuario representado con el avatar a la escena *Laboratory*.
2. **Acceder a la clase:** Cuando el usuario aproxime el avatar a la puerta que da acceso a la clase, donde hay un letrero que indica **Acceso a clase**, se realizará una animación y un cambio de escena que situará el avatar en la escena *Classroom*.
3. **Volver al menú principal:** Si el alumno quiere salir del entorno virtual desde la escena actual, debe situarse cerca de las puertas dobles, donde hay un letrero con el rótulo **Salida**.

La composición de la escena del *hall* se ha realizado gracias a los componentes que se pueden apreciar en la figura 4.17 y que se detallan a continuación:

- **Directional Light:** Componente encargado de iluminar la escena mediante un objeto que proyecta luz con un patrón direccional.
- **CameraCine:** Implementa la cámara que va a visualizar el alumno; es la vista de escena. Este objeto es controlado por el componente *CM vcam1* mediante el complemento (*plugin*) *Cinemachine*.

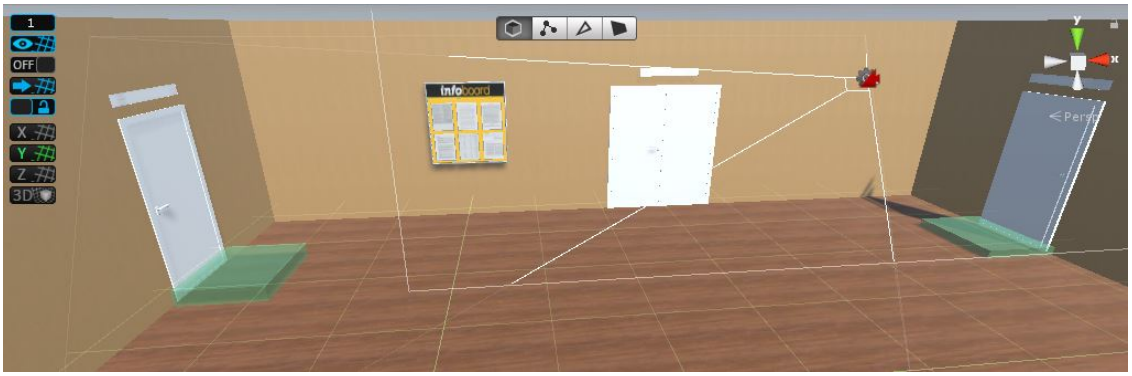

Figura 4.16: Diseño del *Hall*.

Figura 4.17: Estructura de componentes de la escena *Hall*.

- **Hall:** Es un *GameObject* vacío que contiene como hijos los objetos que componen la estancia, paredes, suelo, puertas, etc.
- **ThirdPersonController:** El avatar con el que el alumno interactúa. Se compone de un objeto 3D con un controlador para poder detectar las teclas que presiona el alumno en el teclado y realizar las acciones correspondientes.
- **CM vcam1:** Se trata de un *transform* sobre un objeto *GameObject* vacío que representa una cámara virtual. Esta cámara virtual se considera un controlador de cámara y rastrea el *GameObject* que es de su interés, posicionándose en consecuencia tras el avatar.
- **Canvas:** Este *GameObject* es el padre de todos los componentes UI de la escena. Dentro de este se encuentra el componente *Canvas* en el que se representa el área donde se sitúan los componentes. Este componente generalmente se representa por un rectángulo en la vista de diseño. En la vista de escena, el usuario ve todos los componentes situados sobre este rectángulo o área *Canvas* los cuales se dibujan en orden jerárquico, si un componente se solapa con otro, se muestra el último primero.
- **Event System:** El componente *Event System*, se encarga de gestionar la entrada por teclado y determinar que *GameObject* se considera como seleccionado. Este componente está ligado al componente *Canvas*.

Para el ajuste de la correcta iluminación, se ha realizado un procesamiento de las luces en la escena. Este tratamiento de las luces se ha realizado mediante unos paquetes externos (*Post processing* y *Loght-weight RP*) instalados en el entorno para este fin. El efecto es visible para el usuario al gestionar de forma correcta la capa en la que se aplica este efecto, ya que en el proceso de generación (renderizado) es necesario establecer un orden de capas para poder situar este procesamiento de la imagen en el correcto orden para permitir ser visualizado y que la merma de rendimiento no sea elevada. Al realizar un tratamiento de las luces de la escena, se consigue una imagen más realista del entorno, simulando un entorno real.

Aula de teoría

El aula de teoría (véase figura 4.18) es fundamental en el diseño del entorno virtual, permitiendo simular las aulas de teoría real donde se lleva a cabo el proceso de aprendizaje de aspectos principalmente teóricos de la asignatura o materia que se pretende enseñar.

Figura 4.18: Diseño del aula de teoría del entorno virtual para el aprendizaje de materias.

En la siguiente reunión se detallará el desarrollo de esta escena, así como el valor que aporta al proyecto.

Laboratorio

Frecuentemente, en las asignaturas técnicas se divide su aprendizaje en clases de teoría y clases de práctica. Esto hace necesario crear dos estancias donde exponer de forma diferente los distintos tipos conocimientos. En el laboratorio se realizarán actividades prácticas sobre los conocimientos teóricos expuestos en el aula de teoría.

En esta estancia (véase figura 4.19) se puede observar la recreación en tres dimensiones de un

aula de laboratorio de informática. Esta aula recrea, con gran nivel de detalle un entorno real. Se simula el entorno de un laboratorio de la Universidad, en el que se disponen de diferentes elementos funcionales que van a intervenir en la impartición de las clases.

Figura 4.19: Captura de pantalla laboratorio.

Las opciones de interacción que pueden realizarse por parte del usuario mediante el avatar son las siguientes:

1. **Movimiento del avatar:** El alumno podrá mover el avatar utilizando los controles. Estos se podrán utilizar con el uso de las teclas del teclado.
2. **Acceso a la actividad:** Cuando el avatar del alumno se aproxime a la zona donde están situadas las mesas y los ordenadores, el alumno perderá el control del avatar y accederá al menú del equipo. Esto se realizará mediante una animación.
3. **Salir al *hall*:** Si el alumno sitúa al avatar en la posición original de la que se parte en la escena, se realizará un cambio de escena para volver a la escena *Hall*.

Esta estancia se compone de diferentes objetos, algunos prefabricados y otro no, que se sitúan en la jerarquía siguiente (véase figura 4.20):

- **Main Camera:** Representa la cámara principal de la escena, que se encuentra estática en la posición que se puede ver en la figura 4.19.
- **Directional Light:** Será el componente encargado de iluminar la escena de forma general y direccional.

Figura 4.20: Estructura de componentes de la escena *Laboratory*.

- **Floor:** Este componente representa el plano sobre el que se sitúan los demás componentes que constituyen la escena.

- **Room:** Se trata de un componente *prefab*, la estancia que representa el aula al igual que en la escena *Classroom*. Sé instancia de nuevo en esta escena y se añaden los componentes necesarios característicos de esta aula, como son los elementos de escritorio, monitor, ordenador, teclado y ratón.

- **Canvas:** En esta escena el componente *Canvas* se utiliza para poder realizar la transición entre escenas.

- **Alumno:** Está formado por varios subcomponentes que representan el avatar que puede controlar el usuario.

- **EventSystem:** Componente *EventSystem*, con el cual se llevan a cabo diferentes acciones. Se encarga de gestionar la entrada por teclado y, también, tiene como objetivo determinar que *GameObject* se considera como seleccionado. Este componente está ligado al componente *Canvas*.

Segunda reunión

La primera reunión fija las bases del diseño del proyecto y en esta segunda sesión se ampliarán esas bases del diseño, realizando añadidos y rectificaciones a todo lo anterior.

En la primera sesión, se presuponía que en la estancia del **Aula de teoría** se iban a proporcionar los conocimientos teóricos al alumno y en la estancia del **Laboratorio** se iba hace lo mismo con los conocimientos prácticos. En esta segunda reunión se ha decidido prescindir de esta sala (aunque en esta fase ya se ha desarrollado) debido a que a nivel del proceso de aprendizaje es más correcto exponer los conceptos en un mismo espacio.

En la gestión de los espacios virtuales se trabaja mediante el uso de escenas (véase punto 4.2). Cada escena engloba diferentes objetos y componentes en un mismo entorno virtual. La mecánica de diseño implica el uso de diferentes escenas y la relación entre estas. El diseño de escenas se plantea de la siguiente manera:

1. **Escena de menú principal:** Es el inicio del mundo virtual, donde se representa el menú principal que contendrá diferentes opciones como son: a) acceder al mundo virtual, b) obtener información acerca del entorno virtual y c) finalizar la ejecución de la aplicación.

2. **Escena de *hall* o recibidor:** Se llevará a cabo una escena que recree, mediante diferentes componentes 3D, una simulación de un recibidor o *hall* de una universidad donde se pueda acceder a otras estancias. Esta estancia servirá como punto de interconexión entre diferentes secciones del mundo virtual.

3. **Escena aula de práctica (laboratorio):** Al basarse este proyecto en el aprendizaje de una asignatura mediante un entorno virtual, es necesario diseñar una estancia dedicada a la docencia de contenido teórico en la que el alumno pueda adquirir los conocimientos necesarios para comprender la asignatura en estudio.

4. **Escena de resolución ejercicio o actividad:** Para fijar los conocimientos teóricos, se usan las actividades prácticas que refuerzan la adquisición de conocimientos mediante ejercicios basados en los conceptos expuestos en la parte de teoría. En esta escena el alumno podrá resolver los diferentes casos o ejercicios prácticos que le serán planteados.

En esta reunión se decide añadir dos nuevos espacios que comprenden el menú principal y la escena de actividad. A continuación se pasan a detallar en mayor profundidad estas escenas:

Escena Menú principal

Esta escena contiene un menú principal para el proyecto (véase figura 4.21). En esta se proporciona una barra lateral con un fondo de color azul. Aquí se mostrarán tres opciones:

1. **Acceder:** Esta primera opción va a permitir al alumno tener acceso al entorno virtual creado y comenzar con la exploración del proyecto. Su selección conlleva automáticamente un cambio de escena en la que el usuario se situará en la escena *Hall*.

2. **Acerca de:** Esta segunda opción mostrará una pantalla con un cuadro de texto en el cual se podrán ver todos los detalles relacionados con el proyecto.

3. **Salir:** Como última opción, se permite al usuario salir y finalizar la ejecución.

Figura 4.21: Menú principal.

En esta escena se pueden apreciar además, varios componentes con un importante papel. A continuación se detalla la estructura de la escena (véase figura 4.22):

Figura 4.22: Estructura de componentes de la escena Menú principal.

- **Main Camera:** Se trata de la cámara principal (*main camera*) que va a permitir al alumno visualizar la escena.

- **UIController:** Permite generar el menú con el que el usuario va a poder interactuar y con el cual tendrá acceso al entorno virtual de aprendizaje creado. El alumno puede clicar sobre la opción **Acceder** y una vez pulsada, se realiza un cambio de escena mediante el gestor de escenas de Unity, o puede pulsar en **Acerca de**, y se mostrará un panel con información del proyecto.

- **Event System:** Se encarga de diferentes tareas. En primer lugar, gestiona la entrada por teclado. Y en segundo lugar, determina que *GameObject* se considera como seleccionado. Este elemento está ligado al componente Canvas.

- **Game Manager:** Se encarga de almacenar en variables valores relativos al estado del entorno virtual de aprendizaje. Además, también es importante añadir que se mantiene entre los cambios de escena.

Escena de Actividad

La última escena de esta reunión, es la escena de actividad (véase figura 4.23) clave en la asimilación y verificación de los nuevos contenidos. Se caracteriza por implementar componentes dinámicos que permiten generar ejercicios con un enunciado dinámico, de forma aleatoria y detectando los errores del usuario en tiempo real. El diseño de esta actividad se abordará en mayor profundidad en el apartado 4.4.

Figura 4.23: Escena de Actividad.

A continuación se enumeran las diferentes acciones que puede realizar el usuario en la escena:

1. **Realizar ejercicio:** Siguiendo el orden establecido por la ejecución, el alumno podrá seleccionar el valor de referencia haciendo clic y situarlo en el marco de memoria[8] deseado. Esto

[8]Bloques de tamaño fijo de memoria virtual.

disparará un evento de corrección antes de desbloquear la siguiente columna. En caso de cometer dos errores, se mostrará un mensaje y se devolverá al alumno a la diapositiva de la teoría correspondiente.

2. **Salir:** Esta opción, a la cual se accede pulsando la tecla Escape, devuelve al alumno a la diapositiva desde la que había accedido a la actividad.

La jerarquía de componentes de esta escena (véase figura 4.24) es de las más extensas y cuenta con un notable aumento del número de objetos que la componen en comparación de las escenas anteriores. A continuación se detalla cada componente dentro de la jerarquía:

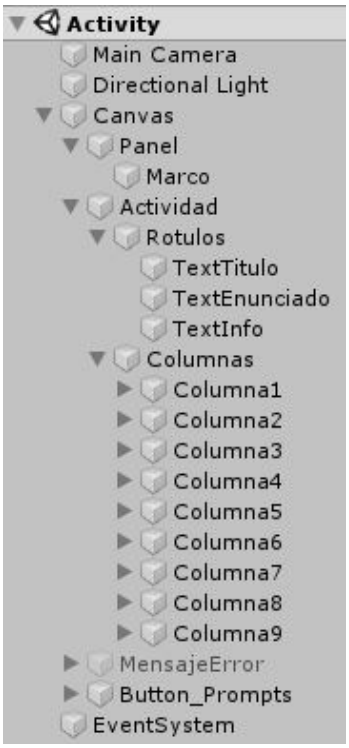

Figura 4.24: Estructura de componentes de la escena Actividad.

- **Main Camera:** Este componente permite que el alumno pueda visualizar la escena, representa el visor de la escena.
- **Directional Light:** Componente encargado de iluminar la escena de forma general y direccional.
- **Canvas:** Este *GameObject* se utiliza, como en las anteriores escenas, para contener todos los componentes de la interfaz de usuario (UI).

  A continuación se describen los componentes hijo que conforman el objeto *Canvas*:

- **Panel:** Este primer componente del *canvas* se encarga de alojar el marco, una imagen importada y convertida a *spriteUI*[9], para generar el borde negro que simula el aspecto de una aplicación ejecutándose en un ordenador.

- **Actividad:** Elemento principal de la escena. Contiene los rótulos de la actividad y las columnas. Estos componentes son la clave de la actividad e incorporan los diferentes *scripts* que proporcionan funcionalidad a los objetos y permiten la interacción del usuario con el entorno para una mayor adquisición de los contenidos (algoritmos) a la vez que se realiza un aprendizaje más intuitivo.

- **MensajeError:** Cuando el alumno comete un error en algún paso durante la resolución del algoritmo de gestión de memoria, se activa este componente. En el momento de activación, se hace visible al usuario junto con un mensaje de error aclarativo.

- **Button_Prompts:** El sistema de ayuda se desarrolla mediante el uso de unos iconos intuitivos y unos mensajes como leyenda que aportan significado a las acciones que implementan.

- **EventSystem:** El componente *EventSystem*, se encarga de gestionar la entrada por teclado y determinar qué *GameObject* se considera como seleccionado. Este componente está ligado al componente *Canvas*.

Tercera reunión

Una vez definidas las escenas anteriores, falta ubicar los contenidos teóricos del curso y crear un mecanismo de acceso desde la teoría a la actividad práctica. Para ello se crea una nueva escena que sirve de enlace y soporte a los contenidos del curso.

Escena Principal

En el desarrollo de la exposición teórica de los contenidos a la que tiene acceso el usuario, se ha implementado un menú de selección con todas las opciones posibles para que alumno pueda comprobar el temario. Del mismo modo se ha llevado a cabo un sistema de visualización del contenido basado en el concepto de diapositiva (*slide*), añadiendo a su vez un sistema de navegación que permita al usuario seleccionar entre las diferentes diapositivas. Dentro de cada categoría presentada en el menú se tendrá acceso a cada área del temario (véase figura 4.25).

Al situarse en esta escena el alumno puede realizar las siguientes opciones:

[9]Gráficos en 2D aptos para una interfaz de usuario.

1. **Seleccionar opción del menú:** Haciendo clic en alguno de los botones del menú, el alumno puede acceder a los diferentes recursos disponibles.

2. **Salir:** Para poder salir de la escena, el usuario puede seleccionar el botón rotulado con la palabra salir o puede pulsar la tecla escape del teclado. De este modo, volverá a la escena de menú principal.

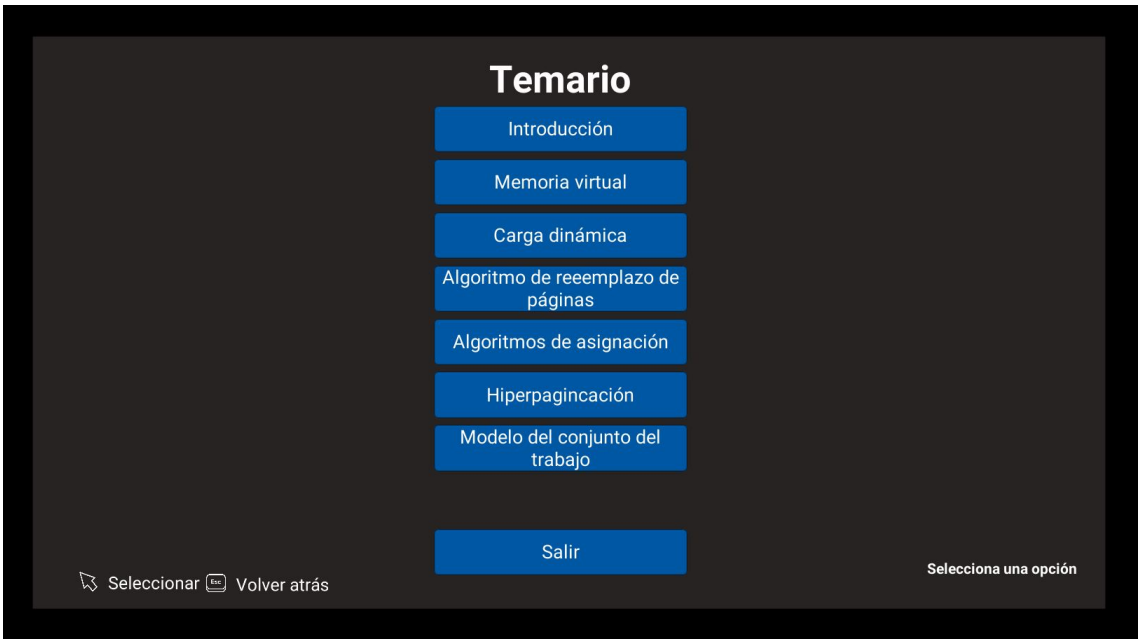

Figura 4.25: Escena principal.

La estructura de esta escena es de las más complejas a la hora de desarrollar el proyecto, teniendo en cuenta que en esta intervienen múltiples tipos de componentes basados tanto en *assets* como en *prefabs*. Debido a esta integración, hay componentes que en el estado inicial van a estar desactivados y que durante la ejecución se activarán cuando sean requeridos. Este hecho estará determinado por las acciones que realice el usuario. A continuación, se detallan los diferentes componentes situados en la jerarquía de componentes de la escena (véase figura 4.26):

- **Main camera:** Este componente permite que el alumno pueda visualizar la escena; representa el visor de la escena.

- **Directional Light:** Componente encargado de iluminar la escena de forma general y direccional.

- **Canvas:** Este Game Object es el padre de todos los componentes UI de la escena. Dentro del *Canvas* se encuentran varios elementos. Entre ellos, existen dos componentes clave:

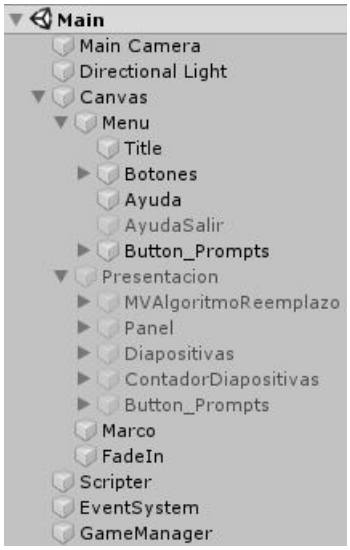

Figura 4.26: Estructura de componentes de la escena principal.

- **Menu:** Este componente está activo por defecto y permite al usuario seleccionar entre los diferentes botones con lo que cuenta. El menú se compone de un título, unos botones y unas indicaciones de ayuda para el alumno.
- **Presentacion:** El segundo componente más importante de la escena es la presentación. Inicialmente, se encuentra desactivado, pero si el usuario hace clic en algún botón correspondiente a algún tema, se activará la presentación y se ocultará el menú. La presentación se compone de las diapositivas del temario, situadas dentro de un *GameObject* vacío llamado *MVAlgoritmoReemplazo*.

- **Scripter:** La mayoría de los *scripts* que controlan el funcionamiento de la presentación se han centralizado en este *GameObject* vacío. Dentro, encontramos el *script* Presentador, encargado de activar la presentación o el menú, el *script* presentación, el cual se encarga de dar funcionalidad a la presentación y la capacidad de interacción con el usuario, por último, el *script* Generar Jerarquia Presentación, encargado de recorrer el listado de hijos del componente MVAlgoritmoReemplazo y listarlos en el panel de navegación.

- **Event System:** El componente *EventSystem*, se encarga de gestionar la entrada por teclado, determinar que *GameObject* se considera como seleccionado. Este componente está ligado al componente Canvas.

- **GameManager:** Componente encargado de almacenar en variables valores relativos al estado del entorno virtual de aprendizaje. Este componente se mantiene entre los cambios de escena. Por motivos de depuración del entorno virtual, se añadió este componente en la escena

MainMenu y en esta escena. Únicamente sé instancia un componente al mismo tiempo; no se duplica.

Diseño instruccional del curso

En el diseño de los recursos en los que se basa la presentación y la actividad, se han creado a base de conceptos del diseño instruccional. Según [L.Broderick, 2001] el diseño instruccional es el arte y ciencia aplicada de crear un ambiente que favorezca el aprendizaje, los materiales claros y efectivos, que facilitan al alumno el desarrollo de la capacidad para lograr ciertas tareas.

El diseño instruccional describe el método que permite a los estudiantes alcanzar los objetivos de aprendizaje después de llevar a cabo un conjunto de actividades utilizando los recursos de un entorno [Amorim et al., 2006]).

Todo esto junto con el auge de Internet ha provocado un desarrollo en los estándares que representan el contenido del aprendizaje, recursos educativos y metodologías de diseño instruccional de tal modo que, los materiales y diseños educativos realizados en una plataforma puedan ser compartidos en otras plataformas [Párraga, 2010].

El modelo ADDIE, como diseño de sistema de instrucción, fue desarrollado por las fuerzas armadas estadounidenses, a través del centro de tecnología educativa de la Universidad Estatal de Florida, en los años 70 [Branson et al., 1975].

Para el diseño de la materia a enseñar en el entorno se ha elegido el modelo ADDIE, uno de los modelos de diseño instruccional más usados actualmente.

Este modelo se plantea como un diseño instruccional interactivo, que consta de cinco fases básicas y en el cual la autoevaluación es clave tanto en cada fase del diseño como del producto final, pudiéndose introducir cambios y mejoras continuas (véase figura 4.27).

ADDIE es el acrónimo del modelo, definido a partir de sus fases:

- **Análisis:** Es necesario estudiar la situación y las necesidades formativas del alumno.
- **Diseño:** Se desarrolla un programa del curso deteniéndose especialmente en el enfoque pedagógico y en el modo de secuenciar y organizar el contenido.
- **Desarrollo:** La creación real (producción) de los contenidos y materiales de aprendizaje basados en la fase de diseño.

Figura 4.27: Modelo ADDIE [Ceibal, 2019].

- **Implementación:** Ejecución y puesta en práctica de la acción formativa con la participación de los alumnos.
- **Evaluación:** Esta fase consiste en llevar a cabo la evaluación formativa de cada una de las etapas del proceso ADDIE y la evaluación sumativa a través de pruebas específicas para analizar los resultados de la acción formativa.

Una vez establecida la metodología de diseño instruccional, se definen los objetos de aprendizaje que forman parte del aprendizaje de la materia a enseñar. El desarrollo de este curso se va a centrar en la asignatura de Sistemas operativos. En concreto, el aprendizaje de la gestión de la memoria virtual, y más específicamente, en los algoritmos de reemplazo de página de un grado de Informática de la UAH. Sin estos algoritmos la gestión de la Memoria Virtual(MV) [Prieto, 2005] sería inoperable. Este tema posee características técnicas complejas. Los alumnos necesitan un apoyo práctico continuo, basado en ejemplos sencillos. Un aprendizaje teórico independiente, sin una aplicación detallada e inmediata, puede dificultar enormemente su aprendizaje y, por tanto, la adecuada interiorización de los conceptos y procedimientos asociados.

Fijado la metodología instruccional, se pasa a definir los Objetivos de Aprendizaje (OA) del curso a desarrollar. Pero antes de eso, se define que es un Objetivo de Aprendizaje como “Una entidad digital, autocontenible y reutilizable, con un claro propósito educativo, constituido por al menos tres componentes internos editables: contenidos, actividades de aprendizaje y elementos de contextualización. A manera de complemento, los objetos de aprendizaje han de tener una estructura (externa) de información que facilite su identificación, almacenamiento y recuperación: los

metadatos"[Chiappe Laverde et al., 2007].

Los principales Objetivos de Aprendizaje que se han de definir son los siguientes:

- **OA1:** Comprender los conceptos básicos del sistema de gestión de memoria virtual, el concepto de paginación y los algoritmos de reemplazo de página, entre otros.
- **OA2:** Comprender y aplicar los algoritmos de gestión de memoria virtual, en concreto, algoritmos para la gestión de memoria virtual.
- **OA3:** Analizar la organización de un sistema de gestión de memoria virtual.
- **OA4:** Desarrollar conocimientos sólidos sobre el impacto de rendimiento de los diferentes algoritmos de gestión de memoria.

El diseño del contenido del curso que se va a implementar a modo de presentación, basada en diapositivas, se va a centrar en el temario de la asignatura de Sistemas Operativos impartida en la UAH al grado de Informática. En concreto, se va a desarrollar por cuestiones de tiempo y envergadura del trabajo únicamente el tema correspondiente, titulado, Algoritmos de reemplazo de páginas.

En la parte de la actividad, se va a implementar un simulador de la actividad que se realiza en la asignatura. Esta actividad se presenta en formato tabla, representando las filas los valores de los marcos de memoria[10] de cada instante de tiempo en que se ejecuta el algoritmo. Las columnas representan los conjuntos de marcos de memoria de los que dispone teóricamente el sistema simulado. La parte superior de la tabla, representa la cadena de referencia de los valores que se toman como entrada para los diferentes algoritmos.

4.5. Implementación de la aplicación

Una vez establecido el diseño, en el cual se ha utilizado la arquitectura y el funcionamiento de Unity como base de la que partir, se continúa a la implementación del proyecto. Dicha implementación tendrá como resultado la tarea de desarrollar el entorno virtual. Para mayor simplicidad del desarrollo y siguiendo el diseño establecido en el punto 4.4, se va a exponer el desarrollo de cada una de estas secciones.

En el desarrollo de cada uno de las secciones van a intervenir los diferentes componentes clave pertenecientes a la arquitectura de Unity como son los siguientes: las escenas, *prefabs*, *scripts*, *canvas*

[10]Un marco de memoria es una posición de la Memoria Virtual(MV) que contiene un valor.

y componentes de UI, etc. El dominio y la comprensión previa de estos componentes es clave para su creación y utilización mediante los conocimientos adquiridos a través, principalmente, de la documentación oficial[11] de la plataforma de Unity.

Para comenzar la implementación y poder crear así un proyecto en la plataforma de Unity en la versión 2018.4.9f1 LTS (*Long Term Support*) se debe crear el proyecto desde el lanzador[12] que proporciona Unity.

Una vez nos hemos situado en la ventana del editor Unity, se llevará a cabo la importación de los diferentes paquetes y assets (avatar, scripts, etc.) a la sección del proyecto. En esta fase, se sitúa tanto la jerarquía de carpetas del proyecto como las carpetas con las que se van a desarrollar el entorno virtual (véase figura 4.28).

Figura 4.28: Jerarquía de carpetas del proyecto.

Escenas

Una vez disponibles la jerarquía (véase figura 4.29) y los *assets*, se dará comienzo al desarrollo del entorno mediante la realización de las diferentes escenas diseñadas previamente en el punto 4.4. Para ello, desde la interfaz de Unity que se encuentra situada en el panel de jerarquía (*hierarchy*), siguiendo las indicaciones anteriores del punto 4.2. Posteriormente, hay que guardar todo lo realizado en la carpeta *Scenes* mediante la opción *Save Scene as* para así poder almacenarlas en el directorio deseado.

Una vez creadas las escenas, se debe ajustar la configuración de la compilación del proyecto. Es esencial incluir las escenas que han sido creadas. Lo primero que hay que hacer es añadir dichas escenas al menú *File*, haciendo clic sobre la opción de configuración de compilación (*Build Settings*) de la figura 4.30.

[11] Documentación Unity: https://docs.unity3d.com/2018.4/Documentation/Manual/index.html

[12] Unity Hub es el lanzador de la aplicación del editor de Unity. Permite crear nuevos proyectos, acceder a los ya creados, permite controlar las versiones del editor de Unity instaladas y además incluye una sección de aprendizaje.

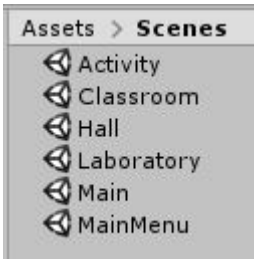

Figura 4.29: Jerarquía de las escenas.

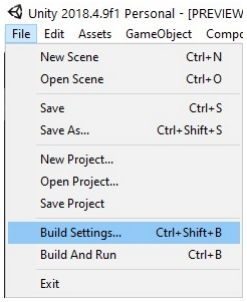

Figura 4.30: Menú desplegable *File*.

A continuación, aparecerá desplegado un panel (véase figura 4.31) con las diferentes opciones de compilación, una lista de ajustes que variará según la plataforma seleccionada y otra lista con las escenas que estén incluidas en la compilación. En este panel se mostrarán aquellas escenas añadidas en el proyecto que se deseen incluir a la hora de generar el proyecto en la plataforma de Unity. Del mismo modo, también existe la posibilidad de seleccionar alguna otra plataforma distinta compatible con Unity. Algunas de estas podrían ser web (HTML5), videoconsolas como PlayStation 4 o Xbox, que son a día de hoy las más populares, o como una aplicación para Windows o Mac. Por esta razón, se puede ver de manera clara que Unity posee soporte multiplataforma, y para llevar a cabo nuestro proyecto esta característica nos aporta muchas posibilidades de exportación. Al igual que se puede hacer uso de otra plataforma, también se pueden ajustar diferentes parámetros en cada una de ellas. Por último, podemos generar el proyecto compilando todas las escenas cargadas en la lista a la plataforma seleccionada, según las opciones que se hayan marcado. Del mismo modo, tendremos la posibilidad de construir el proyecto y abrirlo en la opción de reproducción en su plataforma de destino.

Para poder añadir las escenas al listado, se pueden arrastrar desde la carpeta *Scenes* del inspector del proyecto o bien, cargarlas una a una mediante el botón de añadir escena (*Add Open Scenes*). El orden en el que se van a añadir las escenas va a ser muy relevante, teniendo en cuenta que la primera, que va a tener en el indicador 0, va a ser la primera a la que se accederá al iniciar la ejecución del proyecto.

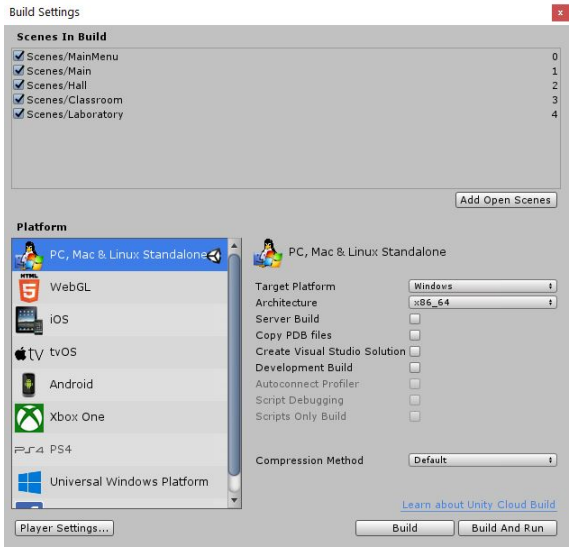

Figura 4.31: Menú desplegable ajustes de proyecto.

Avatar

En los entornos virtuales de aprendizaje se permite crear representaciones digitales del individuo dentro de los mundos virtuales que se crean. A través de los avatares, el usuario se introduce en el mundo virtual.

Los avatares son representaciones digitales, normalmente, con forma de humanoide que permiten situar de forma figurada al usuario que accede al mundo virtual. Los humanos responden a los avatares de forma similar a como responden a otros humanos en la realidad [Bailenson et al., 2001].

Como indica la definición, el avatar tiene aspecto humanoide (véase figura 4.32), que simula el aspecto de un alumno tipo. Este componente se ha importado como un *prefab* en el proyecto.

Figura 4.32: Avatar.

La jerarquía del objeto avatar, de tipo *prefab*, es la que se muestra en la figura 4.33.

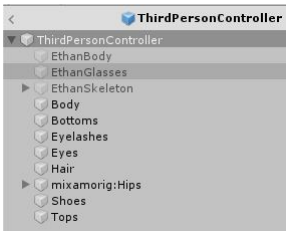

Figura 4.33: Jerarquía de componentes del Avatar.

A continuación, se muestra el código que permite captura las órdenes del usuario y que permite que el avatar se pueda mover por el entorno virtual.

```
1  private void FixedUpdate()
2      {
3          // read inputs
4          h = CrossPlatformInputManager.GetAxis("Horizontal");
5          v = CrossPlatformInputManager.GetAxis("Vertical");
6          bool crouch = false;
7
8          // calculate move direction to pass to character
9          if (m_Cam != null)
10         {
11             // calculate camera relative direction to move:
12             m_CamForward = Vector3.Scale(m_Cam.forward, new Vector3(1, 0, 1)).normalized;
13             m_Move = v*m_CamForward + h*m_Cam.right;
14
15         }
16         else
17         {
18             // we use world-relative directions in the case of no main camera
19             m_Move = v*Vector3.forward + h*Vector3.right;
20         }
21 #if !MOBILE_INPUT
22                     // walk speed multiplier
23             if (Input.GetKey(KeyCode.LeftShift)) m_Move *= 0.5f;
24 #endif
25
26         // pass all parameters to the character control script
27         m_Character.Move(m_Move, crouch, m_Jump);
28         m_Jump = false;
29     }
```

Código 4.2: Fragmento de código función *FixedUpdate()* correspondiente al avatar.

Mediante el uso de la clase *CrossPlatformInputManager* podemos realizar la lectura de los valores de entrada correspondientes según el eje de movimiento del avatar.

El funcionamiento del código 4.2, que representa la función *FixedUpdate()*, se ejecuta un número de veces fijo en cada iteración del bucle principal que proporciona Unity. Dentro de este código, se

realiza una lectura de las variables **Horizontal** y **Vertical** con las cuales se calcula el vector de movimiento que debe seguir el avatar. Este cálculo incluye la rotación del avatar respecto a su eje. En las acciones posibles, el avatar podría realizar la función de agacharse aunque por motivos de simplicidad se ha desactivado esta función.

Creación de presentación

La presentación se sitúa en la escena principal como se ha visto en el punto 4.4 del diseño. Es una de las partes clave en el desarrollo del proyecto que involucra multitud de componentes y requiere la dedicación de más recursos en su desarrollo. Se ha trabajado con componentes diseñados exclusivamente para el desarrollo de este objeto así como, del mismo modo, se han creado componentes prefabricados que se han usado en múltiples ocasiones en la creación de la presentación.

El principal objetivo de la presentación es facilitar al alumno la adquisición de los conocimientos teórico/prácticos de una forma visual, su interacción y potenciar el aprendizaje intuitivo. La exposición de estos conocimientos se realiza por medio de objetos que simulan diapositivas creadas sobre el componente *Canvas*.

Para acceder a la presentación, el alumno debe acceder a la escena principal y seleccionar entre los diferentes subtemas que hay disponibles en el menú de temario. En esta ocasión, se ha llevado a cabo el desarrollo del contenido para el tema *Algoritmo de reemplazo de páginas*. Al seleccionar esta opción en el menú, se oculta el menú y se muestra la presentación de la figura 4.34.

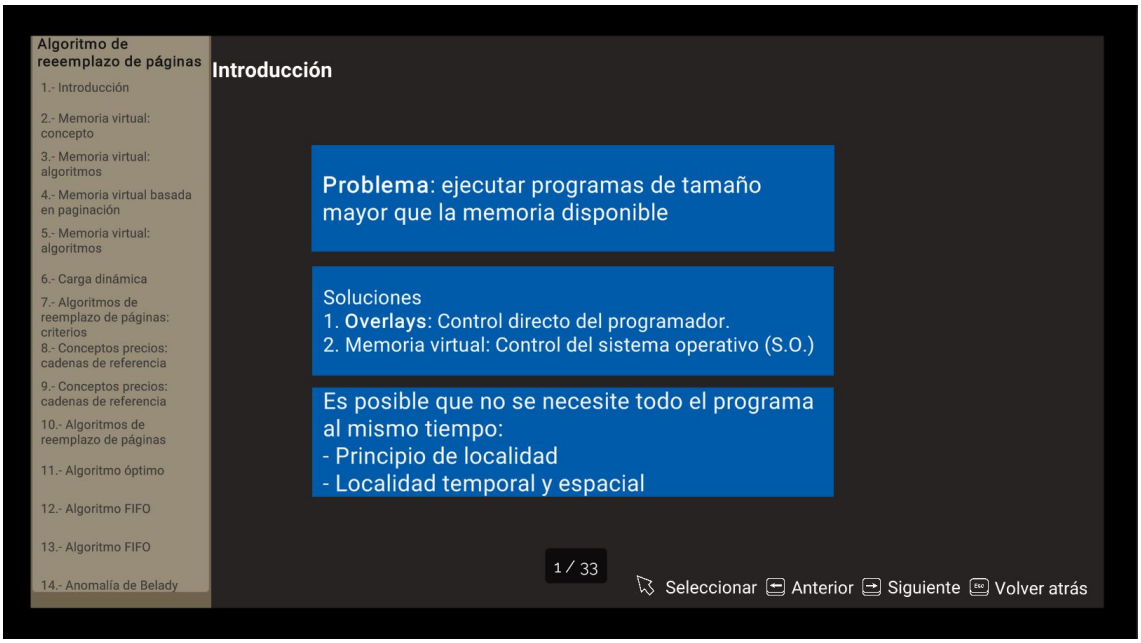

Figura 4.34: Visualización de la presentación.

En la figura anterior (figura 4.34) puede observarse que la interfaz posee varias zonas o áreas claramente diferenciadas. En el margen superior izquierdo, se sitúa el título del tema de la presentación. Debajo de este, se sitúa el índice con los contenidos (subtemas) que componen la presentación y que, seleccionando cada uno de ellos, se realiza una animación de transición hasta esa diapositiva. En la sección central y hasta el margen derecho se sitúa la diapositiva en la que se sitúa actualmente la presentación. La diapositiva tiene una estructura sencilla e intuitiva, en la parte superior izquierda, el título de la diapositiva y en la parte central el contenido de la misma; En la parte inferior se ubica información relevante al contador de dispositivas; el número de la dispositiva actual y el número total de diapositivas que componen la presentación. En la parte más inferior de la pantalla y situado en la parte derecha, se muestra la ayuda al usuario con las acciones disponibles al usuario permitiendo el uso de las teclas que activan esa opción o bien, mediante el cursor del ratón.

La estructura de la presentación está formada por los siguientes componentes:

- **MVAlgoritmoReemplazo:** Este *GameObject* vacío es el padre de todos los componentes diapositiva. Cada una de diapositivas se conforman partiendo de la base del componente *prefab* creado para tal fin, llamado *Diapo*.

  - **Diapo:** Componente *prefab* dependiente de *Canvas* y de la capa UI. Este *prefab* ha sido creado mediante la siguiente jerarquía (véase figura 4.35):

Figura 4.35: Estructura de la componente *prefab Diapo*.

    - **Diapo:** Componente padre del *prefab*. Contiene el subcomponente de tipo *Canvas renderer* para poder mostrar contenido en el *canvas* de Unity. También cuenta con un componente *Image* que permite representar contenido. Incluye el *script Slide*, que permite activar y desactivar la diapositiva. Finalmente, el componente *Animator* permite al *prefab* poder ser animado mediante la máquina de estados del *Animator Controller*.
    - **Título:** Representa el título de la diapositiva mediante un componente de tipo *text* que se sitúa en la parte superior izquierda de la diapositiva.

    - **Image:** Cuenta con un componente de tipo *Image* que representa un rectángulo en la pantalla en el que se muestra un texto, que es hijo de este componente mediante un componente *Text Mesh Pro UGUI*.

- **Panel:** Este componente de tipo panel contiene el sistema de navegación de la presentación, el subcomponente **Navegación**, un objeto de tipo *Scroll Rect* que permite generar un panel con scroll vertical y con un listado de la navegación generado automáticamente. El objeto navegación tiene como hijo un componente panel, que es donde se sitúan los elementos del *scroll*. Inicialmente se sitúa el título de la presentación y un elemento posicionador, un *Gameobject* vacío.

- **ContadorDiapositivas:** Para indicar al alumno el número de diapositivas que tiene la presentación y el número de la diapositiva actual. Se ha utilizado un componente *image* para representar el contador con un rectángulo con un fondo negro y un objeto hijo de tipo text donde se muestran los números de las diapositivas.

- **Button Prompt:** Como sistema de ayuda al usuario, se ha implementado un conjunto de iconos acompañados con un texto explicativo indicando las acciones disponibles para el usuario y el método para hacer uso de ellas.

Script presentación

Una vez el usuario ha seleccionado algún elemento del menú, se carga la presentación desactivando el objeto menú y posibilitando el control de la presentación. En ese momento, entra en acción el *script presentación* el cual gestiona la transición entre la simulación de diapositivas. El *script* (véase código 4.3) *presentación* se encarga de leer las teclas que pulsa el alumno y, de acuerdo a las ayudas proporcionadas al usuario, ejecutar las operaciones asociadas a estas teclas.

```
1  void Start()
2    {
3        // Set contador
4        CargaContador();
5        // presentador
6        presentador = FindObjectOfType<Presentador>();
7    }
8
9    void Update()
10   {
11       Entrada();
12   }
13
14   void Entrada()
15   {
16       // Siguiente diapositiva
17       if (Input.GetKeyDown(KeyCode.RightArrow) && currentSlide < diapos.Length - 1 && action)
```

```
18      {
19          Debug.Log("Siguiente diapositiva");
20          nextSlide();
21          CargaContador();
22          StartCoroutine(startStop());
23      }
24      else if (Input.GetKeyDown(KeyCode.LeftArrow) && currentSlide > 0 && action)
25      {
26          Debug.Log("Anterior diapositiva");
27          previousSlide();
28          CargaContador();
29          StartCoroutine(startStop());
30      }
31      else if (Input.GetKeyDown(KeyCode.Escape))
32      {
33          currentSlide = 0;
34          CargaContador();
35          presentador.Menu();
36      }
37  }
38
39  void CargaContador()
40  {
41      contador.text = currentSlide + 1 + " / " + diapos.Length;
42  }
```

Código 4.3: Fragmento de código del *script presentacion*.

La implementación del *script* 4.3 se basa en la invocación de un método desde la función *Update()*. Esta función se ejecuta por Unity en cada *frame* lo que permite que se compruebe la entrada de usuario de forma constante a lo largo de la ejecución de la escena *main*, donde se sitúa la presentación. Desde la función *Update()* se llama al método *entrada()*, este método lee el valor *Input* que permite realizar la lectura de las diferentes entradas de datos, en concreto usamos el método *GetKeyDown()* que devuelve un valor *true* en el *frame* en el cual el usuario empieza a pulsar la tecla que indicamos por parámetro y que vamos a evaluar si ha sido pulsada por el alumno. En este caso, las teclas las identificamos por *KeyCode.RightArrow* para pasar la siguiente diapositiva, *KeyCode.LeftArrow* para pasar a la diapositiva anterior y la tecla *KeyCode.Escape* para salir de la presentación y volver al menú del temario donde el alumno podrá volver a seleccionar otro tema o podrás salir si lo desea.

Para evitar problemas en la transición entre diapositivas, se ha implementado un sistema de bloqueo basado en la invocación de la corrutina[13] *startStop()*. Esta es llamada cada vez que el usuario pulsa la tecla de siguiente o anterior diapositiva. La corrutina que controla la ejecución de la transición entre diapositivas impidiendo un posible solape entre las transiciones al avanzar o retroceder de diapositiva.

[13]Una corrutina es una función que tiene la habilidad de pausar su ejecución y devolver el control a Unity para luego continuar donde lo dejó en el siguiente fotograma [Technologies, 2016].

```
1 IEnumerator startStop()
2    {
3        action = false;
4        yield return new WaitForSeconds(0.35f);
5        action = true;
6    }
```

Código 4.4: Fragmento de código de la corrutina *startStop()*.

El funcionamiento de la corrutina (véase código 4.4) es sencillo: mediante *action*, un valor booleano que activa o desactiva el control del método *entrada()*, se controla la posibilidad de avanzar o retroceder en el avance de las diapositivas de la presentación. Mediante la línea 4 se añade un retardo (*delay*) de 350 milisegundos durante el cual la variable *action* tiene el valor *false* y se bloquea el cambio de diapositiva. Una vez transcurrido ese tiempo de bloqueo, se vuelve a asignar el valor *true* a la variable *action* permitiendo así al usuario controlar de nuevo la ejecución del próximo *frame*.

Máquina de estados

Una vez explicado el funcionamiento y componentes de la presentación, a continuación se detalla el funcionamiento de la máquina de estados que permite realizar transiciones entre diapositivas en la presentación.

La máquina de estados se encuentra dentro del objeto *animator controller*. Este objeto se encarga de gestionar las animaciones ligadas a los objetos que tengan un componente del tipo *Animator* vinculado al *controller* de *Presentacion*. Para acceder a la vista del panel que a continuación se puede observar en la figura 4.36, se debe seleccionar el objeto *animator controller* y acceder a la pestaña *animator*.

Como podemos observar en la figura 4.36, aparecen dos áreas claramente diferenciadas:

- **Parámetros (*parameters*):** Estos valores permiten activar o desactivar las transiciones entre los estados. Cada uno de estos parámetros implementa una acción; retroceder diapositivas o avanzar diapositiva y lo correspondiente en este contexto.
- **Capa base (*base layer*):** Representación gráfica de los diferentes estados y sus transiciones. Cada transición se ejecuta con la activación de su correspondiente parámetro. En este caso, se configura como *trigger* un valor booleano que se reinicia desde el controlador cuando se efectúa la transición.

Los parámetros que controlan la máquina de estados se activan y desactivan desde el *script*

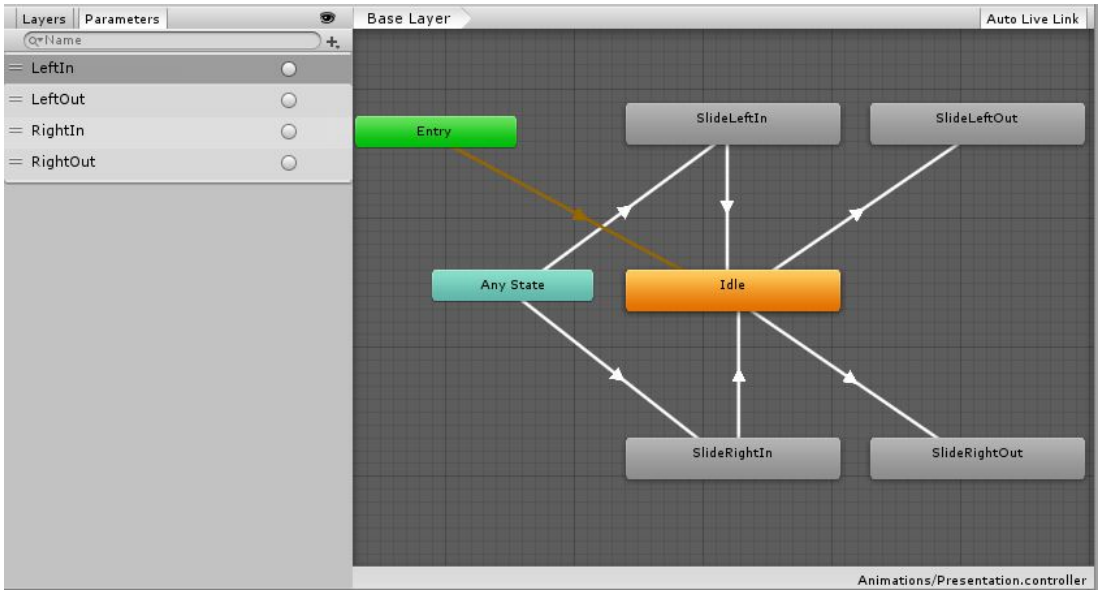

Figura 4.36: Máquina de estados *presentacion*.

presentacion cuando el usuario acciona la tecla correspondiente. Desde este *script*, se incrementa el número de la diapositiva anterior según corresponda y se introduce la nueva diapositiva desde el lado contrario mediante las animaciones activando y desactivando los *triggers* anteriormente mencionados.

Los *triggers* se manejan mediante una gestión de comandos (*scripting*). Esto implicará activar los parámetros según se desee accionar una transición u otra. En el código 4.5 se muestra un fragmento de la función *NextSlide()* con el procedimiento que se usa para incrementar el número de diapositiva y transición entre diapositivas.

```
1 private void nextSlide()
2   {
3       diapos[currentSlide].GetComponent<Animator>().SetTrigger("RightOut");
4       currentSlide++;
5       diapos[currentSlide].SetActive(true);
6       diapos[currentSlide].GetComponent<Animator>().SetTrigger("RightIn");
7   }
```

Código 4.5: Fragmento de código función *NextSlide()*.

1. Se activa el *trigger RightOut*, que pasa del estado de reposo (*idle*) al estado *SlideRightOut* para la diapositiva actual. Este último estado tiene asociada la animación homónima que desplaza la diapositiva fuera de la pantalla y antes de finalizar, mediante un evento, invoca a la función *slideDisable()* que desactiva el objeto correspondiente a esa diapositiva.

2. Se incrementa el contador de diapositivas en uno para dar paso a la siguiente diapositiva.

3. Se activa la siguiente diapositiva, que ya es la diapositiva actual. Esta acción la hace visible e interactivo.

4. Finalmente, se activa el parámetro *RightIn* para cambiar del estado *idle* a la animación *RightIn*, que desplaza el nuevo objeto de tipo diapositiva a la posición final desde el margen derecho.

Del mismo modo, se ha implementado la función *previousSlide()* que permite realizar la acción de retroceder a la diapositiva anterior. Para permitir que el usuario pueda navegar mediante el sistema de navegación, se ha implementado otra función, *loadSlide()*, que permite ir hasta el número de diapositiva pasado como parámetro. Estas funciones implementan operaciones similares de activación de *triggers* para ejecutar la transición de las diapositivas.

Generación de navegación automática

El panel de navegación utilizado en la presentación (véase figura 4.37) se ha desarrollado para facilitar al usuario el acceso a cada diapositiva de forma rápida y accesible. Su implementación se basa en una jerarquía de objetos (*prefab* y componentes). La mayor parte de estos elementos se engloban bajo el objeto *MVAlgoritmoReemplazo* en la presentación.

Figura 4.37: Estructura de objetos del panel de navegación.

Los objetos que componen el elemento de navegación son los siguientes:

- **Navegacion:** Componente de tipo *Scroll Rect* en el cual se genera el listado con los nombres de las diferentes diapositivas (véase figura 4.34).
- **Panel:** Componente contenedor, hijo del objeto *Navegacion*, del listado de diapositivas. Su tamaño se ajusta automáticamente según el cálculo del número de nodos hijos, es decir, el número de diapositivas que hay en la *presentacion*. Si el número de elementos es superior al tamaño visible, se permite realizar la acción de *scroll*, es decir, que el panel se desplace en el eje vertical para poder acceder a todos los elementos de la lista.
- **Button:** Se trata de un componente *prefab* que representa un enlace a una diapositiva. Cada uno de ellos contiene el título de la diapositiva. Si el alumno hace clic sobre alguno de ellos, se sitúa el contador en el valor del elemento seleccionado y se produce el cambio de diapositiva. El

prefab de ente componente tiene asociado un componente *animator controller* con diferentes estados y animaciones según los diferentes posibles estados del objeto *button*.

- **Posicionador:** Es un objeto de tipo *Rect Transform* con unas dimensiones específicas ajustadas al tamaño del panel. La función principal de este componente es la de situarse en la posición en la cual se va a ubicar un nuevo botón en el panel. Cuando se añade un elemento *button* en la posición actual del objeto *posicionador*, este se desplazada una distancia igual a su medida más un valor de margen para poder situar un nuevo elemento a continuación del anterior.

Explicación de la máquina de estados de *button*

El *prefab* incluye un componente *animator* vinculado a un *animator controller*. La máquina de estados permite que el objeto *button* pueda realizar distintos cambios mediante transiciones entre animaciones (véase figura 4.38).

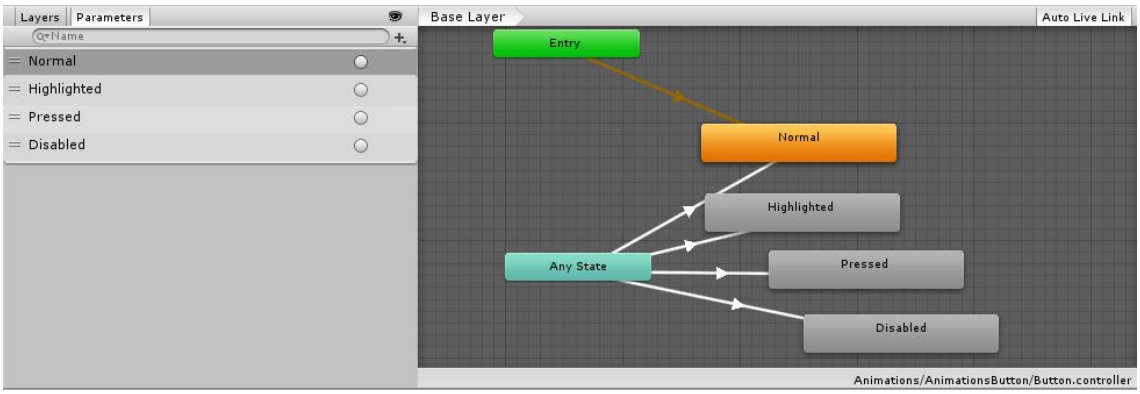

Figura 4.38: Máquina de estados *Button*.

Los diferentes estados se detallan a continuación:

- **Normal:** Es el estado por defecto. No realiza ninguna acción.
- **Highlighted:** Cuando el usuario posiciona el cursor sobre el botón, se detecta el evento y realiza un cambio de estado.
- **Pressed:** Al hacer clic sobre el botón el evento genera un cambio de estado identificando que ha sido presionado.
- **Disabled:** Este estado no permite la selección del botón.

Generador de elementos de navegación

Para automatizar el proceso de creación del menú de navegación, es necesario analizar de forma automática todas las dispositivas de la presentación y obtener el título de cada diapositiva para ser mostrado y enlazado. Este proceso se realiza mediante un *script* llamado *GeneradorJerarquia-Presentacion()* que tiene como parámetro de entrada el objeto que contiene las diapositivas, lo que permite realizar acceso mediante código a cada una de las diapositivas, que son elementos hijos del objeto padre de tipo *presentacion*.

Este *script* recibe varios parámetros (véase figura 4.39) necesarios para su funcionamiento. A continuación, se pasa a detallar el código:

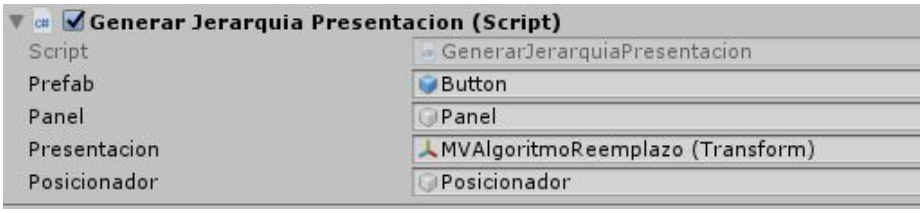

Figura 4.39: Parámetros del *Script GenerarJeraquiaPresentacion*.

- **Posicionador:** Objeto encargado de desplazarse por el panel, del cual se pueden obtener los datos de su posición para instanciar el nuevo botón a generar. Este objeto se desplaza de nuevo para dejarlo preparado para la siguiente ejecución del bucle de ejecución del *script*.

- **Presentacion:** Este objeto es la clave para poder recorrer todos sus hijos, las distintas diapositivas que forman la presentación, obteniendo el valor del título de cada una de ellas para poder insertar ese valor en el nuevo botón que conforma el panel navegación.

- **Panel:** El componente principal de la navegación es el panel donde se sitúan todos los elementos que representan las distintas diapositivas de la presentación. A medida que el bucle va recorriendo cada uno de los hijos del objeto presentación, se obtiene el título de la diapositiva. En cada iteración del bucle generador, sé instancia un objeto *button* de tipo *prefab*. Cuando sé instancia este objeto se modifica mediante un método el título del *button*. Al instanciar este objeto, se sitúa en la ubicación del elemento posicionador. Posteriormente, se traslada el posicionador al siguiente valor de posición a la espera del nuevo ciclo de ejecución del bucle.

Teniendo en cuenta la tarea que desempeña cada uno de los parámetros que recibe el *script*, queda claro el desempeño de tareas del mismo. Primeramente, sé instancia el *GameObject* mediante la función *Instantiate()*.

La función 4.6 recibe varios parámetros: el objeto a instanciar, una posición de tipo *Vector3* con la posición a instanciar, la rotación deseada de tipo *Quaternion* y el objeto padre en la jerarquía del que depende este nuevo objeto.

```
1 public static Object Instantiate(Object original, Vector3 position, Quaternion rotation, Transform parent);
```

Código 4.6: Definición de la función *Instantiate()*.

Después de instanciar el objeto, se accede al componente *indiceButton* que es el *script* encargado de almacenar los valores que se muestran en el botón. Mediante la función *setSlideNum(i)* sobre el *botonInstance*, editamos el valor de la diapositiva con la que se corresponde respecto a la presentación. Esta asignación permitirá poder seleccionar en la navegación un elemento dentro del panel y al hacer clic, el evento *OnClick* mostrará la diapositiva con el valor correspondiente en la presentación.

Una vez instanciado el objeto y establecido el valor de diapositiva, se genera el texto que se va a mostrar sobre el botón. En la línea 9 del código 4.7 se genera el texto mediante la concatenación en una variable cadena que posteriormente se asigna al valor del componente del tipo *text* del botón.

```
1  private void generaBotones()
2    {
3        float distancia = 0f;
4        print(posicionador.transform.position);
5        for (int i = 0; i < presentacion.childCount; i++)
6        {
7            GameObject botonInstance = Instantiate(prefab, posicionador.transform.position, Quaternion.identity,
                 panel.transform);
8            botonInstance.GetComponent<indiceButton>().setSlideNum(i);
9            string texto = (i + 1).ToString() + ".- " +
                 presentacion.GetChild(i).transform.GetChild(0).GetComponent<Text>().text; // Text ==>
                 TextMeshProGUI
10           botonInstance.GetComponent<indiceButton>().setShowText(texto);
11
12           posicionador.transform.Translate(0, -(Screen.height * 0.06f), 0);
13
14       }
15       posicionador.transform.Translate(0, +(Screen.height * 0.06f), 0);
16
17       distancia = 37f * presentacion.childCount;
18       print(distancia);
19       Debug.Log("Imprime RECT:" + panel.GetComponent<RectTransform>().rect);
20       RectTransform rt = panel.GetComponent<RectTransform>();
21       Debug.Log("panel width = " + rt.sizeDelta.x);
```

```
22        rt.sizeDelta = new Vector2(rt.sizeDelta.x, distancia);
23        Debug.Log("panel width = " + rt.sizeDelta.y);
24        Debug.Log("Imprime RECT MOD:" + panel.GetComponent<RectTransform>().rect);
25    }
```

Código 4.7: Fragmento de código de la función *generaBotones()*.

En la línea 10 del código 4.7 se realiza una llamada a la función *setShowText()*, que ajusta el valor del parámetro de texto para visualizarse en el botón. Este texto es el que podrá ver el alumno en el panel de navegación en la escena *Main*.

Al final del bucle, se desplaza el objeto posicionador de tipo *GameObject* en el eje Y sobre el panel donde está situado (línea 12 del código 4.7). La traslación se calcula en relación con el tamaño de la pantalla multiplicando este valor por un factor de desplazamiento suficiente que permita dejar la suficiente separación entre cada componente botón en el panel.

Una vez finalizada la ejecución del bucle, es decir, recorridos todos los hijos de la presentación y enlazada cada diapositiva con su botón correspondiente, se procede a realizar un ajuste del objeto posicionador siguiendo el desplazamiento de la línea 15 del código 4.7. Llegados a este punto, es necesario ajustar el tamaño del panel según el número de elementos introducidos en el mismo. Si no se realiza este ajuste (ver línea 22 del código 4.7), no se puede acceder a todos los elementos del panel, puesto que la acción de *scroll* se realiza desplazando el panel sobre la máscara que oculta el resto del contenido, no visible este gracias a esta máscara. Por este motivo, esta función posee un valor que cabe mencionar.

Creación de las actividades

Las actividades ligadas a cada algoritmo en el entorno virtual desarrollado son una parte fundamental en el proceso de enseñanza. Con estas, el fin es garantizar la asimilación de los conceptos teóricos mediante la aplicación detallada, en la que el alumno interactúe con el entorno, en un escenario concreto y controlado.

El principal objetivo de la actividad es reforzar el proceso de aprendizaje del alumno mediante la aplicación de los conceptos teóricos en un caso práctico, por medio de la realización de un ejercicio. Después de observar todo el contenido relativo a cada algoritmo de reemplazo, el alumno necesita un elemento con capacidad de interacción que le permita reforzar cada uno de los conceptos abstractos previamente visualizados de forma teórica en la presentación en el entorno virtual. De igual modo, la actividad posee capacidad de detectar errores generados por el alumno y alertar al mismo sobre esta cuestión.

El alumno puede acceder a este tipo de actividades directamente mediante un botón implementado en las diapositivas referentes a cada uno de los tipos de algoritmos de reemplazo a aprender. Esto le permite realizar de forma inmediata y activa, tras la visualización de la teoría, una aplicación de cada algoritmo. Con este método se obtienen mayores garantías sobre el aprendizaje del alumno.

La implementación de la actividad (véase figura 4.23) se realiza mediante una interfaz con un diseño de tipo matriz (una tabla), con filas y columnas, donde se situarán los diferentes bloques de información. Este tipo de actividad tendría un carácter de autoevaluación con el fin de mejorar la asimilación de los conceptos, como se ha mencionado anteriormente.

En la primera fila, o cabecera, se mostrarán los valores de la cadena de referencia generados de manera aleatoria al iniciar la actividad. Estos valores son los que el alumno debe situar sobre los marcos de página, siguiendo los pasos del algoritmo pertinente.

Las filas inferiores, sombreadas en un color gris oscuro, representan los marcos de página donde el estudiante debe situar cada valor de la cadena de referencia según lo requiera el algoritmo de reemplazo.

El desarrollo de esta actividad se ha dividido en las diferentes columnas que se pueden observar en la figura 4.23. Cada una de ellas está compuesta del valor referencia junto con 3 marcos de página y un valor sustituido (en el caso de que exista).

A continuación se detallará la estructura jerárquica de los objetos que forman parte del objeto columna (véase figura 4.40):

- **Columna:** Objeto padre encargado de contener toda la estructura de componentes que proporcionan funcionalidad al componente.

- **Referencia:** La cadena de referencias es una lista de referencias a páginas en memoria [Prieto, 2005]. Muestra el valor generado de forma aleatoria perteneciente a la cadena de referencia. Este valor debe ser asignado de acuerdo al algoritmo de reemplazo de página que se esté aplicando.

- **Marcos:** Este componente representa el número de página en memoria.

- **Resultado:** Este componente desempeña la función de indicar al usuario si se ha producido un fallo de página o si se ha sustituido un valor de un marco con contenido por un nuevo valor obtenido desde la cadena de referencia.

Figura 4.40: Estructura de componentes de cada elemento columna.

La mecánica de funcionamiento es sencilla y muy intuitiva, ya que únicamente el usuario puede resolver la actividad trabajando sobre una columna antes de continuar con la siguiente columna. De esta forma, se simula con la participación activa del alumno la aplicación de los pasos del algoritmo correspondiente. Esto permite que el alumno se autoevalúe tras el aprendizaje de cada algoritmo e interiorizar mejor su funcionamiento.

El primer paso, una vez el usuario ha accedido a la actividad, consiste en situar el primer valor de la cadena de referencia de la primera columna en el primer marco (el situado más cerca del valor de referencia). Esta operación se realizará haciendo clic con el botón izquierdo del ratón sobre el valor de referencia. Ello permitirá después al alumno arrastrar el valor junto al cursor del ratón de forma simulada.

Como segundo paso, el alumno podrá situar sobre un marco de memoria habilitado el valor arrastrado haciendo clic con el botón izquierdo del ratón de nuevo en el marco sobre el que se desea depositar ese valor.

Una vez situado el valor de referencia en la posición deseado por el usuario, el corrector de la actividad comprueba si es correcto este valor en el marco correspondiente. Si no se detecta ningún error, se bloquea la interacción con el usuario en la columna actual y se habilita la siguiente columna. En caso contrario, se bloquea la interacción con los componentes, se activará un mensaje que podrá leer fácilmente el usuario en la pantalla. Este mensaje indicará que ha cometido un error en el valor que ha seleccionado y ubicado en un marco de memoria. Pasado un segundo, se recarga el valor de cada marco de esa columna, ocultando el mensaje de error y permitiendo que el alumno vuelva a poder interactuar con los elementos de la actividad e intentar de nuevo la resolución del ejercicio.

Si el alumno comete dos errores o bien completa todos los marcos de todas las columnas, se interrumpe la actividad y se vuelve a la escena anterior donde se sitúa la presentación. En caso de cometer dos errores, el mensaje indicará que se debe revisar el contenido teórico perteneciente a ese algoritmo. Este cambio de escena lo realiza el componente *GameManager* que almacena el valor de

la diapositiva desde la que partió el alumno para comenzar la actividad, permitiendo continuar la presentación en esa misma diapositiva.

El proceso de aprendizaje combina la presentación teórica con la actividad sin colisionar un área con la otra y permitiendo que el alumno pueda fluir entre la presentación de los temas y las actividades diseñadas para reforzar este aprendizaje.

Corrector de la actividad

Con el fin de ofrecer una retroalimentación de la actividad al alumno, se ha creado un sistema para poder corregir cada uno de los algoritmos. Este sistema se encarga de detectar si el valor de los marcos es el correcto teniendo en cuenta el algoritmo a comprobar y la cadena de referencia.

Cuando el usuario sitúa el valor de referencia de una columna en el marco seleccionado, se comprueba el valor siguiendo los pasos del algoritmo a evaluar. Esta corrección se realiza sobre cada columna antes de dar paso a la siguiente y habilitar su contenido.

El sistema del corrector de la actividad se compone de dos *scripts* principales que intervienen en esta operación de forma conjunta:

- **Director:** Gestiona el cambio entre columnas en la actividad comprobando previamente si hay un error en los valores de los marcos, según el algoritmo de la actividad.
- **Checker:** Implementa los diferentes algoritmos que comprueban los marcos. Este *script* es llamado por el *script Director*.

El funcionamiento del corrector o comprobador de la actividad se basa en varias condiciones de evaluación. La primera es, que una vez la ejecución entra dentro de la función *siguienteColumna()*, se evalúa si el contador de errores es igual a 2. En este caso, se muestra un mensaje de error y se sale de la actividad, retornando a la diapositiva de la presentación desde la que se cargó la actividad. La segunda condición se basa en la comprobación del algoritmo que debe ser ejecutado para comprobar los valores de los marcos de la columna actual. El valor del algoritmo se almacena en el componente *Game Manager* al cual se accederá desde la función 4.8 y mediante una sentencia *switch-case* se ejecuta la función correspondiente al valor contenido en el componente *Game Manager*.

```
1 string value = GameManager.Instance.Algorithm.ToLower();
2       switch (value)
3       {
4           case "optimo":
5               marcosCorregidos = corrector.add_reference(frame_size, frame, reference[contador], contador,
                    columnas.Length, reference);
```

```
6               break;
7           case "fifo":
8               marcosCorregidos = corrector.add_referenceFifo(frame_size, frame, reference[contador]);
9               break;
10          case "lru":
11              marcosCorregidos = corrector.add_referenceLru(frame_size, frame, reference[contador], contador,
                    columnas.Length, reference);
12              break;
13          case "clock":
14              marcosCorregidos = corrector.add_referenceClock(frame_size, frame, reference[contador], contador,
                    columnas.Length, reference);
15              break;
16          case "fifo2":
17              marcosCorregidos = corrector.add_referenceFIfo2(frame_size, frame, reference[contador], contador,
                    columnas.Length, reference);
18              break;
19          case "nfu":
20              marcosCorregidos = corrector.add_referenceNfu(frame_size, frame, reference[contador], contador,
                    columnas.Length, reference);
21              break;
22          default:
23              break;
24      }
```

Código 4.8: Fragmento de código de la función *checkActivity()*.

Cada algoritmo recibe diferentes parámetros que son ajustados en su correspondiente llamada realizada desde la función 4.8. Estas llamadas retornan valores que se almacenan y que serán utilizados para comprobar si es correcta la solución respecto al valor que ha introducido el usuario.

Game Manager

El entorno virtual de aprendizaje se compone de diferentes escenas a las que el alumno va a accediendo de forma transparente mediante cambios de escena. Estas variaciones de escena se pueden realizar cuando el avatar colisiona con un determinado objeto que posee un componente del tipo *collider* ajustado como *trigger*. Esto último detecta la colisión y ejecuta un evento *OnColliderEnter()* que ejecuta el cambio de escena llamando al *SceneManager* de Unity.

Al realizar los cambios de escena, Unity carga la jerarquía de componentes de la nueva escena eliminando todos los datos de la jerarquía de la escena anterior. Este proceso impide poder almacenar ciertos campos de información necesarios para la siguiente escena. Cuando eliminamos toda la jerarquía, se eliminan todos los datos almacenados en tiempo de ejecución, por lo que es necesario utilizar un mecanismo que permita almacenar ciertos valores durante la ejecución del entorno virtual de aprendizaje que serán usados en diferentes escenas.

Para realizar lo anterior, se usa el objeto de tipo *GameManager* que nos permitirá almacenar cierta información durante la ejecución (véase línea 18 código 4.9). Además, este componente no se debe eliminar al cambiar entre las distintas escenas. El principal objetivo de este componente es ofrecer un lugar centralizado y accesible desde todas las escenas en el cual se almacenan todos los datos relativos al estado del entorno virtual y funcionalidades que puedan ser necesarias por parte de cualquier otro componente o *script* desde cualquier escena.

```
1  public class GameManager : MonoBehaviour
2  {
3      #region Singleton
4      private static GameManager _instance;
5      public static GameManager Instance
6      {
7          get
8          {
9              if (_instance == null)
10                 Debug.LogError("Camera behaviour is null dumb!");
11
12             return _instance;
13         }
14     }
15
16     #endregion
17
18     private int slideNum;
19
20     private string algorithm;
21     public string Algorithm { get => algorithm; set => algorithm = value; }
22
23     private bool fromActivity;
24
25     // Start is called before the first frame update
26     private void Awake()
27     {
28         if (_instance != null)
29         {
30             Destroy(gameObject);
31         }
32         else
33         {
34             _instance = this;
35             DontDestroyOnLoad(gameObject);
36         }
37     }
38
39     public void setSlideNum(int slide)
40     {
41         if(slide >= 0)
42         {
43             slideNum = slide;
44         }
45     }
```

```
46
47    public int getSlideNum()
48    {
49        return slideNum;
50    }
51
52    public void setFromActivity(bool value)
53    {
54        fromActivity = value;
55    }
56
57    public bool getFromActivity()
58    {
59        return fromActivity;
60    }
```

Código 4.9: Código del componente *GameManager*.

El componente *GameManager* únicamente debe ser instanciado una vez en tiempo de ejecución, para evitar duplicidad del componente *Game Manager*. Por este motivo, se ha visto necesario implementar un patrón de diseño[14] que permita controlar esto (véase línea 5 de código 4.9).

El patrón de diseño *Singleton* asegura que para una clase dada solamente se pueda instanciar un objeto de dicha clase, es decir, que solo puede definirse una única instancia del objeto. Este patrón requiere una implementación cuidadosa para evitar problemas de concurrencia [Rodríguez Portela, 2019]. En la línea 3 del código 4.9 se define la región del patrón *Singleton* que solo devuelve una única instancia al constructor. De igual modo, se debe ajustar que este objeto no se destruya en la carga de una escena. En la línea 35 del código 4.9 se realiza la llamada a la función interna de Unity *DontDestroyOnLoad()* desde la función *awake()*. Esta última función se ejecuta cuando se comienza a cargar el componente.

4.6. Pruebas

En todo desarrollo software es necesario realizar una fase de pruebas o validaciones. Esta fase tiene como objetivo detectar errores antes de que se produzcan en un sistema en producción, realizar la comprobación de cumplimiento de los requisitos definidos, evaluar el rendimiento del sistema, etc. Debido a la metodología de desarrollo y la forma en la que se desarrolla el proyecto con Unity, las pruebas se realizan cada vez que se lleva a cabo un cambio (pruebas unitarias), ya que fácilmente podemos realizar la validación en el entorno de Unity. A lo largo de las reuniones (véase punto

[14]Un patrón de diseño ofrece una solución para un problema de diseño común, describiendo cómo una sociedad de clases (y objetos) trabajan juntos para resolver ese problema en un contexto particular [Molina, 1999]. Esto es característico del paradigma de programación basado en objetos.

4.4), se ha comprobado cada implementación realizada, evaluando las fases correspondientes del desarrollo tras cada reunión.

Tras la realización de pruebas unitarias a lo largo del desarrollo, las pruebas más relevantes son las de integración. Con esta actividad, se analiza el comportamiento completo del sistema comprobando que todos los componentes y la comunicación entre ellos funciona, probándolos en conjunto.

En la tabla 4.2 se muestran las pruebas más relevantes realizadas en el sistema. Se empieza comprobando el control básico tanto en la interfaz gráfica como del avatar. Posteriormente se evalúa las diferentes acciones del entorno virtual como son los cambios de escena y la detección de colisiones en los elementos de transporte entre escenas.

Acción	**Descripción**
Acceso desde el menú principal	Control mediante ratón y detección de cursor.
Control del avatar	Lectura de elementos de entrada de datos.
Cambio de escenas	Detección de colisión y cambio entre escenas.
Selección ítem menú	Detección de evento pulsación sobre botón.
Acceso actividad	Activación de navegación al pulsar botón.
Detección de errores en columna	Evento disparado al pasar a la siguiente columna.

Tabla 4.2: Pruebas de integración más relevantes.

Finalmente, se comprueban todas las acciones relacionadas con las actividades, tanto de control de la interfaz de usuario como la detección de errores de las actividades con su correspondiente generación de mensajes.

Capítulo 5

Conclusiones

Una vez concluido el desarrollo del entorno virtual construido en Unity para la mejora del aprendizaje sobre el temario de la asignatura de Sistemas Operativos, es el momento de reflexionar sobre todo el proyecto y exponer al lector aquellos aspectos más relevantes de la realización de este trabajo.

Los beneficios que posee un desarrollo en un entorno virtual son múltiples y demuestran un gran potencial de esta técnica. Desde el punto de vista económico, permite reducir costes en la implantación en el proceso de aprendizaje porque se permite el acceso a estos recursos sin importar la ubicación del usuario. Este aprovechamiento digital de los conocimientos ha mejorado notablemente en comparación con hace unos años, cuando las capacidades técnicas eran más limitadas.

La enseñanza tradicional debe adaptarse a estos nuevos recursos para aumentar aún más su valor. Estas nuevas técnicas no buscan suplantar o eliminar la enseñanza tradicional, sino que pueden servir para mejorar ciertos aspectos de la misma, en concreto, en asignaturas con contenidos técnicos y procedimentales. Adicionalmente, donde los escenarios sean de difícil reproducción en la realidad, en asignaturas donde los escenarios virtuales permitan que la salud del alumno no corra ningún peligro o bien en asignaturas cuyos escenarios son de alto coste económico, etc.

En el desarrollo de este proyecto se han alcanzado con éxito los siguientes objetivos planteados el comienzo:

- Se ha llevado a cabo el diseño y el desarrollo de un entorno virtual que simula el entorno real de la Escuela Politécnica de la Universidad de Alcalá.
- Se ha creado un sistema de presentación que simula el comportamiento de un conjunto de diapositivas. Esto se basa en los componentes del entorno Unity, con el cual se han aplicado

los contenidos de la asignatura de Sistemas Operativos impartida en la Escuela Politécnica.

- El desarrollo de un avatar que permite al usuario poder interactuar con el entorno virtual de una forma intuitiva y simulando las acciones que se ejecutarían en el entorno real.
- La creación de actividades interactivas donde el alumno pueda reforzar los conocimientos teóricos adquiridos durante el curso.
- Se han creado un conjunto de códigos implementados en el entorno de desarrollo de Unity que permiten la operatividad del entorno virtual en todas las funcionalidades necesarias.

Finalmente, este Trabajo Final de Grado ha permitido al autor del mismo adquirir nuevos conocimientos y aplicar las experiencias previas en el desarrollo del software y en el diseño cumpliendo así el objetivo final de este tipo de trabajo. Del mismo modo, se han mejorado las capacidades de trabajo autónomo del estudiante y la capacidad de investigación y análisis.

Capítulo 6

Líneas de trabajo a futuro

Tras finalizar el desarrollo y las pruebas del entorno virtual de aprendizaje, se resaltan las siguientes líneas de trabajo que se podrían realizar en el futuro a partir del presente Trabajo Final de Grado:

1. **Optimización de los recursos.** Aún habiendo desarrollado el proyecto de forma iterativa, con optimizaciones en cada iteración, es posible mejorar aún más el rendimiento del entorno virtual y disminuir el consumo de los recursos. Adicionalmente, se podría optimizar el consumo de CPU por el entorno virtual de aprendizaje para ser ejecutado en equipo con capacidad de cálculo menor, como puede ser una *tablet*, un teléfono inteligente (*smartphone*) o un ordenador personal con unos recursos *hardware* más limitados. Esto se puede realizar reduciendo la calidad de las texturas o eliminando el cálculo dinámico de luces.

2. **Sistema de carga de cursos y temario.** Otra línea de trabajo futuro muy interesante sería crear una funcionalidad para la carga de un curso/temario. Este sistema incrementaría la potencia del entorno virtual permitiendo generar objetos de aprendizaje fácilmente y, por tanto, facilitando la labor del docente en la gestión de nuevos cursos o temario. La futura funcionalidad haría uso del sistema de elementos prefabricados de Unity junto con los *prefab* ya creados y, quizás, mediante un archivo JSON[1] o YAML[2], se crearía el curso con tan solo subir un archivo al sistema. Mediante un *script*, sería posible crear cada tipo de *prefab* necesario para cumplir con la estructura presente en el archivo de configuración importado.

3. **Ampliación de actividades.** Al hilo del punto anterior, la generación de actividades de distinto tipo como test multiopción, completar una definición o completar un diagrama, puede

[1] Los ficheros JSON almacenan estructuras de datos y objetos simples en formato *JavaScript Object Notation* (JSON), un formato estándar para el intercambio de datos.

[2] Un archivo YAML se trata de un formato de serialización, es decir, exportar valores de un objeto a un formato legible por un humano.

ser muy valiosa para este entorno. Esta alternativa requeriría de un sistema elaborado a partir de elementos prefabricados que fuesen creados según la estructura de un fichero determinado generado por el docente o por una segunda aplicación que pudiera convertir archivos PDF a archivo JSON para importar el contenido.

4. **Implementación de SCORM.** La mayoría de LMSs utilizan los sistemas SCORM para poder evaluar el aprendizaje de los alumnos. Esta línea de trabajo permitiría vincular los objetos de aprendizaje creados y los futuros a los objetivos de aprendizaje definidos en el SCORM. De esta manera, el docente podría analizar mejor las dificultades de la mayoría de los alumnos y centrar sus esfuerzos en aquellas áreas o conceptos que representen mayor dificultad para el mismo.

5. **Definición de un sistema de evaluación.** Tanto si es una enseñanza *online*, *blended*[3] o presencial, este entorno virtual podría ser el punto central de interconexión entre los alumnos y el docente. La generación de evaluaciones para los alumnos a veces genera problemas a los docentes; estos tienen que crear varios exámenes, con similar dificultad, etc. Incorporando un módulo de generación de exámenes y de corrección, el docente podría eliminar esos problemas. Además, el sistema podría generar exámenes más equitativos e incluso adaptados a posibles dificultades o limitaciones del alumno en el aprendizaje.

6. **Internacionalización de la interfaz.** La posibilidad de implementar un sistema de selección de idioma para que el usuario pueda elegir. Generalmente esto se implementa con tablas de traducción que tienen múltiples referencias y la palabra en múltiples idiomas. Mediante un índice se accede a los valores deseados que se seleccionan según el idioma elegido.

[3]Este tipo de aprendizaje combina el e-learning con encuentros presenciales tomando las ventajas de ambos tipos de aprendizajes

Parte III

Apéndices

Apéndice A

Pliego de condiciones

El pliego de condiciones establece los requisitos mínimos que debe cumplir el proyecto de forma general y las especificaciones técnicas. A continuación, se definen cada uno de estos requisitos basándose en las necesidades del proyecto y las cualidades técnicas del mismo.

- **Pliego de condiciones generales**
  - El proyecto debe poder ser ejecutado en un PC con una resolución estándar *HD* o *full HD*
  - El rendimiento de la aplicación irá ligado a la plataforma y dependerá de la potencia de cómputo del dispositivo elegido.
  - Para la ejecución no es necesarios que el PC donde se ejecute la aplicación tenga acceso a la red.
  - A la hora de ejecutar la aplicación no será necesario instalar ningún software de terceros.
- **Pliego de especificaciones técnicas**

  Requisitos del sistema de Unity Editor (Requisitos mínimos).
  - Sistema operativo
    - Windows: 7, 8 y 10, solo versiones de 64 bits.
    - macOS: Capitan 10.11+
    - Ubuntu 12.04+
  - Procesador de 64 bits con soporte del conjunto de instrucciones SSE2.
  - API de gráficos: Se necesita una tarjeta gráfica con capacidades DX10 (modelo de sombreado 4.0).

- Requerimientos adicionales Windows: los controladores deben ser admitidos oficialmente por el proveedor de hardware.
- Requerimientos adicionales macOS: los controladores deben ser compatibles oficialmente con Apple.

Apéndice B

Presupuesto

Este apartado contiene los detalles económicos relativos al desarrollo del proyecto suponiendo la intervención de diferentes perfiles técnicos en las diferentes fases y tareas necesarias para completar el proyecto. El desglose de los perfiles necesarios sería el siguiente:

- **Analista**: Este perfil desempeña funciones relacionadas con el análisis del proyecto:
  - Realizar el análisis de casos de uso de los diferentes actores que van a hacer uso del entorno virtual.
  - Evaluación de requisitos (hardware y software) del proyecto.
  - Realizar los diagramas de la fase de diseño.
  - Se asume un salario bruto de 20€/h.
- **Jefe de proyecto**: Es la persona encargada de dirigir el proyecto. Tendrá que desempeñar las siguientes funciones:
  - Gestionar a los empleados.
  - Realizar un análisis de los resultados obtenidos.
  - Se asume un salario bruto de 25€/h.
- **Programador**: Este perfil desempeña tareas centradas en el área técnica, en la codificación del proyecto. Su trabajo consiste en:
  - Realizar la implementación de los diferentes códigos (*scripts*) necesarios en el desarrollo.
  - Se asume un salario bruto de 15€/h.
- **Diseñador**: Su tarea consiste en elaborar los elementos gráficos y el diseño del entorno.

- Diseño elementos gráficos.
- Se asume un salario bruto de 15€/h.

- **Probador**: Su tarea consiste en comprobar que la herramienta cumple con todos los requisitos pedidos por el cliente.
  - Realizar todas las pruebas de la herramienta.
  - Se asume un salario bruto de 15€/h.

- **Docente**: Es el encargado de realizar la definición de los objetos de aprendizaje.
  - Diseño temario presentación teórica.
  - Desarrollo de los contenidos didácticos de cada tema.
  - Se asume un salario bruto de 12€/h.

Para la elaboración del presupuesto se estimará el número de horas basándonos en la planificación temporal (véase figura B.1) que marca las diferentes fases del diseño así como la duración en días y la fecha programada para su inicio y fin.

El cálculo de este coste se basa únicamente en la operación entre los días de trabajo del empleado, las horas de una jornada completa y el salario bruto estipulado anteriormente.

Etapas del proyecto		Days	Start	End
Fase de estudio marco teórico		**14.0**	**01-11-19**	**18-11-19**
Estudio de conceptos a virtualizar	6.0		01-11-19	08-11-19
Estudio del concepto de desarrollo Unity	8.0		07-11-19	18-11-19
Fase de análisis de requisitos	**6.0**	**6.0**	**19-11-19**	**26-11-19**
Fase de diseño		**13.0**	**27-11-19**	**13-12-19**
Diseño de espacios virtuales	6.0		27-11-19	04-12-19
Definición de objetos por espacio virtual	3.0		05-12-19	09-12-19
Diseño lógica de programación en C#	4.0		10-12-19	13-12-19
Fase de desarrollo		**27.0**	**16-12-19**	**21-01-20**
Implementación espacios virtuales	16.0		16-12-19	06-01-20
Desarrollo lógica de programación en C#	11.0		07-01-20	21-01-20
Fase de pruebas del entorno		**9.0**	**22-01-20**	**03-02-20**
Pruebas unitarias	8.0		22-01-20	31-01-20
Pruebas de integración	1.0		03-02-20	03-02-20
Fase de documentación		**79.0**	**11-11-19**	**14-02-20**
Manual usuario	9.0		04-02-20	14-02-20
Memoria proyecto	70.0		11-11-19	14-02-20

Figura B.1: Planificación temporal proyecto.

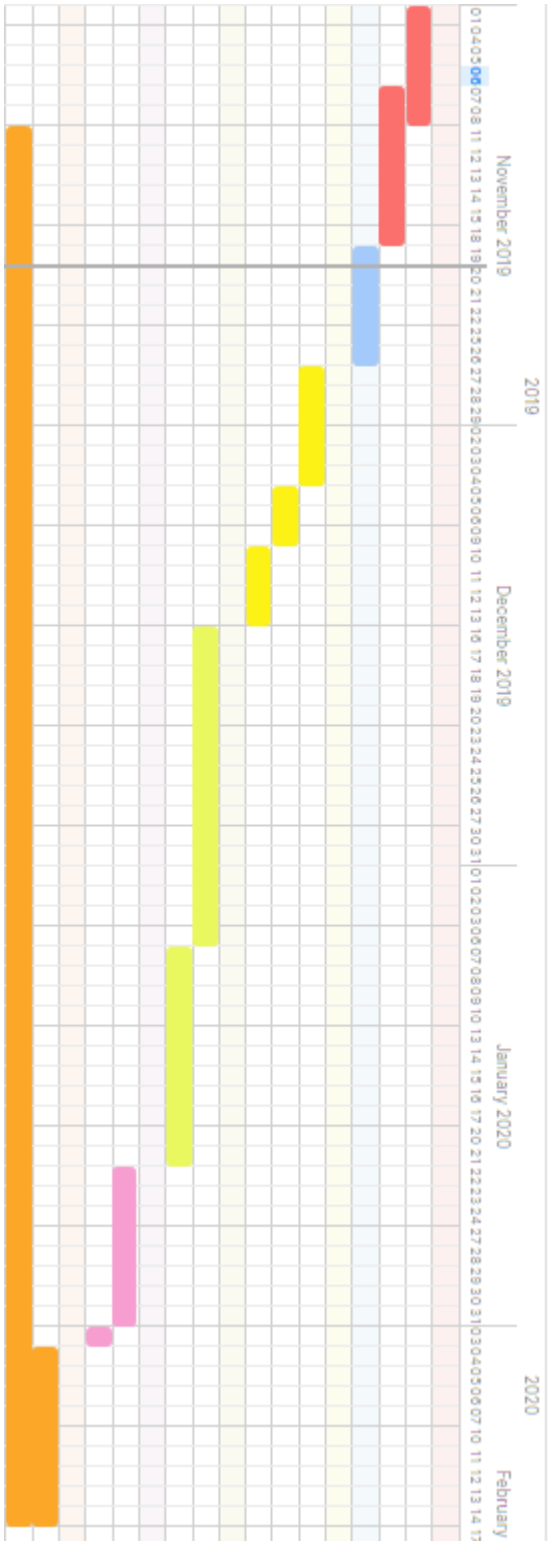

Figura B.2: Estructura de componentes de cada elemento columna.

Parte IV

Bibliografía

Bibliografía

[Alshammari1 et al., 2018] Alshammari1, H., Ali2, M., and Rosli3, S. (2018). Lms, cms and lcms: The confusion among them. *Sci.Int.(Lahore)*, 30:455–459.

[Amorim et al., 2006] Amorim, R., Lama, M., Sánchez, E., Riera, A., and Vila, X. (2006). A learning design ontology based on the ims specification. journal of educational technology and society. *Educational Technology & Society*, 9:38–57.

[Baeini, 2004] Baeini, M. (2004). Scorm primer scorm 2004 primer a (mostly) painless introduction to scorm.

[Bailenson et al., 2001] Bailenson, J., Beall, A., Blascovich, J., Raimundo, M., and Weisbuch, M. (2001). Intelligent agents who wear your face: Users' reactions to the virtual self.

[Beck, 2004] Beck, K. (2004). *Extreme programming explained : embrace change*. Addison-Wesley.

[Boneu, 2007] Boneu, J. (2007). Plataformas abiertas de e-learning para el soporte de contenidos educativos abiertos. *RUSC. Universities and Knowledge Society Journal*, 4:1698–580.

[Branson et al., 1975] Branson, R. K., State, F., and Development, U. S. N. B. O. S. I. C. F. I. S. (1975). *Interservice procedures for instructional systems development*. Center For Educational Technology, Florida State University.

[Ceibal, 2019] Ceibal, P. (2019). Modelo addie — ocw: Rea / laboratorio de objetos de aprendizaje.

[Chiappe Laverde et al., 2007] Chiappe Laverde, A., Segovia Cifuentes, Y., and Rincón Rodríguez, H. Y. (2007). Toward an instructional design model based on learning objects. *Educational Technology Research and Development*, 55:671–681.

[Classcraft, 2021] Classcraft (2021). Classcraft - sistema de gestión de compromiso para educadores k-12.

[Cuenca and UP, 2020] Cuenca, D. and UP, L. (2020). Master class: Arquitectura avanzada en proyectos con unity.

[Desmond and Bahana, 2016] Desmond, M. and Bahana, R. (2016). Bezier vfx plug-in: Improving unity visual effect performance. *2016 1st International Conference on Game, Game Art, and Gamification (ICGGAG)*, pages 1–6.

[Fernández González, 2012] Fernández González, J. (2012). Exaforo.com: Tu foro de informática.

[García, 2009] García, M. M. (2009). Análisis de sistemas - tema 2 - modelos de proceso del software. contenidos contenidos.

[García, 1997] García, V. M. (1997). *Informática, información y Comunicación*. Cáritas Española.

[Jeffries, 2004] Jeffries, R. (2004). *Extreme programming adventures in C#*. Microsoft Press.

[Kahoot!, 2019] Kahoot! (2019). Kahoot! — learning games — make learning awesome!

[L.Broderick, 2001] L.Broderick, C. (2001). Instructional systems design: What it's all about.

[Levinson, 2013] Levinson, W. A. (2013). *Lean management system LMS:2012 : a framework for continual lean improvement*. Crc Press.

[MaxSchlutter, 2013] MaxSchlutter (2013). The psychology of games.

[Molina, 1999] Molina, J. (1999). Ana arquitectura para una herramienta de patrones de diseño.

[Párraga, 2010] Párraga, J. C. (2010). Una propuesta de modelado del estudiante basada en ontologías y diagnóstico pedagógico-cognitivo no monótono.

[Poveda Criado et al., 2014] Poveda Criado, M. , Tuset, T., and del Carmen, M. (2014). Mundos virtuales y avatares como nuevas formas educativas. *Historia y Comunicación Social*, 18.

[Pressman, 2010] Pressman, R. S. (2010). *Ingeniería del software: un enfoque práctico*. Mcgraw-Hill.

[Prieto, 2005] Prieto, S. S. (2005). *Sistemas Operativos*. Universidad de Alcalá, 2nd edition.

[Quiroz and Salvat, 2011] Quiroz, J. S. and Salvat, B. G. (2011). *Diseno y moderacion de entornos virtuales de aprendizaje, EVA*. Uoc.

[Robles and Ángel Gallardo Vigil, 2013] Robles, A. S. and Ángel Gallardo Vigil, M. (2013). Entornos virtuales de aprendizaje: Nuevos retos educativos. *Etic@net: Revista científica electrónica de Educación y Comunicación en la Sociedad del Conocimiento*, 13:3.

[Rodríguez, 2014] Rodríguez, M. (2014). Metodologías ágiles en proyectos de "no-desarrollo".

[Rodríguez García, 2006] Rodríguez García, D. (2006). Metodologías en la ingeniería del software métodos orientados a objetos.

[Rodríguez Portela, 2019] Rodríguez Portela, A. E. (2019). Patrón de diseño singleton. *Departamento Educación Virtual*.

[rstals, 2021] rstals (2021). Unity-scorm-integration-kit/scorm test application guide.pdf at master · rstals/unity-scorm-integration-kit.

[SANABRIA, 2021] SANABRIA, E. (2021). Modelo de desarrollo en cascada – comparasoftware.

[Stellman and Greene, 2014] Stellman, A. and Greene, J. (2014). *Learning agile*. O'reilly.

[Technologies, 2016] Technologies, U. (2016). Unity - manual.

[Technologies, 2021] Technologies, U. (2021). Unity - manual: Glossary.

[Teixes, 2014] Teixes, F. (2014). *Gamificacion : fundamentos y aplicaciones*. Uoc.

[Unity, 2016] Unity (2016). Unity - manual: Execution order of event functions (orden de ejecución de funciones de evento).

[Zah et al., 2010] Zah, W., Ali, W., Wong, S., and Luan (2010). Factors influencing students use a learning management system portal: Perspective from higher education students.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá