



VIRTUAL-CAV®



MEMORIA DE VIRTUAL - CAV

Trabajo de Fin de Grado en Comunicación Audiovisual (Doble Grado)

Facultad de Comunicación

Autores: García Guardia, Santiago
Páramo Pérez, Borja Jesús

Tutor: José Luis Navarrete Cardero

Curso 2017/2018

Agradecimientos

Dedicamos este pequeño apartado antes de comenzar a analizar el videojuego para agradecer a todas esas personas que creyeron en nosotros desde el principio.

A nuestra familia, por tantas tardes aguantando nuestras conversaciones eternas por hangouts. A nuestros amigos, por haber tenido que probar Virtual-CAV infinidad de veces encontrando todos los fallos. Y, por supuesto, a nuestro tutor José Luis Navarrete Cardero, por enseñarnos y mostrarnos los primeros pasos en el desarrollo de este proyecto. Esperamos que os guste Virtual-CAV.

ÍNDICE

1. Introducción.....	4
2. Objetivos.....	5
3. Desarrollo del videojuego.....	6-94
a. Guion	6-14
b. Guía, Diseño y Programación de las Escenas	15-81
1. Universidad.....	16-39
2. Asignaturas	40-70
3. Transiciones.....	71-74
4. Otros	75-81
c. Recursos	82-91
1. Objetos.....	82-84
2. Imágenes.....	85-86
3. Tipografías.....	87
4. Música y Sonidos Fx	88
5. Menú de pausa	89-91
d. GitLab.....	92-94
4. Conclusiones.....	95
5. Bibliografía.....	96-99
6. Anexos.....	100-142

I- INTRODUCCIÓN

Los estudiantes que tienen interés en cursar un Grado pueden consultar las páginas web de las Universidades y Facultades para encontrar la información que precisen. En concreto, disponen de planes de estudio¹, guías de estudiantes, presentaciones... cuya principal característica es la poca originalidad, en tanto en cuanto todos se presentan en un mismo formato que resulta poco atractivo.

En la sociedad de nuestros días, las nuevas tecnologías copan un espacio predominante. Por ello, resulta necesaria su inserción en todas las disciplinas de la vida, incluida la Universidad, aspecto que ya se ha logrado en múltiples áreas. Sin embargo, esta adaptación no ha llegado a los planes de estudio.

Por ello, proponemos desarrollar estas herramientas a disposición de los futuros estudiantes en un formato que les sea familiar y atractivo: un videojuego, debido a que es la industria que más crece en el plano audiovisual², en gran medida, por el uso de dichos potenciales estudiantes de grados.

¹ Plan de Estudios del Grado en Comunicación Audiovisual (2018). Disponible en

http://www.us.es/estudios/grados/plan_192?p=7 Consultado el 10 de octubre de 2018.

² Berengueras, Josep M. (2018) “La industria del videojuego reclama el mismo trato que el cine y la música” Disponible en: <https://www.elperiodico.com/es/ocio-y-cultura/20180827/videojuegos-cine-musica-cultura-comparativa-7001883> Consultado el 10 de noviembre de 2018

2- OBJETIVOS

2.1. Objetivo principal

- Desarrollar un videojuego que proporcione la información necesaria para resolver las dudas de los futuros estudiantes del Grado en Comunicación Audiovisual.

2.2. Objetivos específicos

- Recrear con el mayor grado de fidelidad la Facultad de Comunicación de la Universidad de Sevilla.
- Desarrollar niveles basados en las asignaturas contempladas en el Plan de Estudios vigente del Grado en Comunicación Audiovisual.
- Generar un proyecto de videojuego no cerrado, sino abierto a futuras ediciones (una por cada Grado impartido en la Facultad de Comunicación y a otros estudios de la Universidad de Sevilla).

3- DESARROLLO DEL VIDEOJUEGO

3.A GUIÓN

Este proyecto se desarrolló en un principio con la idea de abarcar las diez asignaturas de primer año que un alumno tiene que superar para poder avanzar hasta segundo curso.

Sin embargo, por cuestiones de tiempo y de horas de trabajo, se acabó reduciendo al primer cuatrimestre, ya que cumplía la misma función de introducir tanto a los nuevos alumnos como a aquellas personas, que, recién salidas de sus exámenes de la Prueba de Acceso a la Universidad (Selectividad), dudan si cursar o no el Grado en Comunicación Audiovisual de la Facultad de Comunicación de la Universidad de Sevilla.

Por este motivo, el guion del proyecto es tan denso. Sin embargo, se ha ido adaptando a las diferentes necesidades. Ideas planteadas para asignaturas que no se han realizado, se han reutilizado en el desarrollo de Virtual-CAV. El guion original es el siguiente:

"A continuación se recogen las distintas asignaturas que componen el Plan de Estudios del 1º y el 2º curso del Grado en Comunicación Audiovisual:

Primer curso

Primer cuatrimestre

- ***Historia de la Cultura Contemporánea***
 - *BSO: Terminator.*
 - *“Aprobar no forma parte de los Derechos Humanos” y otras citas célebres de Don Antonio Parejo.*
 - *Prueba final: “Escape Room”. Esta actividad consiste en poder abandonar un espacio cerrado (por ejemplo un aula) en un determinado tiempo. Para ello, el jugador, deberá resolver distintos enigmas. En esta asignatura, al alumno se le*

presentará un escenario del que deberá salir mediante la resolución de problemas relacionados con movimientos y corrientes políticos contemporáneos que se estudian en la materia (Ilustración, comunismo, racismo, fascismo, capitalismo...).

- **Tendencias Literarias en la Cultura Contemporánea**

- *En la asignatura, se presentarán los cinco libros de lectura obligatoria:*
 - *Antología poética*
 - *El Retablo de las Maravillas (Albert Boadella)*
 - *Hamelín (Juan Mayorga)*
 - *El Enredo de la Bolsa y la Vida (Eduardo Mendoza)*
 - *Las lágrimas de San Lorenzo (Julio Llamazares)*
- *El nivel comenzará con los cinco libros depositados en una mesa. El jugador seleccionará uno y se abrirá su historia. El player interaccionará para dar continuidad al argumento (en el caso de las obras de teatro y las novelas). En la Antología poética, el jugador deberá completar el puzzle con las palabras que falten.*

- **Teoría de la Imagen**

- *Tipología de planos, tipos de objetivos, profundidad de campo (obturación y diafragma) y radio / códecs serán los temas principales de esta asignatura, la cual estará ambientada en el parque de detrás de la Facultad. Para superar la materia, el jugador deberá diferenciar entre las distintas imágenes en función de los métodos citados anteriormente para su obtención.*

- **Teoría de la Publicidad y las Relaciones Públicas**

- *Anuncio*
- *Un space Shooter para eliminar las marcas rivales.*

- **Tecnologías para la Información Escrita**

- *Se aprueba con interacción con personajes y una cámara.*

Segundo cuatrimestre

Derecho Audiovisual

Recreación de un caso de un juicio en el que entren en conflicto las siguientes cuestiones:

- Derecho al Honor, a la Intimidad y a la Propia Imagen
- Injuria
- Calumnia
- Ley Orgánica de Protección de Datos, etc.

Se hará un “Scape Room” en el que el jugador, para completar el nivel, deberá resolver el caso.

Psicología Social de la Comunicación Audiovisual

El jugador cuando entre en el aula, verá un compañero que se le acerca. El compañero le dirá:

"Esta es la clase de Psicología Social de la Comunicación. Aquí tenemos que aprender a ver más allá del lenguaje de las personas, fijarnos en sus actos o forma de vestir. ¡Además, aprenderemos teorías como la de la retención y muchas cosas más! Shhh, toma asiento que acaba de llegar el profesor.

A continuación, el jugador deberá de sentarse en la silla de al lado del compañero y, acto y seguido, aparecerá el profesor. Se acercará a su mesa y comentará a la clase lo siguiente:

"Hola a todos. Este año tendremos que hacer un ejercicio conjunto por parejas para aprobar esta asignatura. Elegid a un compañero de clase y comenzaremos a trabajar ahora mismo".

El compañero sentado al lado dirá:

"¡Oh, mira qué suerte, podremos trabajar juntos! Ponte conmigo de compañero".

Entonces, el profesor explicará:

"La psicología es la ciencia que estudia los procesos mentales, las sensaciones, las percepciones y el comportamiento del ser humano, en relación con el medio ambiente físico y social que lo rodea. En Comunicación Audiovisual es interesante conocer los detalles más allá de lo que vemos a través de la cámara. El cine se ayuda mucho del lenguaje gestual. La televisión también. Por este motivo, en esta asignatura aprenderéis cosas como la comunicación eficaz, la comunicación en el grupo y organizaciones, la influencia social de la comunicación, los estereotipos, la retroalimentación comunicativa o, incluso, sobre los rumores. Es en este último apartado donde quiero que centréis vuestro trabajo por parejas. Quiero que analicéis la incidencia del rumor en la vida de los grupos y las organizaciones".

Acto y seguido, habrá una escena donde el jugador y el protagonista se pondrán a trabajar. Saldrán 3 preguntas tipo test que el jugador deberá responder:

¿Qué se entiende por rumor?

A: La voz que corre entre el público sin necesidad de ser cierta o no

B: Hechos que siempre están comprobados

¿Puede tener el rumor malas intenciones?

A: Sí, se le conoce como chisme.

B: No

¿Cómo afecta el rumor a la comunicación?

A: No afecta a la comunicación

B: Es una poderosa herramienta social para influenciar a la sociedad de forma anónima e impune

Una vez finalizado el cuestionario, el alumno deberá entregar su trabajo al profesor. Sin embargo, justo de antes de levantarse, su compañero le dirá que prefiere entregarlo él directamente. Entonces se levantarán y le dará el papel al maestro. En principio, el alumno ya ha superado la asignatura.*

Escenario: Clase normal de la Fcom. Pizarra para poner imágenes como el debate de Nixon vs Kennedy. Representación de un aula.

Tecnologías de los Medios Audiovisuales I

El jugador que entre en el aula se deberá sentar en una silla. Cuando lo haga, aparecerá el profesor.

Se acercará a la mesa del profesorado y comentará a la clase lo siguiente:

"Hola a todos. Bienvenidos a Tecnologías de los Medios Audiovisuales I. En esta asignatura trabajaremos el mundo audiovisual desde un enfoque práctico. Aprenderemos a coger las cámaras, a hacer nuestros primeros trabajos en la calle y a editar los vídeos para obtener buenos resultados finales. Esta asignatura tendrá un examen final tipo test, pero el peso mayor de la nota lo tendrán las prácticas. Por este motivo, id pasando uno a uno por este hueco para salir aprender, a modo de introducción, el funcionamiento interno de las cámaras".

A continuación, el jugador deberá de avanzar al hueco que le teletransportará a un nuevo escenario. Aquí habrá una cámara gigante con diferentes huecos y por el suelo estarán repartidas las distintas piezas del artefacto. El jugador deberá llevar correctamente cada pieza al hueco que corresponda. Los diferentes encajes serán:

*- **Obturador:** "Es el dispositivo de la cámara que regula el tiempo de exposición de una imagen. Es decir, el aparato con el que se cierra o se abre el objetivo, para regular el tiempo que queremos que esté la luz dentro de la cámara".*

*- **Objetivo:** "Se trata del dispositivo que contiene el conjunto de lentes convergentes y divergentes y, en algunos casos, el sistema de enfoque y/o obturación, que forman parte de la óptica de una cámara tanto fotográfica como de vídeo"*

*- **Sensor:** "El sensor es el corazón de nuestra cámara. Todo lo que hacemos para capturar una buena foto, desde el momento en que encuadramos hasta el momento del disparo, son acciones y pasos que persiguen conducir la luz hacia el sensor".*

- **Flash:** "El flash fotográfico es un dispositivo que actúa como fuente de luz artificial para iluminar la escena. El flash es una fuente de luz artificial intensa y dura, que generalmente abarca poco espacio y es transportable".

- **Visor:** "El visor es el sistema óptico que permite encuadrar el campo visual que se pretende que abarque la imagen. Es decir, el visor es la ventanilla, pantalla o marco que se sirve el fotógrafo para previsualizar la imagen que estamos encuadrando".

Cada vez que el jugador acierte, saltará el diálogo correspondiente que le explicará de forma detallada cuál es el uso de la pieza puesta.

Una vez completado el puzzle, el alumno habrá aprobado la asignatura.

Escenario: Aula + Pequeña ciudad con una cámara gigante

Historia Social de la Comunicación

El jugador que entre en el aula deberá de sentarse en una silla. Cuando lo haga, aparecerá el tutor y le dirá a la clase lo siguiente:

"Bienvenidos a Historia Social de la Comunicación. En esta asignatura entenderemos cuál ha sido la evolución de la comunicación desde sus orígenes con la escritura hasta nuestros días. Hablaremos de las formas de interacción en la época del Imperio Romano, de la imprenta y de Gutenberg, de la llegada de los primeros periódicos, la Revolución Industrial y lo que supuso para la comunicación y muchos otros períodos históricos. De esta forma, comenzaremos la primera clase. Por favor, den un paso adelante y avancen a la pizarra para comenzar con la explicación".

El jugador deberá de ir al portal y se le trasladará a un puzzle tipo 2D, parecido al que tuvimos que hacer como práctica en la asignatura de Navarrete:*

- *El primero de los puzzles estará ambientado en la época romana. Habrá graffitis (pintadas), paneles con comunicados y oradores.*

- *El segundo estará ambientado en la Edad Media con la imprenta. Habrá caballos con cartas en la mano e impresiones de las primeras gacetas.*

- El tercero se basará en la Revolución Industrial. Aparecerá el público masivo, los primeros periódicos con forma y se introducirá como pinceladas la llegada de la primera cámara fotográfica.

Los puzzles tendrán rampas, cubos y diferentes interacciones del tipo plataforma. Cuando finalice el primero, el jugador se teletransportará al segundo. Y así hasta completar el curso. Una vez finalizados estos puzzles, el jugador habrá aprobado la asignatura.

Teoría de la Comunicación

¿Qué son los Mass Media?

A: Los medios de comunicación de masas (V)

B: Las cámaras de fotos

C: Los videos en alta definición

¿Cuáles son las factores del proceso comunicativo?

A: Emisor - Mensaje - Receptor (V)

B: Remitente - destinatario

C: Orador - Público

La comunicación no verbal:

A: Es el proceso de comunicación sin palabras, es decir, mediante indicios, gestos y signos (V)

B: Es el proceso de comunicación con palabras, es decir, de forma explícita

C: No forma parte del proceso de comunicación

Los medios de comunicación:

A: Tienen la capacidad de configurar las tendencias y la opinión de una sociedad

B: Son importantes herramientas dentro de una Democracia

C: Todas las opciones son correctas (V)

La comunicación y el periodismo se consideran en Democracia:

A: El cuarto poder

B: Empresas

C: A y B son correctas (V)

¿Existe literalmente la objetividad?

A: Sí

B: No, pero se entiende que es el objetivo de toda noticia para ser considerada imparcial (V)

Escenario: periódico gigante (*El Diario de la FCOM*)

Prueba final: tipo test (10 preguntas)

*Una vez termine el segundo cuatrimestre del primer año, el alumno recibirá un cuadro de diálogo que pondrá lo siguiente:

"Muy bien! Has superado tu primer año de carrera!"

"...Espera un segundo, parece que ha habido un error con la entrega de tu trabajo de Psicología Social de la Comunicación. Deberías de ir a hablar con el tutor. Pregunta a los otros alumnos, seguro que pueden darte indicaciones de donde se encuentra su despacho".

El jugador deberá de hablar con otros alumnos que le dirán que el despacho del profesor de psicología se encuentra en la cuarta planta. Concretamente, está dentro de la sección H. Una vez el jugador vaya a la cuarta planta y entre en el despacho, le estará esperando el profesor con lo siguiente.

"Parece que no entregaste tu trabajo correctamente..."

"... ¿Cómo que sí lo hiciste? No tengo tu nombre apuntado en ninguno de los trabajos. No te preocunes, en septiembre aprobarás sin problemas. Ya sabes que la primera matrícula que hagas no corre convocatoria ni en junio ni en septiembre. Tienes hasta x veces para aprobar una asignatura. Empezará a correr convocatoria a partir de la segunda matrícula, es decir, cuando vuelvas a cursarla. Y solamente se contará la convocatoria en la que te presentes al examen y lo entregues. Ten en cuenta que cada vez que suspendas, se aumentará el precio de la matrícula en esa asignatura, así que mucho ánimo y estudia, ¡que todo puede sacarse a la primera sin problemas!"

"... ¿Qué hiciste tu trabajo con Fabricio? A ver que lo compruebe un segundo. ¡Es cierto! Había puesto tu nombre muy pequeño y casi no se entendía. Entonces, enhorabuena, tienes un siete en el trabajo. ¡A seguir disfrutando de la carrera!"

Ahora sí, después de esta escena, volverá a aparecer un nuevo cuadro de diálogo que dirá que, efectivamente, el jugador acaba de desbloquear el primer cuatrimestre del segundo año".

De esta forma, en el resultado final de Virtual-CAV, algunas escenas (como la planteada para Tecnología de los Medios Audiovisuales II) se han reutilizado para desarrollar otras como Teoría de la Imagen. Esto se explica a continuación en el desarrollo de cada uno de los niveles.

3.B GUÍA, DISEÑO Y PROGRAMACIÓN DE ESCENAS

En este apartado se analizarán en forma de bloque cada una de las diferentes escenas en las que está dividida Virtual-CAV. Las agrupaciones de niveles que componen este videojuego se denominan Universidad, Asignaturas, Transiciones y Otros. Cada escena, a su vez, estará subdividida en los apartados Desarrollo, dedicado a explicar lo que debe hacer el jugador para superarla; Diseño, que sirve para analizar los diferentes elementos 3D y las texturas que contiene la escena; y Programación, que se encarga de desarrollar los scripts utilizados en cada nivel del videojuego.

A continuación, se conocerán algunos elementos globales insertados dentro de Virtual-CAV como menús de pausa o sonidos. Por otra parte, se desarrollarán de forma breve los diferentes elementos utilizados. Y por último, se analizará el orden en el que están organizadas las escenas (Project Settings).

De esta forma, comenzamos explicando uno a uno los diferentes bloques de escenas que componen Virtual-CAV.

3.B.1 UNIVERSIDAD

El principal desarrollo del videojuego está planteado para realizarse en la Facultad de Comunicación. Aquí el personaje podrá moverse por las diferentes plantas de este edificio y conocer todo el equipamiento del que dispone. De esta forma, la facultad queda dividida en un total de cuatro escenas que representan cada planta del edificio:

- **Planta baja:** La sala principal de la facultad. El alumno podrá visitar la cafetería, secretaría, copistería y salón de grados.
- **Planta primera:** donde se encuentran la biblioteca y la sala Mac.
- **Planta segunda:** aquí el alumno puede entrar al aula 2.7.
- **Planta tercera:** no se puede acceder porque el jugador aún es alumno de primero. A nivel de desarrollo, se ha tomado esta decisión porque se ha querido dar un toque carismático e individual a cada piso y las plantas segunda y tercera son muy similares.
- **Planta cuarta:** el lugar donde se hallan los despachos de los profesores.

Para conseguir que las escenas se asemejasen lo máximo posible a las instalaciones de la Facultad de Comunicación, se utilizaron los diferentes planos de cada planta del centro. Estos se obtuvieron por internet gracias a la ayuda del profesor José Luis Navarrete Cardero. Además, se realizaron diferentes fotografías del interior de la facultad desde todos los ángulos posibles para no perder detalle en los elementos y la posición de los objetos.

Con carácter general, el player no podrá abrir ninguna de las puertas que aparecen en la facultad. Para facilitar la jugabilidad, existen dos marcas en el suelo cuya función es la siguiente:

- **Flecha:** Si el jugador se acerca a estas puertas, podrá abrirlas al pulsar la tecla E y ver el interior de la misma.
- **Información:** Si el personaje se aproxima a esta marca en el suelo, le aparecerá en pantalla un mensaje con una breve descripción de lo que contiene en el interior esa puerta. Sin embargo, no tendrá la capacidad de abrirla. Simplemente cumple una función informativa para los estudiantes de nuevo ingreso o curiosos que quieran conocer el equipamiento del que dispone la Facultad de Comunicación.



Por otra parte, en el apartado de la iluminación, cabe destacar el uso de una luz direccional general con una reducción en la dureza de las sombras para que se asemeje a los leds de los que dispone el centro. Hay que añadir que las escenas no contienen techo visible para que la facultad dé una sensación 3D de mayor espacio, pero sí dispone de un rectángulo invisible que impide al jugador sobrepasar los límites del escenario.

3.B.1.1 Planta baja

Esta planta queda dividida en el videojuego en dos escenas diferentes debido a que el player debe superar una breve misión antes de poder ser alumno de la Universidad de Sevilla. De esta forma, la sala principal de la facultad estaría fragmentada en:

Planta baja parte 1

Desarrollo

En esta escena, el jugador debe de encontrar su matrícula universitaria y depositarla en el buzón que se encuentra en secretaría. El player comienza perdiendo salud de forma continuada debido a los nervios por entrar en un sitio completamente nuevo y diferente a lo que estaba acostumbrado en su paso por el instituto. Para recuperarla, debe visitar la cafetería de la facultad y comerse una caña de chocolate.

Una vez eso pase, podrá ir sin problemas a buscar su matrícula que se encuentra frente a los platós de televisión de la planta baja de la facultad.



Cuando haya conseguido este objeto, tendrá que ir al buzón de secretaría y depositar su matrícula. De esta forma, se habrá convertido en alumno universitario y podrá comenzar su aventura en la Facultad de Comunicación.



Hay que remarcar que en esta escena el jugador no podrá acceder a otras plantas de la facultad ni tampoco acceder a otras habitaciones. Esa función se desbloqueará una vez complete este nivel.

Diseño del nivel

Esta fue la primera escena de la facultad que se diseñó y con la que comenzó todo este proyecto. Las primeras creaciones fueron las paredes y puertas. Las paredes tienen una textura de ladrillo realizada con Photoshop y otra textura de yeso que se obtuvo de la web Pixabay.

Una vez se levantó al completo el esqueleto de la estructura, se comenzó a detallar los diferentes sub-apartados en los que está dividido la planta baja: principal, pasillo, cafetería, secretaría y administración. La sala principal contiene en su interior puertas, mesas, sillas y tablones de anuncios. Abarca desde la entrada principal a la puerta trasera.

El pasillo se extiende desde la copistería hasta el plató uno de televisión, incluyendo espacio desde el cual se puede acceder al salón de actos. Contiene puertas, ascensores, cristaleras, extintores, sillas, mesas, macetas, papeleras y logotipos de cuartos de baño.



La cafetería del centro contiene una barra central, taburetes, mesas, sillas, cristaleras, puertas, elementos de bollería, embellecedores laterales de la pared y dos refrigeradores. La secretaría dispone de diferentes ventanillas, mesas, sillas, tablones de anuncios, puertas, mesas y papeleras.



La administración abarca la zona final derecha de la facultad y contiene una cristalera, puertas, paredes con textura metalizada, macetas y una cristalera.

Cada elemento de la facultad se ha intentado realizar detalladamente para asemejarse a la realidad a través de fotografías. La mayoría de las texturas que contienen los objetos, como la madera, el metalizado, transparencias o el rosado se han obtenido directamente de Unity. Otras como el yeso o mármol se han descargado de la web Pixabay. Por último, el ladrillado se ha realizado a través de Photoshop.

Por otra parte, el modelado de los objetos se ha conseguido gracias a búsquedas intensivas de materiales 3D gratuitos y libres de copyright que se han modificado desde el propio Unity para conseguir una mayor semejanza con la Facultad de Comunicación.

Programación

En este nivel, cabe destacar cuatro funciones programadas: apertura y cierre de puertas automáticas al acercarse y alejarse; pérdida de vida y recuperación de la misma; interacción con la matrícula y carga del siguiente nivel condicionado al cumplimiento de dos funciones.

Existen dos bloques con sendas puertas automáticas en la sala principal (hall) de la Facultad de Comunicación. Como en la vida real, al acercarse a las mismas estas se abren (deslizándose a la izquierda y la derecha) y al alejarse, se cierran. Esto se consigue creando, en primer lugar, un **Empty GameObject** que contiene, como hijos, la puerta izquierda (que dispone, a su vez, de un **Animator** cuyo transform position adquiere valores negativos) y la puerta derecha (su **Animator** le proporciona valores positivos). El **Empty GameObject** tiene el **Sliding_Doors_Script** que activa las animaciones de las puertas en caso de que el **Player** (que tiene una etiqueta homónima) se acerque o se aleje a su **Box Collider Trigger**:

```

void OnTriggerEnter (Collider coll) {
    if (coll.gameObject.tag == "Player")
        SlideDoors (true);
}
void OnTriggerExit (Collider coll) {
    if (coll.gameObject.tag == "Player")
        SlideDoors (false);
}
void SlideDoors (bool state) {
    leftAnim.SetBool ("slide", state);
    rightAnim.SetBool ("slide", state);
}

```

El nerviosismo del jugador por entrar en un espacio desconocido conlleva la pérdida de vida. A nivel de programación, esto se consigue creando un **Empty GameObject** con etiqueta “muerte” y fijando como **Trigger** su **Box Collider**. Al player se le añade el **Script Bolas Verdes** donde resultan fundamentales las variables **float salud** (cuyo valor inicial se define como 1), **damage** (0.02) y **bool menosSalud** (fijada como **true** en el **Start ()**):

```

void OnTriggerStay (Collider col) {
    if (col.CompareTag ("muerte") && menosSalud) {
        salud = salud - damage * Time.deltaTime;
    }
}

```

En caso de que la salud llegue a cero, el jugador morirá y se cargará la escena anterior. Sin embargo, existe la posibilidad de recuperar la vida. Cuando el player se acerca a la barra de cafetería (OnTriggerEnter) se cambia el **SetActive** del **GameObject InformacionCafeteriaCanvas** a **True**. Se habilita un canvas que al pulsar la E, el jugador comerá una caña de chocolate. Al hacerlo, se reproduce un sonido parecido a masticar, aumenta la vida a 1 y se destruye el **Empty GameObject** cuya etiqueta era muerte:

```

void Update () {
    if (Input.GetKeyDown(KeyCode.E) && masSalud) {
        Destroy (muerteObjeto.gameObject);
        masSaludObjeto.gameObject.SetActive (true);
        Destroy (roja2.gameObject);
        musicFXCaña.GetComponent< AudioSource > ().Play ();
        Destroy (informacionCafeteriaCanvas.gameObject);
    }
}

```

```

        masSalud = false;
    }
    barraSalud.value = salud;
    if (barraSalud.value == 0) {
        Destroy (gameObject);
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
- 1);}
```

Paralelamente, el player tendrá la misión de encontrar su matrícula universitaria y entregarla en los buzones de secretaría. Existe un **GameObject** etiquetado como “matricula”. El **Script Bolas Verdes** aplica la interacción de dicho objeto haciendo que al tocarlo se reproduzca un sonido, la matrícula desparezca y, en su lugar, se añada un canvas en la esquina superior derecha:

```

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Matricula")) {
        matriculaCubo.gameObject.SetActive (false);
        matriculaCanvas.gameObject.SetActive (true);
        musicFXConseguir.GetComponent< AudioSource > ().Play ();
    }
}
```

A continuación, el jugador deberá acercarse a los buzones. Solo en caso de que haya cogido la matrícula, le aparecerá una información que le invite a pulsar la E para depositar dicho documento. Al hacerlo, habrá completado la segunda misión:

```

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Buzon")) {
        if (matriculaCanvas.gameObject.activeInHierarchy == true) {
            informacionCanvas.gameObject.SetActive (true);
            canDestroy = true;
        }
    }
}

void Update () {
    if (Input.GetKeyDown(KeyCode.E) && canDestroy) {
        Destroy (matriculaCanvas);
        Destroy (roja1.gameObject);
        canDestroy = false;
    }
}
```

En último lugar, cabe destacar que la carga del siguiente nivel está condicionada al cumplimiento de las dos misiones. Durante todo el nivel, el jugador, al pulsar la P, cargará el menú de pausa. En él, el botón de “objetivos” despliega una pantalla en la que

aparecerán las dos tareas: recuperar la vida y depositar la matrícula, acompañadas de sendas bolas rojas. Cada vez que una misión se complete, la bola roja correspondiente será destruida y aparecerá, en su lugar, una verde (de ahí que el nombre del **Script** sea **Bolas Verdes**):

```
void Update () {
    if (roja1.gameObject == null && roja2.gameObject == null) {
        SiguienteNivel ();
    }
}

public void SiguienteNivel () {
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
}
```

Planta baja parte 2

Desarrollo

En esta escena, el jugador ya se ha convertido en alumno de la Universidad de Sevilla. Podrá acceder a las diferentes plantas de la facultad, entrar a las habitaciones disponibles y consultar la información que proporcionan las puertas si se fija en las marcas del suelo³. En concreto, el personaje principal podrá acceder a la copistería y al salón de grados e informarse del horario de secretaría, el servicio de prácticas, el salón de actos, los platós de televisión y el servicio de préstamo de medios audiovisuales.

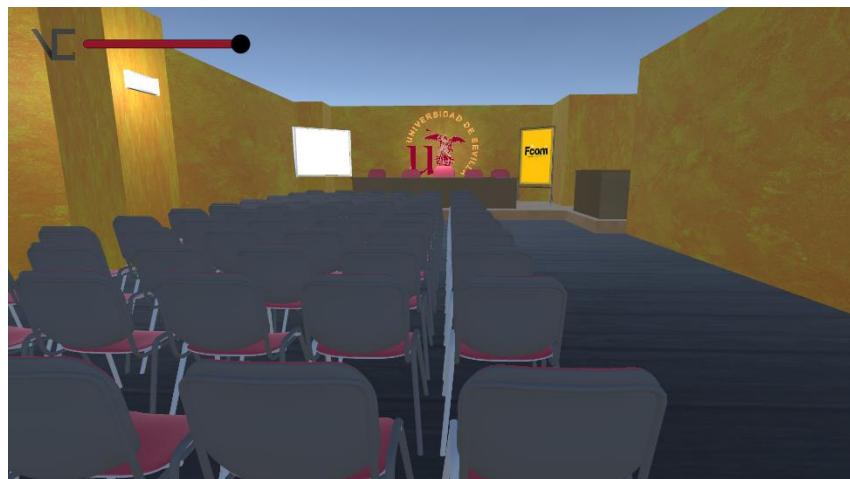
Diseño del nivel

Esta escena mantiene la estructura básica de diseño de la Planta Baja Parte Uno pero contiene en su interior unos cuantos cambios. Se podría decir que es una mejora de la misma, ya que el jugador se ha convertido en alumno y puede acceder a los diferentes servicios que proporciona la Facultad de Comunicación. Los cambios añadidos a nivel de diseño son:

- Los diferentes tablones de anuncios contienen en su interior carteles diseñados a través de imágenes de carteles reales e institucionales de la propia facultad.
- Se puede acceder al interior de la copistería y del salón de grados. En primer lugar, la copistería contiene una estantería de elaboración propia con Unity, un mostrador de elaboración propia, ordenadores de la facultad, mesas, matrículas impresas en folios y una impresora. Por su parte, el salón de grados contiene sillas, dos mesas a modo de atril, una pantalla para proyector, un logotipo de la facultad y su propia iluminación que

³ Puede ver más información sobre estas marcas en la página 17

procede de las lámparas interiores de esta habitación que alumbría la sala y la parte central del ícono de la universidad.



Programación

Las plantas cero (parte dos), uno y dos están entrelazadas de tal modo que se puede viajar de una a otra. Puede hacerse de manera lineal (con las escaleras accedes a la inmediatamente siguiente o anterior) o específica (con el ascensor). Cabe destacar que en el Project Settings el orden de las escenas es el siguiente: planta cero (número cuatro), uno (cinco) y dos (seis), lo que posibilita las funciones recogidas en el **Script Escaleras** que se encuentra en el player de todas las escenas:

```
void OnTriggerEnter (Collider other) {
    if (other.tag == "Escaleras") {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
    }
    if (other.tag == "Rampas") {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
- 1);
    }
    if (other.tag == "Ascensores") {
        SceneManager.LoadScene ("_Ascensor");
    }
}
```

Las escaleras y ascensores, tienen etiquetas para activar esas funciones. Hay particularidades en este sentido en la planta dos y cuatro que se desarrollarán en el apartado de programación de dichas fases. Cabe destacar que las puertas de los ascensores tienen la misma función que las puertas automáticas (ya explicada en la Planta Baja Parte Uno).

Por otro lado, desde este nivel existen puertas interactivas mediante dos procedimientos (que vienen representados por una flecha o una “i” dibujadas en el suelo).

Las flechas indican que se puede entrar en la habitación. Para ello se aplica un procedimiento parecido al de las puertas automáticas, aunque con algunas diferencias. La primera es que el hijo tiene un **Animator** en el que en vez de cambiar los valores del

transform position, se modifican los parámetros del **transform rotation**. El Script **Rotation_Doors** añade, además, una información sobreimpresionada en pantalla que invita a pulsar la E para abrir la puerta, reproduce un sonido y cierra la puerta en caso de que se aleje el player:

```

private void OnTriggerEnter(Collider coll) {
    if (coll.gameObject.tag == "Player") {
        informacionCajon.gameObject.SetActive(true);
    }
}

void OnTriggerStay (Collider coll) {
    if (coll.gameObject.tag == "Player" && Input.GetKeyDown(KeyCode.E)) {
        RotateDoors (true);
        informacionCajon.gameObject.SetActive(false);
        puertaSonido.GetComponent< AudioSource >().Play();
    }
}

void OnTriggerExit(Collider coll) {
    if (coll.gameObject.tag == "Player") {
        RotateDoors(false);
        informacionCajon.gameObject.SetActive(false);
    }
}

void RotateDoors (bool state) {
    rotation.SetBool ("rotate", state);
}

```

La “í” representa que el player no puede acceder a la sala, pero sí que podrá recibir información sobre qué hay en ella. Esto se consigue con las siguientes funciones (se encuentran en el **Script Información Puertas**, componente de las puertas correspondientes):

```

private void OnTriggerEnter(Collider coll) {
    if (coll.gameObject.tag == "Player") {
        informacionPuerta.gameObject.SetActive(true);
    }
}

void OnTriggerExit(Collider coll) {
    if (coll.gameObject.tag == "Player") {
        informacionPuerta.gameObject.SetActive(false);
    }
}

```

3.B.1.2 Planta Primera

Desarrollo

En esta escena, el jugador puede acceder a la biblioteca y a la sala de ordenadores.

Además, dispone de información de la videoteca, las salas de edición de audio, las cabinas de radio y el aula de videojuegos. Hay que destacar que el personaje principal, en la sala de ordenadores de esta planta, puede ver imágenes reales de la Facultad de Comunicación. Por último, el jugador puede decidir cambiar de planta por las escaleras, por las que puede subir al segundo piso o descender a la sala principal, o por el ascensor, desde el que puede acceder a todas las plantas, salvo la tercera.

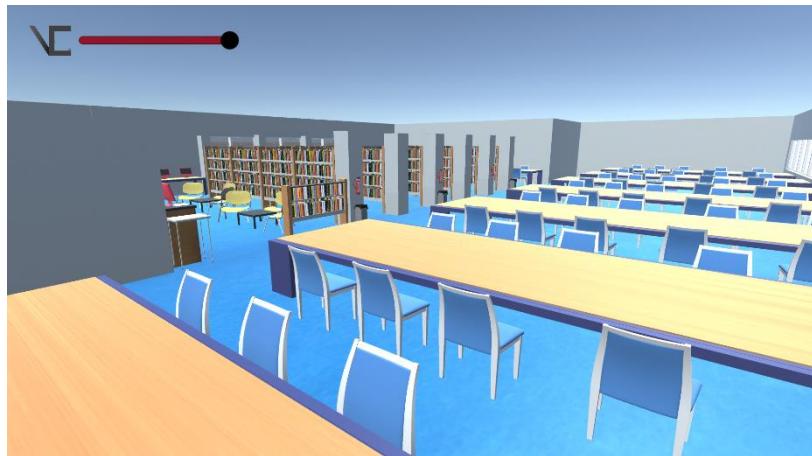
Diseño del nivel

Esta escena mantiene la estructura básica de las paredes, puertas, papeleras, extintores y tablones de anuncios que ya se comentaron en la Planta Baja Parte Uno. El diseño de este nivel corresponde al plano de la planta primera de la Facultad de Comunicación. Además, los tablones contienen carteles con nueva información que no se podía encontrar en la planta baja, además de algunos repetidos por si el jugador no los ha visto en la anterior escena. Las tres principales estructuras de este nivel son:

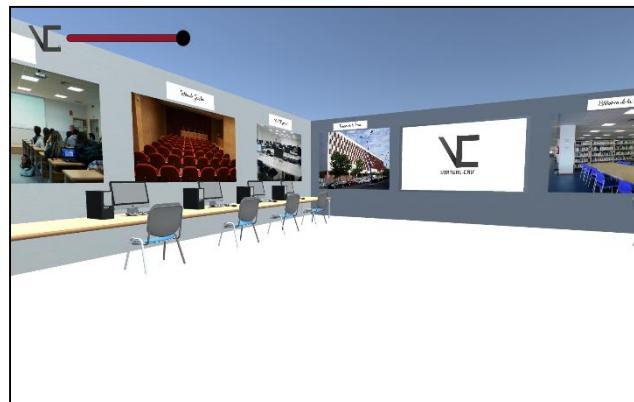
- El pasillo: El lugar donde el jugador puede acceder a las diferentes puertas. A diferencia de la planta baja, este pasillo tiene forma alargada. Dispone de dos cuartos de baño y cristalerías en sus paredes.



- **La biblioteca:** Está diseñada a imagen y semejanza de la propia biblioteca de la Fcom gracias al uso de fotografías realizadas en el interior de la misma. La puerta de acceso es automática. Nada más entrar en la sala, se encuentra el mostrador principal y a la izquierda los ordenadores de consulta rojos con el logotipo de la facultad. Las estanterías, de elaboración propia, están divididas en grupos de dos. Las sillas y mesas mantienen la misma estructura de madera, gris y azul de la realidad. El suelo tiene una textura de mármol azul obtenida de Pixabay. A la derecha del mostrador, se encuentra el ordenador de préstamo y enfrente de este, sillas amarillas en las que los alumnos se pueden sentar a leer. Por último, las paredes tienen una textura de yeso y contienen cristaleras en su interior.



- **La sala de ordenadores:** Esta habitación contiene imágenes reales de la Facultad de Comunicación, además de tablones informativos de las mismas. Por otro lado, se pueden observar sillas, mesas, ordenadores, una pantalla y el logotipo de Virtual-CAV a modo de guiño al propio juego.



Programación

En esta planta se repite el proceso de las puertas automáticas para acceder a la biblioteca, las escaleras y el ascensor para cambiar de nivel y las pinturas en el suelo con el método flecha (se puede acceder a la sala MAC) e “i” (no se puede entrar, pero sí recibir información del aula de videojuegos, videoteca y salas de radio).

3.B.1.3 Planta Segunda

Desarrollo

Este nivel contiene el aula 2.7, el lugar donde se comienzan a desarrollar las diferentes asignaturas de forma consecutiva. Para que eso suceda, el jugador debe acercarse a la pizarra de la clase y pulsar la tecla E. Además de esto, el personaje principal puede decidir volver a la planta uno o coger el ascensor para acceder al cuarto piso. Sin embargo, si el avatar intentase subir a la tercera planta, se le informa de que no puede ya que todavía es un alumno de primero.

Diseño del nivel

A nivel de diseño, la escena presenta una estructura básica similar a la planta primera, es decir, en forma de pasillo. Para darle un toque carismático, se han cambiado los carteles, se ha añadido una administración de objetos perdidos y se ha incluido el aula 2.7.

Por un lado, la administración de objetos perdidos se ha realizado con Unity basada en una fotografía. De apariencia metalizada, cuenta con puertas de diferente color a las de las clases y ventanas con una textura realizada en Photoshop.



Por otro, el aula 2.7 representa fielmente el diseño de las aulas de la Facultad de Comunicación. Contiene banquetas de madera con sillas plegables, una pizarra, una pantalla de proyector, un tablón de anuncios y una mesa, ordenador y silla de profesor. Además, la pizarra contiene un mensaje informativo hecho a tiza para que el jugador sepa que las asignaturas comienzan al acercarse a la misma.

Cabe añadir que se ha pretendido que este piso funcione a modo de guía con la introducción de rótulos informativos para los estudiantes de nuevo ingreso, visitantes que no conozcan la facultad o los propios alumnos que tengan dudas de dónde se encuentra un aula específica situada en esta planta. Estos rótulos, similares a los que se pueden encontrar en la propia facultad, nombran de forma exacta las clases correspondientes a las diferentes puertas. De esta forma, el visitante o alumno que quiera encontrar, por ejemplo, el aula 2.4 en la que se va a realizar una charla informativa, puede acceder a Virtual-CAV y comprobar en qué parte del edificio se encuentra.

Programación

Aunque se mantienen las funciones del ascensor y de las escaleras hacia la planta inferior (planta uno), en este caso no podrá acceder a la planta tres. Como se ha explicado, se ha querido dar un diseño diferenciador a cada planta y, debido a que la dos y la tres son muy parecidas (por no decir idénticas atendiendo a los planos de la Facultad de Comunicación), no se ha construido dicha planta. Cuando el player quiera subir las escaleras, se activará un canvas que le explica que la planta tres no es accesible para él, sino solo para los alumnos de segundo. La puerta que podrá abrir será la del aula 2.7 (habrá una flecha indicativa).

Esto responde al reparto de aulas fijados por la Facultad de Comunicación para el curso 2018/2019⁴. Al entrar, el usuario podrá ver un texto en la pizarra: “Asignaturas aquí”. Cuando el player se acerque, desparecerá el texto y lo sustituirá otro que invite a pulsar la E para cargar las distintas materias. Tanto esto, como la información de no poder acceder a la planta tres se encuentra recogido en el **Script No Puedes Subir** del player:

```
void Update () {
    if (pizarraCanvas.gameObject.activeInHierarchy == true && Input.GetKeyDown(KeyCode.E)) {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1);
    }
}

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Aviso")) {
        avisoCanvas.gameObject.SetActive (true);
    }
    if (col.CompareTag ("Fire")) {
        pizarraCanvas.gameObject.SetActive (true);
        pizarraCanvas2.gameObject.SetActive (false);
    }
}
```

⁴ Junta de Facultad (12 de julio de 2018) “Primero de Grado Comunicación Audiovisual”. Disponible en <https://www.dropbox.com/s/emqms2h96wqmmc1/1%C2%BA%20Grado%20CAV.docx?dl=0>

```
void OnTriggerExit (Collider col) {
    if (col.CompareTag ("Aviso")) {
        avisoCanvas.gameObject.SetActive (false);
    }
    if (col.CompareTag ("Fire")) {
        pizarraCanvas.gameObject.SetActive (false);
        pizarraCanvas2.gameObject.SetActive (true);
    }
}
```

3.B.1.4 Planta Cuarta

Esta planta está sub-dividida en dos partes:

Planta Cuarta Parte 1

Desarrollo

Esta escena sirve para que el jugador pueda conocer cómo es el diseño de la planta cuatro para ver el sitio donde se encuentran las zonas de tutoría de los profesores. Sin embargo, no puede acceder a ningún despacho en concreto, sino que esa función se desbloqueará cuando complete todas las asignaturas. A esta planta solo se puede acceder a través del ascensor.

Diseño de Nivel

En lo que se refiere a este apartado, la escena mantiene una estructura similar a la Planta Segunda pero elimina la administración de objetos perdidos y el aula. El jugador puede ver los carteles personalizados de los que dispone esta planta y dónde se encuentran los despachos.



Programación

En este nivel, el player tiene asociado el **Script Escaleras**, lo que le permite acceder a otras plantas a través del ascensor (el cual tiene el **Sliding_Doors_Scripts** para tener la función de las puertas automáticas). Sin embargo, las escaleras no tendrán etiquetas para no activar las funciones de cargar las escenas anteriores.

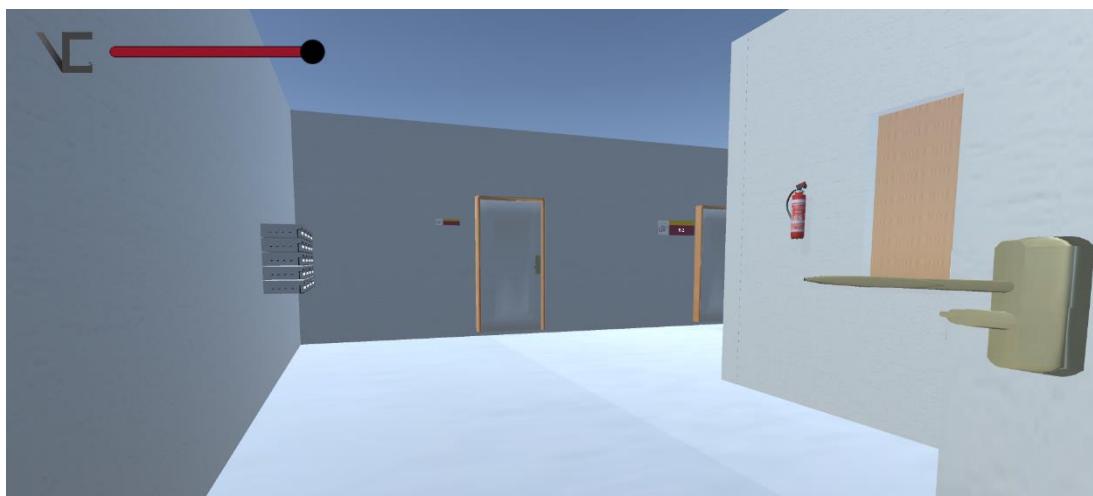
Planta Cuarta Parte 2

Desarrollo

Esta escena se convierte en el nivel final antes de poder visualizar los créditos. Solo se puede acceder a él una vez el jugador apruebe el examen final que se realiza cuando se completan todas las asignaturas. Para superar esta escena, el jugador debe entrar al despacho E-2, una información que se le proporciona en los tablones de anuncios que hay en repartidos por el piso.

Diseño de Nivel

El diseño es similar a la Planta Cuatro Parte Uno. Sin embargo, se introduce la posibilidad de acceder a los despachos de la Sección E. Esta habitación dispone de unos buzones de elaboración propia, puertas y paredes de yeso.



Programación

En este nivel, el player no podrá utilizar las funciones recogidas en el **Script Escaleras** al no tener los objetos interactivos las etiquetas correspondientes. En su lugar, el **Script Player Planta 4** permitirá cargar la última escena (los créditos):

```
void OnTriggerEnter (Collider col) {  
    if (col.CompareTag ("Fire")) {  
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex  
+ 1);  
    }  
}
```

Por último, cabe destacar que la puerta del despacho E-2 se abrirá siguiendo el mismo procedimiento que las puertas interactivas cerca de las cuales había una flecha dibujada.

3.B.2 ASIGNATURAS

Los desafíos de Virtual-CAV se realizan en las diferentes asignaturas de primero de Comunicación Audiovisual. De esta forma, el alumno irá desbloqueando las escenas conforme vaya superando las anteriores. Cada nivel tiene una transición inicial, que explica en qué consiste la asignatura y qué debe realizar el alumno para completar la prueba, y una transición final, que le felicita por haberla superado y le otorga algunos detalles más sobre la asignatura en cuestión. Por lo tanto, el jugador de Virtual-CAV podrá acceder a las siguientes materias en este orden:

- **Teoría de la Información Escrita:** Una simulación de los pasos que se tienen que seguir para solicitar un libro prestado en la biblioteca de la Fcom.
- **Teoría de la Publicidad y las Relaciones Públicas:** Un juego de naves para eliminar el merchandising de otra empresa.
- **Teoría de la Imagen:** Una búsqueda en un gran mapa de los elementos que componen una cámara fotográfica
- **Historia de la Cultura Contemporánea:** una habitación repleta de cuadros con misterios por resolver.
- **Tendencias Literarias:** Cuatro pruebas diferentes de las lecturas obligatorias de la asignatura.

Todos estos niveles son de elaboración propia. Las fases a superar se diseñaron con el objetivo de entretenir al alumno con una dificultad progresiva. Además, se pretende que dichos niveles no sean repetitivos, sino que contenga una variedad de pruebas, con un diseño personalizado para cada asignatura. Cuando el jugador complete las cinco materias, desbloqueará el examen final.

3.B.2.1 Teoría de la Información Escrita

Desarrollo

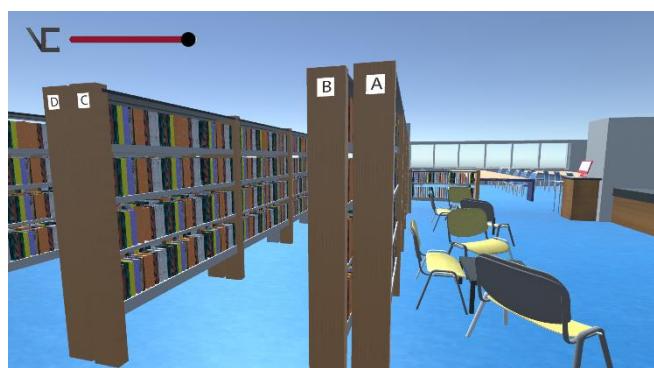
Este nivel representa una simulación de lo que una persona real debe hacer para pedir prestado un libro en la biblioteca de la Facultad de Comunicación, una información que se explica con detalle en la transición que da inicio a esta escena. Por este motivo, el jugador debe buscar tres obras repartidas en las diferentes estanterías de la sala. Para saber la ubicación exacta de las mismas, tendrá que mirar en los ordenadores de consulta pulsando la tecla E. Una vez el jugador consiga los tres libros, podrá acercarse al ordenador situado a la derecha del mostrador principal y pedir prestados los libros. Una vez realice esa acción, habrá completado el nivel y podrá avanzar hasta Teoría de la Publicidad y las Relaciones Públicas.

Diseño del Nivel

Este nivel mantiene el mismo diseño que la biblioteca de la Facultad de Comunicación situada en la escena Planta Primera⁵.

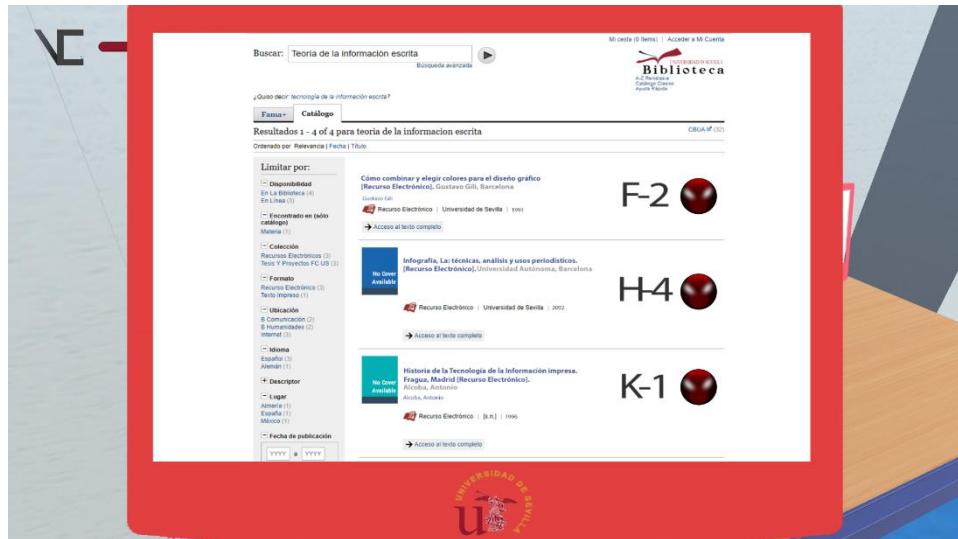
A esta base, se han añadido anotaciones a los extremos de las estanterías y en las baldas de estas para que el jugador tenga mayor facilidad a la hora de encontrar la ubicación exacta de los libros.

Además, las imágenes de las obras que aparecen cuando el jugador consigue encontrarlas se han recuperado de la web de imágenes gratuitas Pixabay.

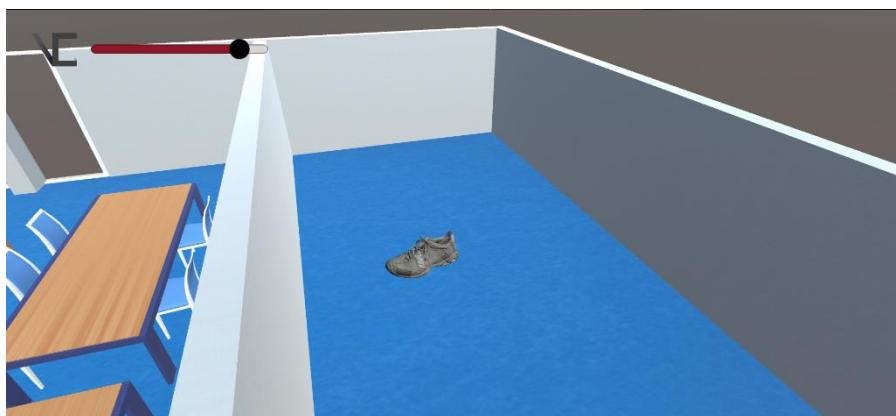


⁵ Puede ver con más detalle este diseño en la página 30 de este documento

Por otro lado, la pantalla de ordenador que aparece al pulsar la tecla E es de diseño propio a partir de una captura de pantalla realizada en la página web Fama.us.es. El nombre de los libros que aparecen en el buscador están recogidos en la bibliografía del plan de estudios de esta asignatura de primero de audiovisuales.



Como último apunte, se ha introducido una zapatilla a modo de Easter Egg⁶ en una de las salas de estudio a la que el jugador no tiene acceso salvo que salte a través de las diferentes estanterías.



⁶ La traducción literal es 'Huevo de Pascua'. Este término hace referencia a un elemento oculto contenido dentro de un videojuego, solamente accesible si se completan una serie de pasos.

Programación

La programación de este nivel reproduce otras funciones ya explicadas anteriormente, para lo cual se ha creado el **Script Player TDIE**. En primer lugar, el jugador perderá vida conforme pase el tiempo (tal y como ocurría en la Planta Baja Parte Uno). En esta ocasión, no habrá modo de recuperar vida, solo de completar el nivel a tiempo (puesto que el **Damage** está fijado en 0.003, tendrá 5 minutos y 34 segundos para hacerlo).

En la esquina superior derecha, el jugador podrá ver los libros que ya ha recogido (siguiendo el mismo formato por el que se optó en la planta baja al coger la matrícula). La única diferencia es que para entender como conseguido cada uno de los libros, el jugador deberá pulsar la E cuando esté próximo a él. Esto, además, activará otra función. El menú del ordenador, que carga la pantalla con la ubicación de los libros, tiene tres bolas cuyo color indica “no conseguido” (rojo) o “conseguido” (verde). Pulsar la E en cada libro destruye la bola roja. Tener destruidas las tres bolas rojas activa la opción de cargar el siguiente nivel. Las variables **bool Libro1, Libro2 y Libro3** se activan y desactivan con los métodos **OnTriggerEnter** y **OnTriggerExit**, respectivamente:

```
void Update () {
    barraSalud.value = salud;
    if (Input.GetKeyDown (KeyCode.E) && ordenadorB) {
        Ordenador ();
    }
    if (Input.GetKeyDown (KeyCode.E) && Libro1B) {
        completa1.gameObject.SetActive (true);
        Destroy (incompleta1.gameObject);
        Destroy (Libro1.gameObject);
        sonidoLibro.GetComponent< AudioSource > ().Play ();
        Libro1B = false;
        libro1Canvas.gameObject.SetActive (true);
    }
    if (Input.GetKeyDown (KeyCode.E) && Libro2B) {
        completa2.gameObject.SetActive (true);
        Destroy (incompleta2.gameObject);
        Destroy (Libro2.gameObject);
        sonidoLibro.GetComponent< AudioSource > ().Play ();
        Libro2B = false;
    }
}
```

```

        libro2Canvas.gameObject.SetActive (true);
    }
    if (Input.GetKeyDown (KeyCode.E) && Libro3B) {
        completa3.gameObject.SetActive (true);
        Destroy (incompleta3.gameObject);
        Destroy (Libro3.gameObject);
        sonidoLibro.GetComponent< AudioSource > ().Play ();
        Libro3B = false;
        libro3Canvas.gameObject.SetActive (true);
    }
    if (incompleta1.gameObject == null && incompleta2.gameObject == null
&& incompleta3.gameObject == null && pedirLibros && Input.GetKeyDown (KeyCode
.E)) {
        SiguienteNivel ();
    }
}

```

En cuanto al Easter Egg, la zapatilla tiene asignada una etiqueta que permite la carga del siguiente nivel. Cabe destacar que este nivel es el único que, aun estando ambientado en la Facultad, carece de techo, pues es el único modo para que el player pueda acceder a la sala donde se encuentra el Easter Egg.

3.B.2.2 Teoría de la Publicidad y las Relaciones Públcas

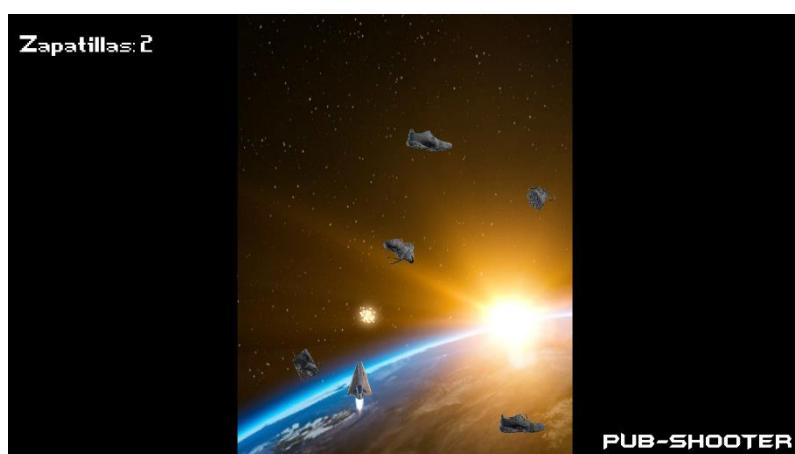
Desarrollo

La siguiente asignatura que cursará el alumno será Teoría de la Publicidad y las Relaciones Públcas. Como se explica en la transición de la materia, el método para aprobar dicha asignatura consistirá en la realización de un trabajo: un juego de rol en el que usuario, perteneciente a una empresa deportiva, se convierte en una nave aeroespacial cuyo objetivo consistirá en destruir bienes producidos por una compañía rival: zapatillas, gorras y camisetas. Como conclusión a la escena, se encuentra un enemigo final que metaforiza el dinero, el principal recurso y la base en la que se desarrollan las empresas. Por ello, se ha desarrollado y programado un nivel con cuatro fases con formato *Simple Top Down Arcade Style Shooter*⁷.

Diseño del Nivel

En cuestión de diseño, se puede desglosar este nivel en tres partes diferentes: Fondo, Objetos y Personajes.

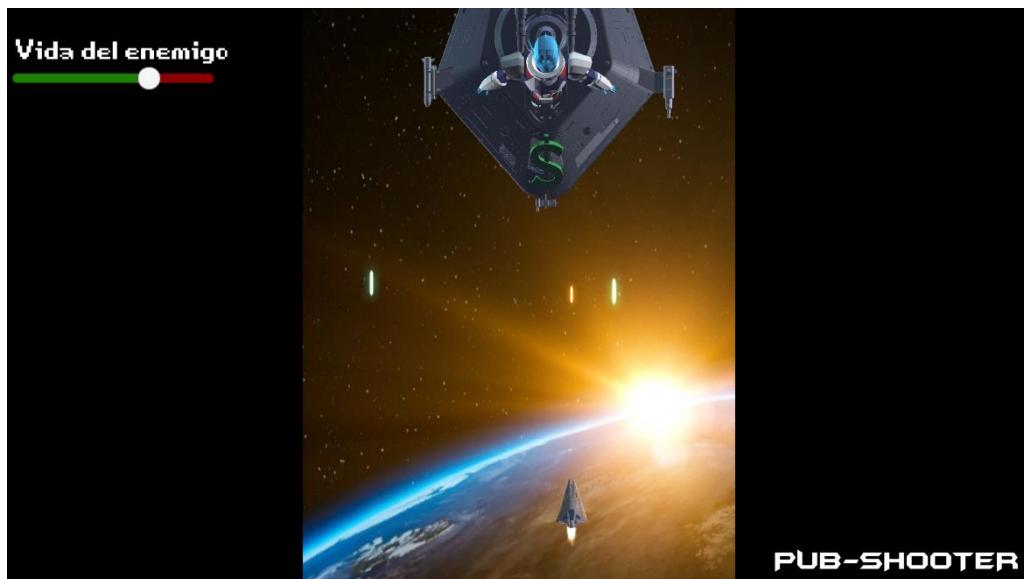
En lo que se refiere al fondo, se ha realizado una edición con Photoshop de una fotografía que simule el espacio exterior. En el tema de los objetos, las gorras, camisetas y zapatillas se han obtenido de webs de material 3D gratuito y se han texturizado.



En cuestión a los personajes, hay que destacar que el juego tiene dos principales:

⁷ Space Shooter Tutorial. Disponible en <https://unity3d.com/es/learn/tutorials/s/space-shooter-tutorial>
Consultado el 15 de febrero de 2018.

- La nave del jugador: Este barco espacial se ha diseñado a partir de los Assets gratuitos que aporta Unity. Tanto los disparos que proyecta como las texturas son de uso libre y se han recopilado del propio programa.
- La nave del enemigo final (el dinero): El antagonista del nivel es de diseño propio a partir de elementos 3D gratuitos recopilados de diferentes webs. En este caso, la nave estaría dividida en la estructura principal del barco espacial, el marciano que la pilota y el símbolo del dólar.



Hay que destacar que el este nivel tiene una vista en cámara 2D desde la parte superior del personaje principal.

Programación

Se trata del único nivel de las cinco asignaturas que no se presenta con el formato de First Person. En su lugar, aparecerá en pantalla la nave (jugador), que tiene el **Script Player Controller** como componente. Los movimientos se realizarán con las teclas WASD o flechas en un baremo de valores recogidos en las variables **xMin**, **xMax**, **zMin** y **zMax** (lo que imposibilita salirse de la pantalla):

```
void FixedUpdate () {
    float moveHorizontal = Input.GetAxis ("Horizontal");
```

```

        float moveVertical = Input.GetAxis ("Vertical");
        Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
        rb.velocity = movement * speed;
        rb.position = new Vector3
            (Mathf.Clamp (rb.position.x, boundary.xMin, boundary.xMax),
            0.0f,
            Mathf.Clamp (rb.position.z, boundary.zMin, boundary.zMax));
        rb.rotation = Quaternion.Euler (0.0f, 0.0f, rb.velocity.x * -tilt);
    }
}

```

No podrán usarse las teclas interactivas E ni P, es decir, no habrá menú de pausa. La nave disparará pulsando la barra espaciadora (el **fireRate** está fijado en 0.25, por lo que podrá disparar 4 veces por segundo):

```

void Update () {
    if (Input.GetButton ("Fire1") && Time.time > nextFire) {
        nextFire = Time.time + fireRate;
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
        audio.Play ();
    }
}

```

La salud del Player es 1, por lo que recibir un impacto supone la muerte del jugador y cambiará la asignación de la variable **bool restart** de false (recogida en el **Start ()**) a true. Para comenzar de nuevo la fase, deberá pulsar la tecla R (gracias a la siguiente función recogida en el **Script GameController** del objeto homónimo):

```

if (restart) {
    if (Input.GetKeyDown (KeyCode.R)) {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 0);
    }
}

```

En las fases uno, dos y tres, el **Script GameController** recoge el objeto que actúa como asteroide (enemigo que cae sobre la nave y que puede ser destruido por la misma). La variable **public GameObject Hazard** se corresponde con un prefab que depende de la fase: zapatilla (uno), gorra (dos) y camiseta (tres).

Dichos prefabs tienen como componentes principales: **Rigidbody**; **Capsule Collider**; **Script Mover** (que provoca el movimiento hacia abajo) y **Script Random Rotator** (que provoca la rotación del prefab). Por último, el **Script Destroy By Contact** inicia las explosiones del player en caso de colisión con el hazard o del prefab si este es

alcanzado por un proyectil del jugador, lo que, además, actualiza la puntuación (score), que se sitúa en la esquina superior izquierda de la pantalla.

La cantidad de hazards que caen viene determinada por varias funciones. La variable **public int hazardCount** recoge la cantidad de prefabs que aparecerán en cada tanda: 10 zapatillas, 15 gorras e infinitas camisetas. La variable **public Vector3 spawnValues** contiene los valores máximo y mínimo del intervalo de posiciones desde las que se inician los hazards. Las variables **public float spawnWait, startWait y waveWait** recogen información relacionada con el inicio de las tandas y el tiempo de espera entre las mismas.

```
IEnumerator Zapatillas () {
    yield return new WaitForSeconds (startWait);
    while (true) {
        for (int i = 0; i < hazardCount; i++) {
            Vector3 spawnposition = new Vector3 (Random.Range (-spawnValues.x, spawnValues.x), spawnValues.y, spawnValues.z);
            Quaternion spawnrotation = Quaternion.identity;
            Instantiate (hazard, spawnposition, spawnrotation);
            yield return new WaitForSeconds (spawnWait);
        }
        yield return new WaitForSeconds (waveWait);
    }
}
```

Existe una variable **public float Ganar** que fija el número de hazards que deberá destruir el player para cargar la siguiente fase. En la uno, deberá eliminar 10 zapatillas; en la dos, 15 gorras y en la tres, 20 camisetas. Cuando se alcance el objetivo de la fase, el tiempo se congelará y aparecerá un botón que, de pulsarlo, cargará la siguiente fase:

```
public void AddScore (int newScoreValue) {
    score += newScoreValue;
    UpdateScore ();
    if (score == ganar) {
        win = true;
    }
}

void Update () {
    if (win) {
        siguiente.gameObject.SetActive (true);
        Time.timeScale = 0;
    }
}
```

En la fase cuatro la salud del enemigo final corresponde al impacto de 20 disparos del Player (su barra de health aparecerá sobreimpresionada). Las funciones descritas en el **Script GameController** referentes al hazard carecen de valor por no ser necesarias para esta fase. La nave enemiga, que tendrá un movimiento horizontal en bucle gracias al componente **Animator**, podrá disparar para eliminar al jugador. Los disparos se dividirán en dos tandas: una primera con dos proyectiles laterales (uno procedente de la torreta izquierda y otro de la derecha) y una segunda con una única munición proveniente de la torreta central:

```
void Start () {
    InvokeRepeating ("Fire1", delay1, fireRate1);
    InvokeRepeating ("Fire2", delay2, fireRate2);
}

void Fire1 () {
    Instantiate (shot1, boss1.position, boss1.rotation);
    Instantiate (shot1, boss2.position, boss2.rotation);
    fxDisparo.Play ();
}

void Fire2 () {
    Instantiate (shot2, boss3.position, boss3.rotation);
    fxDisparo.Play ();
}
```

3.B.2.3 Teoría de la Imagen

Desarrollo

En esta escena, el jugador deberá de encontrar tres partes principales de una cámara para poder superarla: el objetivo, la lente y el visor. Para ello, será transportado a un enorme descampado en el que deberá buscar por el mapa estos elementos con un tiempo limitado.

- El **objetivo** se encuentra en el fondo izquierdo del mapa dentro de una cabaña de madera.
- La **lente** se sitúa en lo alto de la montaña central del escenario.
- El **visor** se localiza en una plataforma flotante en el fondo derecho del terreno a la que el jugador debe acceder superando un mini-juego de saltos.

Una vez se consigan estos objetos, se volverá a formar la cámara y el jugador habrá superado el nivel, lo que le permitirá avanzar hasta la asignatura Historia de la Cultura Contemporánea.



Diseño del Nivel

El escenario se ha realizado con la herramienta de Unity denominada Terrain. Esta función permite la introducción de elementos visuales de forma rápida y sencilla, así como el modelado de estructuras naturales como montañas. Esto significa que los objetos que conforman el mapa se han obtenido todos del propio programa: el césped, los árboles, las montañas, los arbustos y la textura de cada uno de ellos.



Diferente es el caso de los elementos interactivos que se han personalizado para la recreación de esta asignatura. La cámara, la lente y el objetivo se han obtenido de páginas web de materiales 3D gratuitos. También la cabaña donde se encuentra el objetivo y el reloj que el personaje

debe coger para poder alcanzar el visor de la cámara. Por otro lado, el visor se ha editado en Unity con la imagen de la pantalla principal de Virtual-CAV que metaforiza que



dicha fotografía es de realización y edición propia.

Por otra parte, el suelo donde aparece el personaje principal está texturizado con el mármol que se utilizó en la Universidad.

Como último apunte, es importante destacar que el mapa es muy grande. Debido a esto, se han introducido plataformas y elementos que sirvan como pista para que el jugador sepa hacia qué lados debe acudir para encontrar los objetos que se le exigen para superar el nivel. Es el caso de la montaña central o la pequeña ladera en la que está situada la cabaña de madera. Además, hay que añadir que se han limitado los extremos del escenario por donde puede caminar el personaje principal con elementos invisibles. Todo para facilitar la jugabilidad y el entretenimiento del alumno a la hora de superar este nivel.



Programación

De nuevo, se ha optado por el formato de nivel en el que el paso del tiempo resta vida al player (en el **Script Interaccion TI** el **Damage** está fijado en 0.0017, lo que significa 9 minutos y 49 segundos para completar la fase).

Como en la Planta Baja Parte Uno y en Tecnología de la Información Escrita, cada vez que se obtenga un objeto, éste desaparecerá y se reproducirá un sonido. La diferencia es que, en esta ocasión, los canvas aparecen al entrar el player en la cámara inicial. Coger los objetos solo supone cambiar el color de su canvas de negro a blanco:

```

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("CamaraInicio")) {
        tiempoCanvas = true;
        CamaraInicio.gameObject.SetActive (false);
        TodosLosCanvas.gameObject.SetActive (true);
        sonidoDestruirCamara.GetComponent< AudioSource > ().Play ();
        objetosInteractivos.gameObject.SetActive (true);
        objetivoInicio.gameObject.SetActive (false);
        objetivoFinal.gameObject.SetActive (true);
    }
    if (col.CompareTag ("Lente")) {
        Destroy (LenteOscura.gameObject);
        LenteClara.gameObject.SetActive (true);
        Lente.gameObject.SetActive (false);
        sonido.GetComponent< AudioSource > ().Play ();
    }
}

```

Para poder adquirir el visor, el player tiene que interaccionar con un reloj que habilita unas plataformas por un tiempo determinado:

```

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Reloj")) {
        inicioTiempo = true;
        Reloj.gameObject.SetActive (false);
        sonidoTicTac.GetComponent< AudioSource > ().Play ();
    }
}

void Update () {
    if (inicioTiempo == true) {
        finalTiempo = 15;
        tiempo = tiempo + Time.deltaTime;
        PlataformasContrarreloj.gameObject.SetActive (true);
        if (tiempo > finalTiempo) {
            PlataformasContrarreloj.gameObject.SetActive (false);
            tiempo = 0;
            inicioTiempo = false;
            Reloj.gameObject.SetActive (true);
            sonidoTicTac.GetComponent< AudioSource > ().Stop ();
        }
    }
}

```

La cámara que le permitirá cargar la siguiente escena aparecerá una vez que el player haya conseguido los tres objetos (y que, por ello, haya destruido los canvas oscuros):

```

void Update () {
    if (LenteOscura.gameObject == null && ObjetoOscurito.gameObject == null
11 && CarcasaOscura.gameObject == null) {
        CamaraHecha.gameObject.SetActive (true);
    }
}

```

3.B.2.4 Historia de la Cultura Contemporánea

Desarrollo

Esta escena se desarrolla dentro de una habitación repleta de cuadros que se estudian en esta asignatura. Para superarla, el jugador deberá de encontrar el cofre perdido gracias a una serie de pistas que se le van introduciendo conforme va resolviendo los acertijos. El primero de ellos es encontrar el cajón donde se halla la frase que le permite seguir avanzando: "Pulsar la tecla E en el cuadro de Francia". Esta primera información también se le aporta en la transición inicial de la asignatura.



De esta forma, el jugador tiene que encontrar en la sala la obra 'La Libertad Guiando al Pueblo' y pulsar dicha tecla para conseguir la llave que le permita abrir una puerta que estaba cerrada. Cuando acceda al interior de esa puerta, el jugador deberá abrir un cajón en el que hay una botella. Esa pista le permitirá adivinar que ahora es el momento de acudir a la barra del bar, que se encuentra iluminada. El cofre está tras la pared de botellas del bar.

Para superar estas pruebas, el jugador tendrá cinco minutos y medio. Una vez haya descubierto el cofre, el jugador accederá a la asignatura Tendencias Literarias.

Diseño del Nivel

Como se ha explicado antes, esta escena se desarrolla en el interior de una habitación. Este escenario tiene su propia iluminación y ambientación que envuelve al personaje en un aura de misterio.



El esqueleto de la habitación se ha obtenido de una web 3D de materiales gratuitos. A esta estructura, se le han eliminado algunos sofás, añadido objetos como camas, cuadros o lámparas, editado la interacción de las escaleras, mejorado la iluminación, añadido elementos interactivos con el personaje, entre muchas otras cosas más.



Hay que destacar que los elementos descargados de internet no tienen cajas de interacción por lo que el personaje puede atravesarlos. Eso quiere decir que, a nivel de diseño, se ha tenido que introducir dichas cajas una por una para que esto no suceda.

Uno de los elementos más importantes de este escenario son los cuadros, todos ellos estudiados en la asignatura. Las imágenes de estos se han conseguido a través de internet y se han insertado en el interior del objeto tablón de anuncios utilizado en la Universidad.



Por último, la iluminación oscura reforzada con pequeños puntos de luz intenta introducir al personaje en un ambiente de misterio. Para conseguir esa intensidad lumínica se han tenido que editar las capas de los objetos para que la luz solamente ilumine al elemento indicado.



Programación

Una vez más, el player perderá vida conforme pase el tiempo (el **Script Player HCC** tiene como valor de **Damage** 0.003: 5 minutos y 34 segundos). Como ocurre con las puertas que se pueden abrir al pulsar la E, hay cajones interactivos con la misma tecla. Para ello, el hijo contiene un **Animator** donde se modifican valores del **transform position**. Los cajones que contienen el libro y la pistola tienen un **Box Collider** que interacciona con el personaje (etiquetado como “player”) gracias al **Script Open Drawer**, cuyo funcionamiento es idéntico al de las puertas que se abren al pulsar la E: al estar el jugador cerca, se activa una información que invita a pulsar la tecla. La información desaparece cuando el player se aleja.

En la planta superior, el cuadro de la Revolución Francesa tiene una animación donde **Y** del **transform position** adquiere valores negativos (desciende gracias al **Script Cae Cuadro**, de idéntico funcionamiento a **Open Drawer** salvo porque en este caso no habrá canvas informativo). La Libertad Guiando al Pueblo (Delacroix) tiene escondido tras él una llave que reproduce las funciones de la matrícula (planta cero parte uno), libros (Tecnología de la Información Escrita) y componentes de la cámara (Teoría de la Imagen): la llave desaparece, se muestra un canvas en la esquina superior derecha y se reproduce un sonido.

Hay un cuarto en la planta superior cuya puerta está cerrada con llave. Si el jugador aún no la ha conseguido, aparece una información (como ocurre cuando en la planta dos de la Facultad el player quiere acceder a la planta tres). Si el jugador tiene la llave, podrá abrir la puerta que tiene el **Script Puerta Parejo** (funciona como otras puertas):

```
void OnTriggerStay (Collider coll) {
    if (coll.gameObject.tag == "Player" && llaveImaginaria == true) {
        if (Input.GetKeyDown (KeyCode.E)) {
            RotateDoors (true);
            puertaSonido.GetComponent< AudioSource > ().Play ();
            llaveCanvas.gameObject.SetActive (false);
    }}
```

```

        }
    }

    void OnTriggerExit(Collider coll) {
        if (coll.gameObject.tag == "Player") {
            RotateDoors(false);
        }
    }

    void RotateDoors (bool state) {
        rotation.SetBool ("rotate", state);
    }
}

```

En el cuarto, el jugador podrá abrir un cajón (como hiciera al inicio del nivel en la planta baja). Lo novedoso del **Script Trigger Parejo** es el cambio en el **SetActive** de dos **GameObjects** de la planta baja: una pared (activa desde el **Start ()**) que no puede ser atravesada y otra (que se activa al abrir el cajón) cuyo **Box Collider Trigger** posibilita al player caminar a través de la misma para descubrir una habitación oculta.

Para ello, ha sido creada una variable **bool paredes** (fijada como **false** en el **Start ()**):

```

void OnTriggerStay (Collider coll) {
    if (coll.gameObject.tag == "Player" && Input.GetKeyDown (KeyCode.E))
    {
        OpenTheDrawer (true);
        informacionCajon.gameObject.SetActive (false);
        paredes = true;
        sonidoCajon.GetComponent< AudioSource > ().Play ();
    }
}

void Update () {
    if (paredes == true) {
        pared1.gameObject.SetActive (false);
        pared2.gameObject.SetActive (true);
        luzPared.gameObject.SetActive (true);
    }
}

```

En la habitación oculta, hay un cofre que permite cargar el siguiente nivel siguiendo la misma función que la zapatilla del Easter Egg de Tecnología de la Información Escrita.

3.B.2.5 Tendencias Literarias

Desarrollo

El nivel está dividido en cuatro fases que representan los cuatro libros de lectura obligatoria de la asignatura: Las Lágrimas de San Lorenzo, el Enredo de la Bolsa y la Vida, Hamelín y El Retablo de las Maravillas. De esta forma, el jugador comienza en una plataforma principal con la obra El Retablo de las Maravillas. Conforme vaya superando las diferentes pruebas ligada a los libros, se irán desbloqueando los siguientes. Los desafíos a superar en cada libro son:

- **El Retablo de las Maravillas:** Un puzzle que simula un robo entre los protagonistas principales de esta historia.
- **Las Lágrimas de San Lorenzo:** Un enorme castillo con diferentes pruebas de salto y esquivar que representan la fugacidad de la vida. Una vez el jugador llegue al final, se trasladará a una pequeña playa nocturna con un enorme cielo estrellado, similar al narrado en esta obra.
- **Hamelín:** Un mini-juego de plataformas en el que se pueden observar un montón de niños pequeños jugando. Cuando se llega a la plataforma final, se descubre que un adulto parece estar abrazando a esos dos niños. Esto representa el tema de la pederastia que abarca este libro.
- **El Enredo de la Bolsa y la Vida:** Un enorme laberinto en el que el jugador deberá encontrar cinco elementos importantes en la historia del libro. Cuando los descubre, se le informa con un pequeño texto en qué se relacionan dichos objetos con esta obra. Cuando encuentre los cinco, habrá superado esta prueba.

Una vez el jugador supere los cuatro desafíos, accederá al examen, el último paso antes de llegar al final de Virtual-CAV.

Diseño del Nivel

Como se ha explicado antes, esta escena está sub-dividida en cinco escenarios, cada una con su diseño y programación propios:

- **Escenario Principal:** Este mapeado tiene un diseño simple. Es una plataforma central rodeada de piedra con cuatro columnas interiores y antorchas que iluminan la caverna. A su vez, hay cuatro mesas en las que se sitúan los libros. Al principio, la caverna está tapiada por todos lados, pero una vez el jugador se acerca al libro y pulsa la tecla E, se desbloquea un camino secreto hacia la prueba.



Hay que destacar que la plataforma principal está rodeada de agua y de montañas, a modo de pequeña isla en el interior de un lago. Las texturas de piedra de esta escena se han obtenido de Unity. Las antorchas y mesas son elementos descargados de webs de material 3D gratuito.

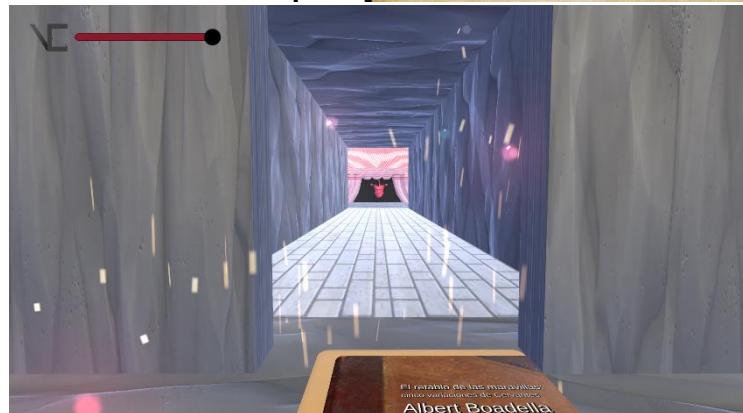
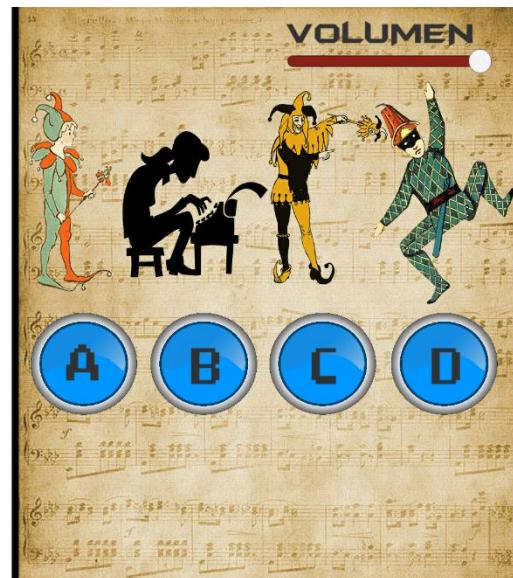
- **El Retablo de las Maravillas:** La primera de las pruebas. Una vez el jugador pulse E tiene acceso a un enorme pasillo en el que se observa al final una especie de carpa de circo y la cabeza de un pequeño buzón en la puerta. Cuando el jugador se acerca, se despliega el siguiente puzzle que debe resolver.

El diseño del puzzle es de elaboración propia gracias a la herramienta Photoshop.

El enunciado es una adaptación de famosos puzzles de otros videojuegos como 'El Profesor Layton'.



Los condes de Daganzo se han enterado de que se han visto envueltos en una trama de timadores. Por lo tanto, quieren que le devuelvan su dinero robado. Uno de estos cuatro señores lo tiene. Sabiendo que el que tiene el botín está mintiendo y que al menos una de las otras tres personas también miente. ¿Puedes deducir quién tiene el dinero robado de los condes?
- Arbequino (A) dice que él no tiene el dinero
- Boadella (B) declara que lo tiene él
- Chanfález (C) afirma que Arbequino no lo tiene
- Rabelín (D) manifiesta que Boadella no es el que tiene el dinero



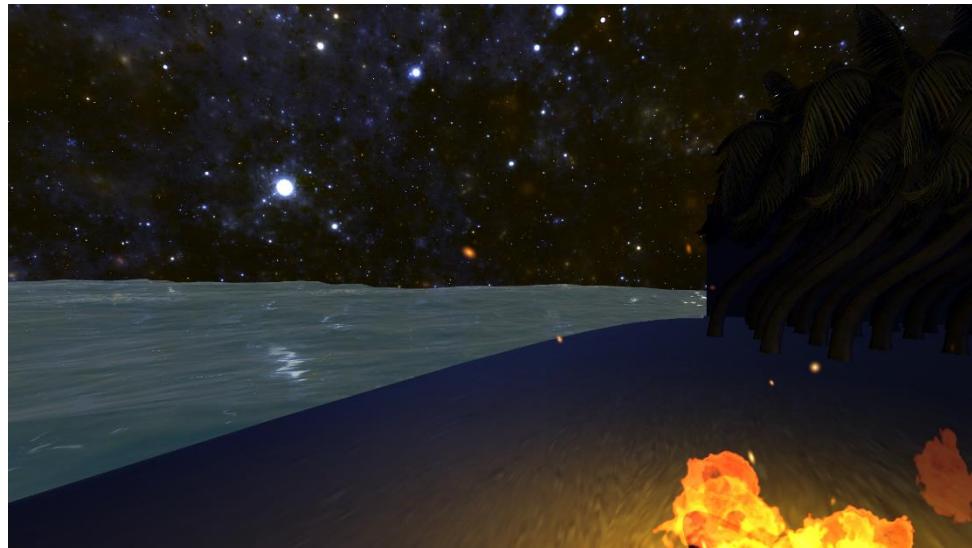
La carpa y la cabeza del bufón se obtuvieron de páginas 3D de elementos gratuitos. Y la textura de las mismas se realizó con Photoshop gracias a fotografías descargadas de Pixabay.

- **Las Lágrimas de San Lorenzo:** El segundo desafío. Esta prueba tiene una iluminación nocturna propia debido a la grandiosidad del castillo. El diseño de este elemento es de elaboración propia. Se utilizaron texturas de la web Pixabay y elementos decorativos como explosiones de fuego de Unity o pilares de roca que caen del techo o las paredes.

La fase a superar se diseñó como una carrera contrarreloj, pero debido a la dificultad de los diferentes saltos se optó por permitir al jugador no tener tiempo para completarla. Los elementos que se obtuvieron de páginas web 3D fueron las columnas, antorchas, la cabeza del dragón y las dos estrellas que se pueden ver en el interior del castillo.



Una vez se supera esta prueba, el jugador accede a la playa narrada en las Lágrimas de San Lorenzo. Para la elaboración de ésta se utilizaron las texturas de arena, palmeras, agua y madera que proporciona Unity. El cielo, por otro lado, se descargó de la página web del programa y envuelve al personaje principal en un aire de calma y tranquilidad, similar a lo que se narra en el libro.

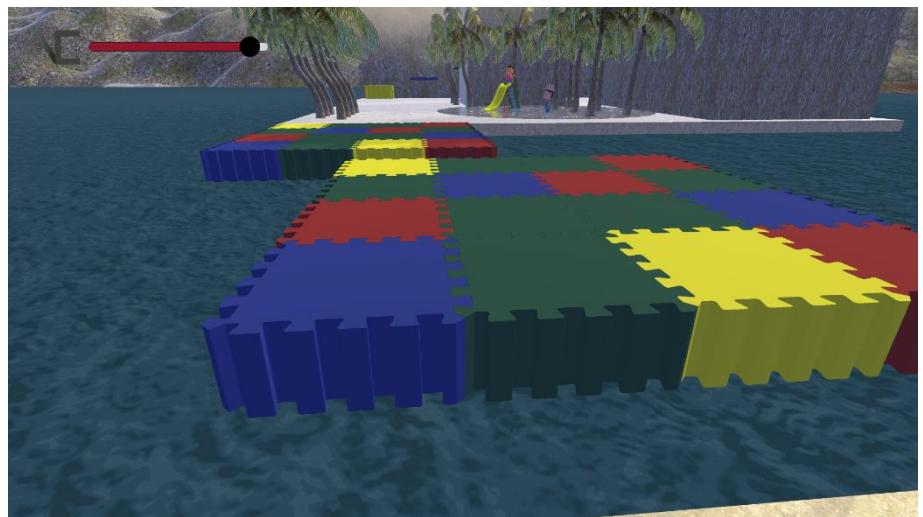


Tras diez segundos, el jugador vuelve a la plataforma inicial y desbloquea el siguiente libro.

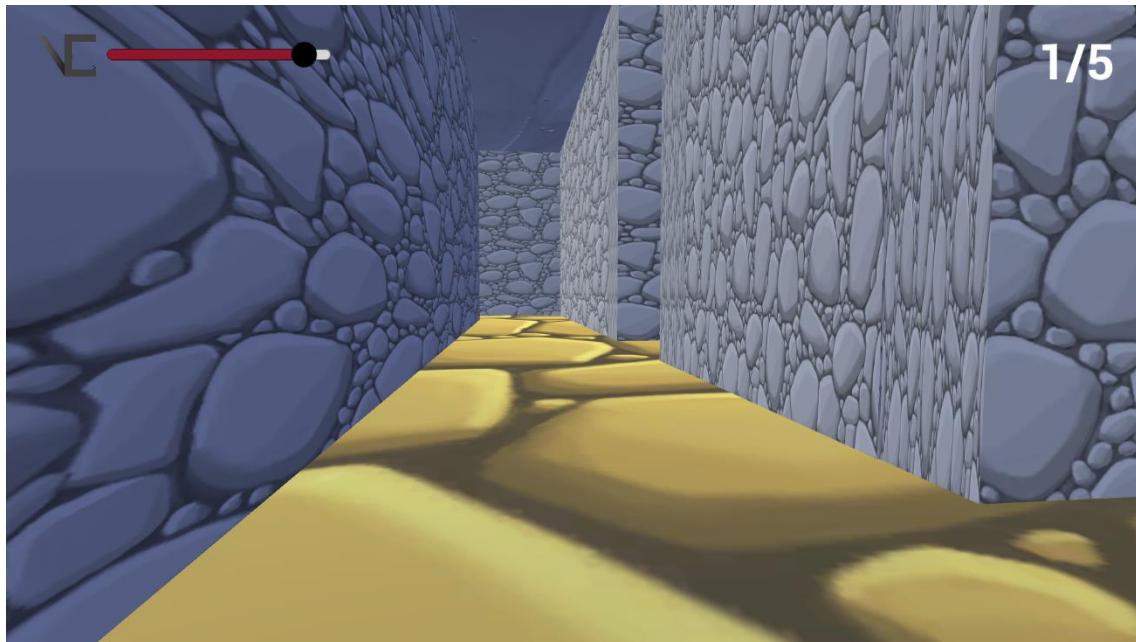
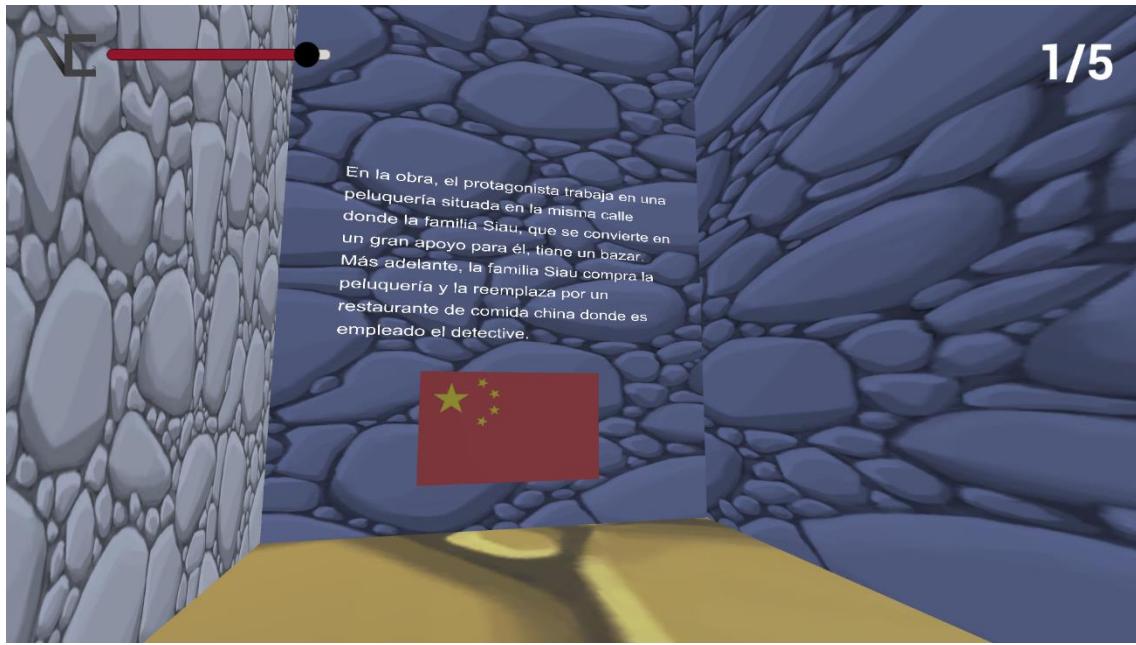
- **Hamelín:** La tercera misión se desarrolla en una playa que simula un lugar recreativo de niños pequeños. Hay toboganes, bicicletas, piezas de puzzle en forma de plataforma y una pequeña carrera animada entre los dos pequeños. La estructura básica de diseño se realizó con los elementos y texturas que proporciona Unity. A su vez, los elementos que dan personalidad como los niños, bicis o toboganes se descargaron de webs de material 3D gratuito.

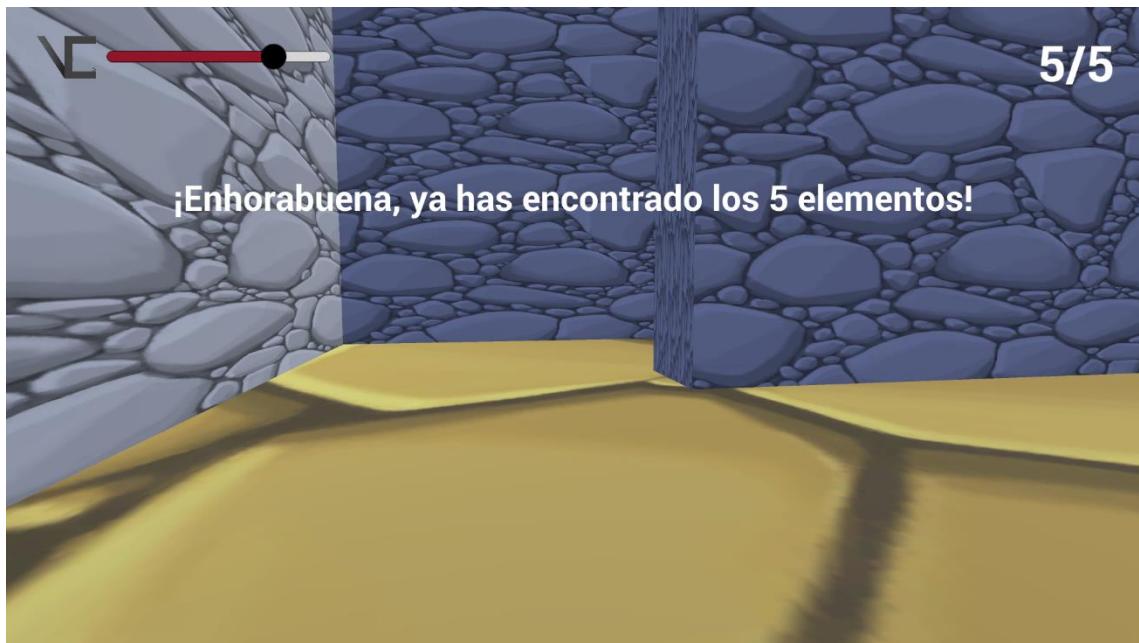


Al completar esta prueba, el jugador puede ver a un hombre abrazando a los dos niños, similar a lo que sucede en este libro.



- **El Enredo de la Bolsa y la Vida:** La última de las pruebas. El diseño del laberinto es de elaboración propia. Las paredes se han texturizado con el propio Unity. El fuego que impide el paso antes de encontrar los cinco objetos es el mismo que se puede encontrar en el castillo de Las Lágrimas de San Lorenzo.





Programación

La asignatura Tendencias Literarias en la Cultura Contemporánea está dividida en cinco fases entrelazadas entre sí mediante la función de construir la siguiente escena (tal y como se hacía entre las plantas cero, uno y dos de la Facultad de Comunicación).

En las escenas uno, tres y cinco, que se corresponden con una sala interna con cuatro mesas, el player podrá pulsar la E para destruir cada una de las puertas que oculta el

mundo de los libros de lectura obligatoria y activar una explosión. Esto se consigue con una variable **bool** por cada libro que se activa como **true** cuando el player entra al **Box Collider Trigger** de las mesas. Cabe destacar que, además, cada vez que el jugador caiga al agua, volverá al inicio y se reproducirá un sonido:

```

void OnTriggerEnter (Collider Col) {
    if (Col.CompareTag ("Retablo")) {
        RetabloB = true;
    }
    if (Col.CompareTag ("agua")) {
        waterSplash.GetComponent< AudioSource > ().Play ();
        move.transform.position = inicio;
    }
    if (Col.CompareTag ("Escaleras")) {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
    }
}

void Update () {
    if (Input.GetKeyDown (KeyCode.E) && RetabloB == true) {
        Retablo.SetActive (true);
        ParedRetablo.gameObject.SetActive (false);
        Instantiate (Explosion, ObjetoExplosionRetablo.transform.position
, ObjetoExplosionRetablo.transform.rotation);
        RetabloB = false;
    }
}

```

Tras haber descrito las funciones generales, se explican funciones específicas de cada nivel.

- **El Retablo de las Maravillas.** Hay un **GameObject MusicManager** cuyo **Script Game Controller Puzzle** fija en los métodos **Acierto ()** y **Fallo ()** funciones para cargar la siguiente y la anterior escena, respectivamente (igual que en las plantas de la Facultad). En el Canvas general, los UI botones tienen asignados en su función **On Click ()** la reproducción de un sonido y el método acierto (A) o fallo (B, C y D). También hay un controlador de volumen como en el menú de pausa que se desarrollará más adelante.
- **Las Lágrimas de San Lorenzo.** La tercera escena comienza con un canvas sobreimpresionado que desaparece cuando el player pulsa la E para habilitar el

mundo correspondiente. El jugador accederá a un castillo en el que deberá evitar obstáculos para no perder vida. Estos son fuegos (los deberá saltar) y muros (los deberá esquivar). Los bloques de piedra se mueven gracias a su componente **Animator** en el que se recoge una animación que modifica sus valores **X** o **Y** en el **transform position**. En el **Script TTLL Lagrimas** se añade además que en caso de que la salud sea cero, se cargará el nivel desde el inicio:

```
void OnTriggerStay (Collider col) {
    if (col.CompareTag ("Llamas")) {
        salud = salud - damageFuego;
    }
    if (col.CompareTag ("Golpe")) {
        salud = salud - damageGolpe;
    }
}

public void Muerte () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 0
);
}
```

Cuando el player llegue al final del castillo se cargará la cuarta escena. En ella, el jugador no podrá moverse, simplemente observar durante diez segundos las estrellas (Lágrimas de San Lorenzo). Tras esto, se dará paso a la siguiente fase por las funciones recogidas en el **Script Lagrimas Tiempo** del **GameObject Game Controller**:

```
void Update () {
    tiempoInicio = tiempoInicio + tiempo * Time.deltaTime;
    if (tiempoInicio > tiempoFinal) {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
    }
}
```

En la quinta escena, el player verá sobreimpresionado en pantalla un canvas que se eliminará cuando se pulsa la E sobre el tercer libro. En esta fase habrá dos escenarios en los que el jugador pierde vida (tal y como en la Planta Baja Parte Uno, Tecnología de la Información Escrita, Teoría de la Imagen e Historia de la Cultura Contemporánea).

- **Hamelín.** En el primer escenario, el jugador deberá completar un recorrido con ayuda de plataformas móviles (tienen un **Animator**). En las plataformas con

desplazamiento horizontal en bucle, el **Script Plataformas Ping Pong** hace que el jugador se convierta en hijo de dichas plataformas, por lo que se mueve a izquierda y derecha con ellas:

```
void OnTriggerEnter (Collider other) {
    if (other.gameObject == Player) {
        Player.transform.parent = transform;
    }
}

void OnTriggerExit (Collider other) {
    if (other.gameObject == Player) {
        Player.transform.parent = null;
    }
}
```

Cuando termine de saltar entre plataformas, el player podrá ver un circuito de velocidad con dos niños montando en bicicleta (se mueven gracias al **Animator**). Al pulsar la E en dicho circuito se activará una explosión y una plataforma rotatoria que permite el acceso al último terreno desde el cual se entenderá como completado el tercer libro.

- **El Enredo de la Bolsa y la Vida.** De vuelta en la plataforma inicial, el player podrá acceder al cuarto y último escenario, en el que deberá encontrar la salida de un laberinto. Inicialmente, estará bloqueada por un fuego que desaparecerá cuando el jugador encuentre cinco imágenes. Habrá un canvas en la esquina superior derecha con la puntuación actualizada (las siguientes funciones vienen recogidas en el **Script Sumar Enredo** del player):

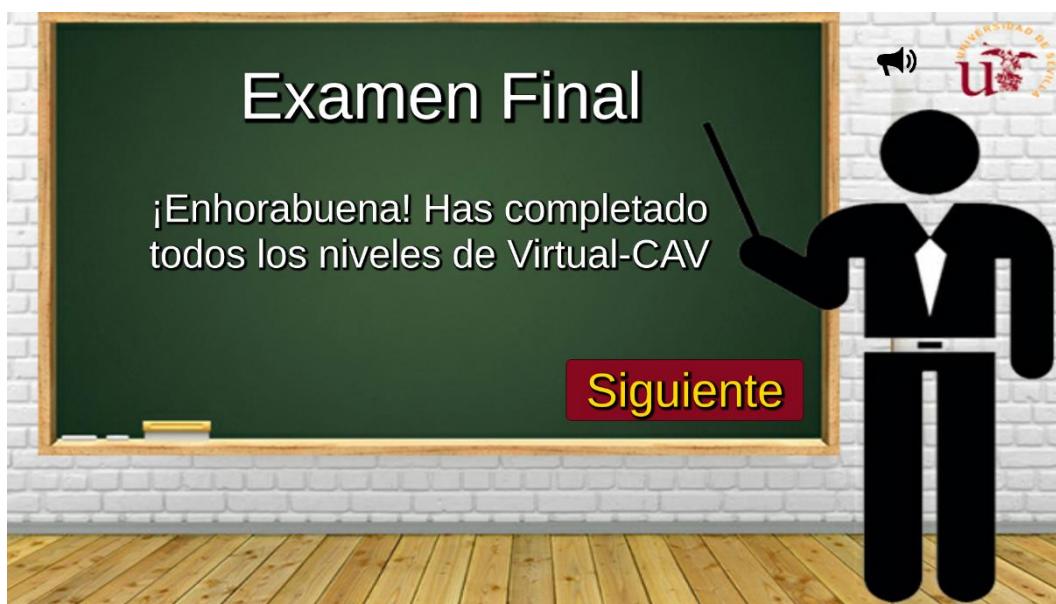
```
void Update () {
    puntuacion5.text = puntuacion + puntuacion0 + puntuacion1 + puntuacion2 + puntuacion3 + puntuacion4 + "/5";
    if (puntuacion5.text == "5/5") {
        Destroy (puertas.gameObject);
        sonido2.GetComponent< AudioSource > ().Play ();
        tiempoCanvas = true;
    }
}

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Uno")) {
        añadirPuntuacion = true;
        Destroy (uno.gameObject);
```

```
        puntuacion0 = 1;
        sonido.GetComponent< AudioSource > ().Play ();
    }
}
```

3.B.3 TRANSICIONES

Las transiciones son pequeñas escenas que sirven como hilo conductor en el desarrollo de Virtual-CAV. Todas están introducidas al inicio de cada uno de los niveles y explican al jugador cuál es el objetivo a completar, como si se tratara de un profesor de la facultad que está impartiendo una lección. Por este motivo, todas mantienen el mismo diseño a nivel visual y programático. En concreto, cuando el Player accede a una de estas transiciones, puede ver la siguiente imagen:



- **El título:** Que sirve para hacer referencia a la asignatura a la que el jugador va a acceder.
- **El cuerpo del texto:** Que explica la prueba e introduce al alumno en la materia correspondiente.
- **El diseño de la imagen:** Que simula la pizarra dentro de las aulas de la Facultad de Comunicación con esa pared blanca trasera y la tarima en los pies del profesor/a.
- **Botones:** Sirven para seguir leyendo las instrucciones o controlar el sonido de la escena.

De esta forma, Virtual-CAV cuenta con un total de siete de transiciones y una especial denominada 'Examen Final'.

Programación

Las principales funciones programáticas de estas escenas se hacen a través del método **On Click ()** de los distintos **UI Botones**. Al clickar, se activan y desactivan los **GameObjects Empty** que actúan como carpetas contenedoras del texto (gracias al **True** y el **False** de los **GameObject.SetActive ()**). El botón del altavoz, además, define el **AudioSource** general (con **.Play ()** y **.Pause ()**). La última función disponible en todas las transiciones es la que permite cargar la siguiente escena, gracias al **Script Transicion Asignaturas** del GameObject Game Controller:

```
public void PlayGame () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
); }
```

La única escena que introduce funciones programáticas más avanzadas es la del examen final. En ella, cada botón tendrá asociado un sonido (acierto o fallo, tal y como se hacía en El Retablo de las Maravillas) y actualizará una puntuación. Después de contestar la última pregunta, aparecerá el total de aciertos. En función del número de puntos, se cargará una pantalla con información sobre el suspenso (menos de siete aciertos) o el aprobado (siete o más puntos). En ambos casos el player podrá volver a hacer el examen o comenzar a hacer las asignaturas de nuevo. En caso de aprobado, el jugador tendrá un tercer botón que le permitirá acceder a la planta cuatro de la Facultad de Comunicación:

```
void Update () {
    if (puntosText.text == "10" || puntosText.text == "9" || puntosText.t
ext == "8" || puntosText.text == "7" && pizarra4.gameObject.activeInHierarchy
== true) {
        siguienteAprobado.gameObject.SetActive (true);
        siguienteSuspensos.gameObject.SetActive (false);
```

```

        }
        if (puntosText.text == "0" || puntosText.text == "1" || puntosText.te
xt == "2" || puntosText.text == "3" && pizarra4.gameObject.activeInHierarchy
== true) {
            siguienteSuspenso.gameObject.SetActive (true);
            siguienteAprobado.gameObject.SetActive (false);
        }
        if (puntosText.text == "4" || puntosText.text == "5" || puntosText.te
xt == "6" && pizarra4.gameObject.activeInHierarchy == true) {
            siguienteSuspenso.gameObject.SetActive (true);
            siguienteAprobado.gameObject.SetActive (false);
        }
    }

    public void AddScore (int newScoreValue) {
        puntos += newScoreValue;
        UpdateScore ();
    }

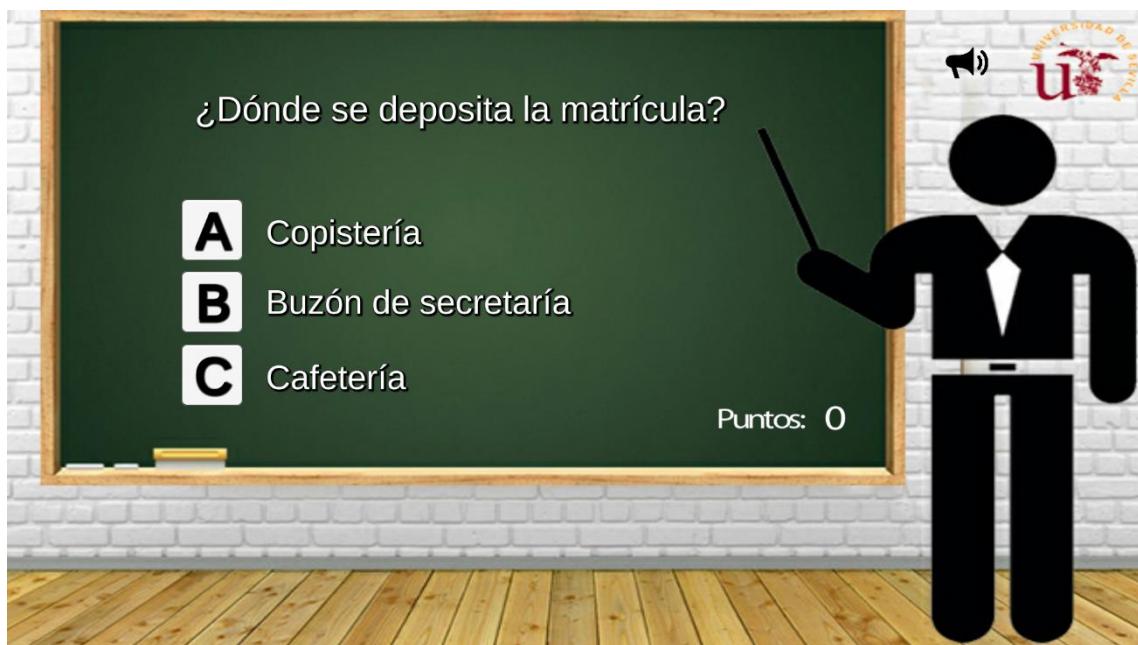
    public void UpdateScore () {
        puntosText.text = "" + puntos;
    }

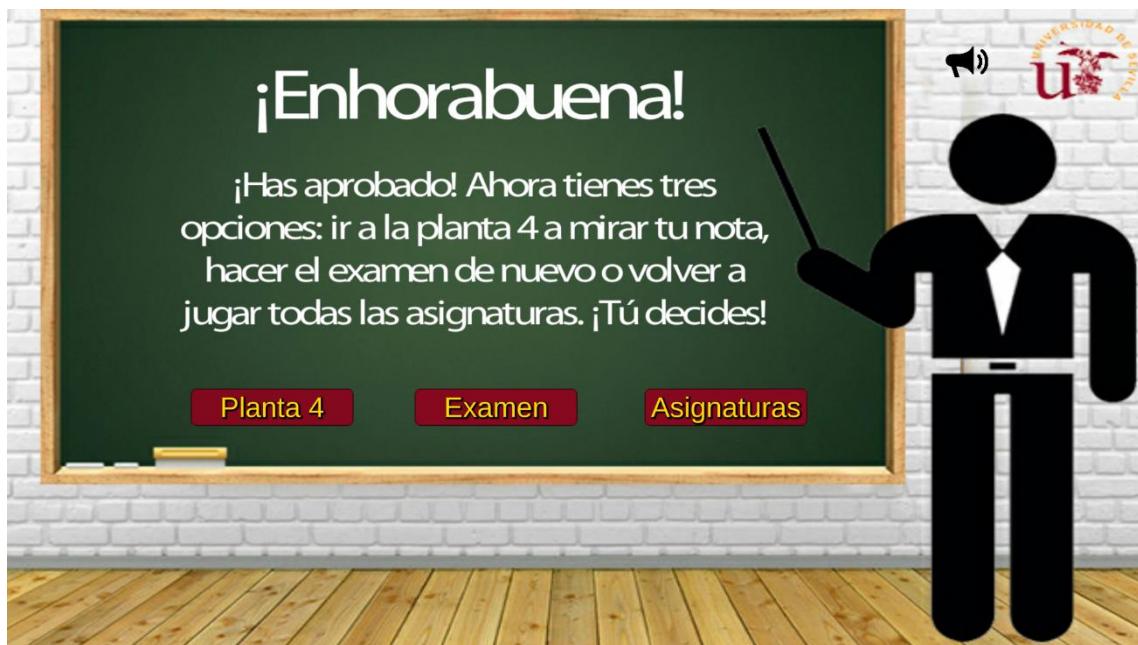
    public void Victoria () {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
);
    }

    public void Examen () {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 0
);
    }

    public void Asignaturas () {
        SceneManager.LoadScene ("_TransicionTIE");
    }
}

```





3.B.4 OTROS

En esta categoría entran todas las escenas que no están relacionadas con los anteriores bloques. En concreto, los niveles son:

- **Inicio:** Es la primera pantalla de Virtual-CAV y sirve para dar comienzo al videojuego.
- **Créditos:** La última escena que aporta los nombres de los realizadores y colaboradores de este proyecto.
- **Ascensor:** Es la pantalla que aparece al entrar en alguno de los elevadores de la Universidad y sirve al jugador para moverse entre las plantas.

Todas estas escenas utilizan la misma función de botones que las transiciones, pero tienen algunas especificaciones que se comentarán a continuación.

3.B.4.1 Inicio

Diseño

Esta escena tiene un diseño totalmente propio. La fotografía de fondo se realizó a la fachada de la Facultad de Comunicación de forma vertical y después se editó en Photoshop para que ganase en gama de colores, perspectiva y calidad. Además, se insertaron los logotipos de Virtual-CAV y Fcom.



Por otro lado, los diferentes botones utilizan una tipografía propia⁸ y cada uno cumple una función:

- **¡Jugar!:** Inicia el videojuego.
- **Controles:** Despliega una pantalla que explica cuáles son las teclas que utiliza Virtual-CAV para mover a su personaje principal.
- **Salir:** Permite salir del juego al alumno.



⁸ Puede consultar el apartado de tipografías en la página 87 de este documento.

Programación

En este nivel, los botones tienen funciones **On Click ()**:

- **Jugar.** Al presionar este botón, se carga la siguiente escena. Esto se consigue con un fragmento del script **MainMenu** asociado al GameController del nivel:

```
public void PlayGame () {  
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1  
);  
}
```

- **Controles.** Al presionar este botón, se carga el **Empty GameObject** que actúa como carpeta contenedora de todas las imágenes que muestran las teclas interactivas. Esto se consigue con las funciones **GameObject.SetActive**, cambiando **True** y **False**.
- **Salir.** Al presionar este botón, se sale del juego. Esto se consigue con un fragmento del script **MainMenu** asociado al GameController del nivel:

```
public void QuitGame () {  
    Debug.Log ("QUIT!");  
    Application.Quit ();  
}
```

- **Play / Stop Música.** Existen dos botones: altavoz con y sin franja roja. Mientras se observe el altavoz sin la marca, la música sonará y, al clickar sobre él, será reemplazado por el mismo altavoz, pero esta vez, teniendo la barra diagonal y la música estará silenciada. El proceso se puede realizar a la inversa. Esto es igual que en las escenas de transiciones.

3.B.4.2 Créditos

Diseño

A nivel de diseño, mantiene la misma estética que la escena de Inicio. Se han cambiado la tipografía, eliminado los logotipos de la parte superior izquierda de la fotografía e introducido estos logotipos como objetos con movimiento en los propios créditos.



Programación

Además de la barra de música (tiene un funcionamiento similar al del menú de pausa que se explica más adelante), hay una función programada para que los créditos se deslicen de arriba abajo (**Script Screen Roll**).

```
void Start () {
    rt = GetComponent<RectTransform> ();
    bajar = true;
    rt.transform.localPosition = posicionInicio;
    fin.gameObject.SetActive (false);
}

void Update () {
    if (bajar == true) {
        rt.transform.localPosition += Vector3.down;
        if (rt.localPosition == posicionFinal) {
            bajar = false;
            fin.gameObject.SetActive (true);
        }
    }
}
```

Cuando ya no aparecen en pantalla, caerá otro texto (“Fin”), que se detendrá en la mitad de la imagen durante cinco segundos (**Script Screen Roll 2**). Tras ellos, el juego terminará:

```
void Update () {
    if (bajar == true) {
        rt.transform.localPosition += Vector3.down;
        if (rt.localPosition == posicionFinal) {
            bajar = false;
            tiempoCorre = true;
        }
    }
    if (tiempoCorre == true) {
        tiempo = tiempo + masTiempo * Time.deltaTime;
        if (tiempo > tiempoFinal) {
            Debug.Log ("FINAL");
            Application.Quit ();
            tiempoCorre = false;
        }
    }
}
```

Cabe destacar que en ambos casos la posición final es definida en Unity.

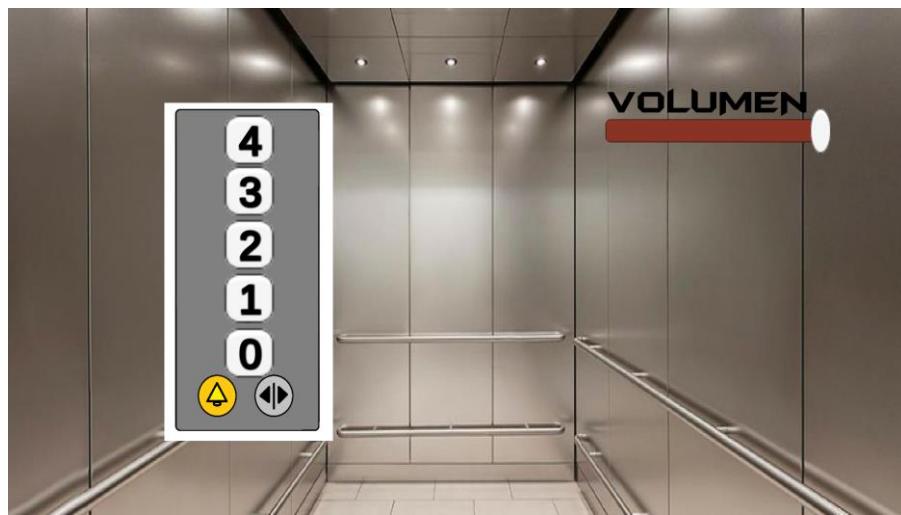
3.B.4.3 Ascensor

Diseño

Esta escena tiene dos partes a nivel de estética: El fondo y los botones del ascensor.

Por un lado, para el fondo se ha utilizado una fotografía de uso libre de la web Pixabay.

En el caso de los botones, son de realización propia a través de Photoshop. Por último, la palabra 'Volumen' tiene una tipografía propia.



Programación

Como ya se ha explicado, esta pantalla se despliega cuando el jugador se introduce en alguno de los ascensores de la Facultad de Comunicación. De esta forma, permite al Player poder acceder a todas las plantas del edificio. Tendrá a su disposición dos opciones:

- Cambiar el volumen. Se sigue un procedimiento similar al del menú de pausa (se explica más adelante).
- Cargar una planta en concreto. Se realiza gracias a la función **On Click ()** de cada uno de los botones que activa los métodos recogidos en el **Script Menu Ascensor**. Cabe destacar que el botón tres no tendrá asociada ninguna función:

```
public void Planta0 () {
    SceneManager.LoadScene ("_PlantaCero");
}

public void Planta1 () {
    SceneManager.LoadScene ("_PlantaUno");
}

public void Planta2 () {
    SceneManager.LoadScene ("_PlantaDos");
}

public void Planta4 () {
    SceneManager.LoadScene ("_PlantaCuatro");
}
```

3.C. RECURSOS

En este apartado se explicarán con detalle los objetos, imágenes, tipografía, sonidos y menús utilizados para la realización de Virtual-CAV.

3.C.1 Objetos

Los objetos mencionados a continuación se han obtenido de páginas web libres de derecho de material 3D. La lista es la siguiente⁹:

Universidad

- Impresora (1)
- Mesas (6)
- Sillas (6)
- Macetas (2)
- Papeleras (3)
- Extintores (2)
- Puertas (5)
- Buzones (2)
- Elementos de Cafetería (8)
- Ventanas (2)
- Monitores (2)
- Pizarras (2)
- Tablones de anuncios (2)
- Banco (1)
- Ordenadores (3)

⁹ La lista indica el objeto y los diferentes tipos del mismo elemento que se han descargado.

- Ascensores (2)
- Libros (4)

Teoría de la Publicidad y Relaciones Públicas

- Naves (2)
- Camiseta (1)
- Gorra (1)
- Zapatilla (1)
- Símbolo del Dólar (1)
- Marciano (1)

Teoría de la Imagen

- Cámara (1)
- Objetivo (1)
- Lente (1)
- Reloj (1)
- Casa de Madera (1)

Historia de la Cultura Contemporánea

- Club Night (1)
- Cama deshecha (1)
- Cofre (1)
- Pistola (1)
- Sofás (2)
- Edificio (1)
- Recepción (1)

- Lámparas (3)

Tendencias Literarias

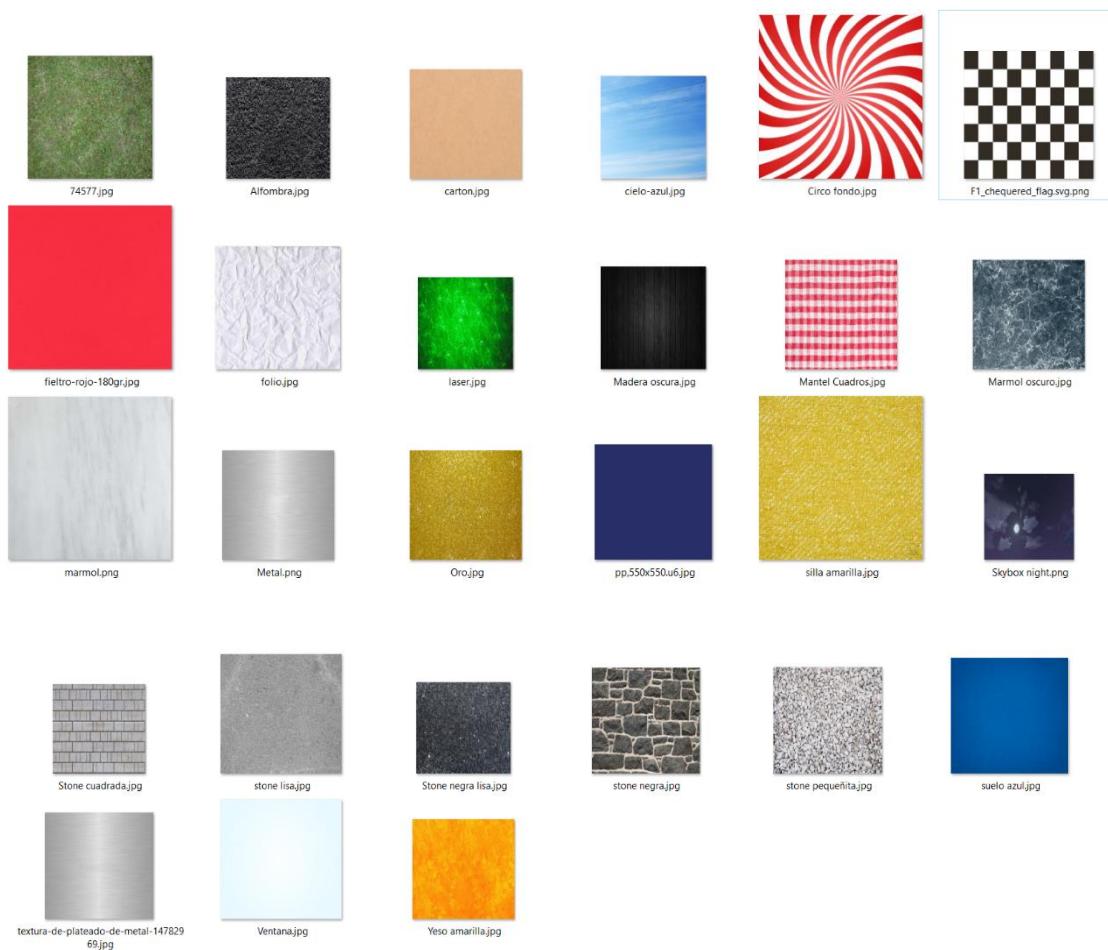
- Fuego (1)
- Panteón (1) (De este objeto se utilizaron solo las columnas)
- Antorcha (1)
- Montaña (1)
- Dragón (1)
- Columna científica (1)
- Estrella (1)
- Carpa de circo (1)
- Cabezas de bufón (2)
- Niños (2)
- Bicicletas (2)
- Adulto (1)
- Flauta (1)
- Piezas de puzzle (1)
- Tobogán (1)
- Triciclo (1)
- Barcos (2)

Todos estos elementos se han obtenido en formato .fbx a través de las siguientes páginas web:

- <https://www.turbosquid.com>
- <https://free3d.com/>
- <https://archive3d.net>

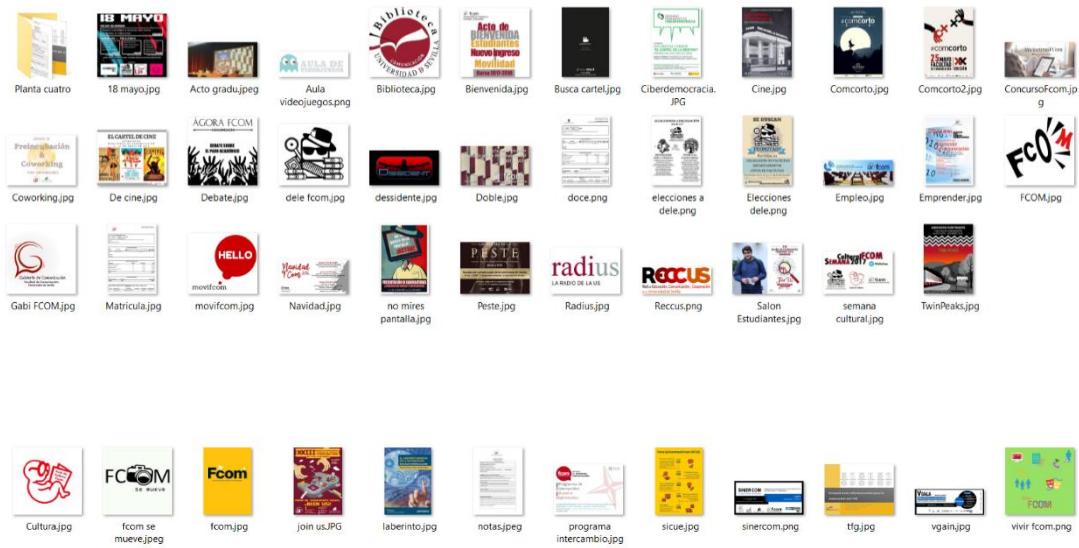
3.C.2 Imágenes

Para texturar los elementos en Unity se utilizan imágenes. Todas estas se han obtenido de la página web de imágenes de licencias gratuitas Pixabay.com. El listado es el siguiente:

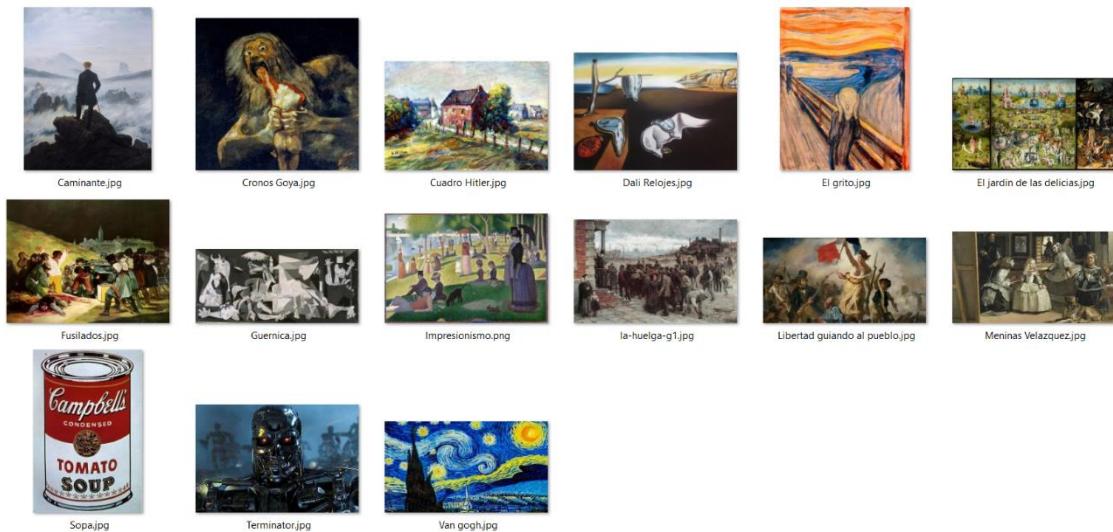


15

De la misma forma, se han utilizado fotografías para los carteles y cuadros. Estos son los carteles que aparecen en la Facultad de Comunicación:



Por otro lado, los cuadros utilizados son:



Por último, se han utilizado imágenes de uso libre como libros o folios que ilustran el momento en el que el jugador obtiene un objeto importante durante el transcurso de las pruebas. Estos elementos se han obtenido de la web Pixabay.com.

3.C.3 Tipografías

En este proyecto se han utilizado distintos tipos de tipografías para todos los textos que aparecen durante el transcurso del videojuego. Un total de 9 estilos de letra diferentes. Las distintas tipografías se han obtenido de la página web Dafont, comprobando que los derechos son libres para uso comercial, y de la página web Open Font Library. Los nombres de estas letras son:

- Some Time Later
- Xolonium
- GreatVibes
- Unique
- Roboto
- Myriad
- Kenyan Coffee
- Game Over
- Players

3.C.4 Música y Sonidos FX

Virtual CAV tiene un total de 24 canciones diferentes en sus escenas. Todas las grandes melodías se han obtenido de la página web Bendsound.com de música gratuita. A su vez, la canción que suena en la Pantalla Principal y en la planta cuatro parte uno son dos obras del productor Jesús Navarro.

Por otro lado, los efectos de sonido se han obtenido de los canales de YouTube sin copyright 'FX Sounds' y 'Banana Sounds'. De esta forma, tenemos una librería de efectos musicales para todas estas situaciones:

- Comer una caña de chocolate.
- Abrir una puerta normal.
- Abrir una puerta automática.
- Abrir una puerta de ascensor.
- Conseguir un objeto importante.
- Conseguir superar una parte de la prueba.
- Pasar un nivel.
- Fracasar.
- Morir.
- Disparos de la nave espacial aliada y enemiga.
- Cronómetro de reloj.

3.C.5 Menú de pausa

El menú de pausa sirve como herramienta para que el jugador pueda tener control sobre elementos del escenario como el volumen. También incluye pistas sobre lo que debe hacer. Y además es la herramienta que permite al alumno salir de Virtual-CAV.

El diseño del menú es de elaboración propia y ha sido realizado con Photoshop. Se utilizó la tipografía 'Playerss' para los botones y la tipografía 'GreatVibes' para los textos que se desarrollan en su interior.



La programación del menú de pausa combina el uso de la tecla P con las funcionalidades de los botones (elementos de UI). Como ocurre en otras secciones de Virtual-CAV, se recurre a las propiedades **On Click ()** para cambiar los valores (true y false) de los **SetActive** de los distintos submenús de pausa: submenú de opciones, objetivos y salir. Clickar sobre “jugar” produciría el mismo efecto que pulsar la P: deshacer la pausa. En el apartado de “opciones”, se despliegan dos sliders de volumen que cambian el nivel de audio de la canción de fondo o de los efectos del player (sonidos de caminar y saltar). “Objetivos” muestra una pequeña guía para orientar al jugador en el nivel. Por último, “salir” presenta al player la cuestión sobre si quiere o no dejar el juego (advirtiendo que en caso de abandono, deberá empezar desde un

principio). Se muestran dos botones (“sí” y “no”) cuya interacción provoca el fin del juego o el retorno al menú de pausa. Estas funciones se recogen en el **Script Game Pause** que se inserta en el **GameObject Game Controller**, objeto disponible en todos los niveles que no pertenecen a la categoría de “transiciones” y “otros”:

```

public Transform canvas;
public Transform Player;
public Transform pauseMenu;
public Transform controlsMenu;
public Transform quehacerMenu;
public Transform salirMenu;

void Start () {
    canvas.gameObject.SetActive (false);
    pauseMenu.gameObject.SetActive (false);
    controlsMenu.gameObject.SetActive (false);
    quehacerMenu.gameObject.SetActive (false);
    salirMenu.gameObject.SetActive (false);
}

void Update () {
    if (Input.GetKeyDown (KeyCode.P)) {
        Pause ();
    }
}

void OnGUI () {
    Cursor.lockState = CursorLockMode.None;
    Cursor.visible = true;
}

public void QuitGame () {
    Debug.Log ("SALIR DEL JUEGO");
    Application.Quit ();
}

public void Controls (bool Open) {
    if (Open) {
        controlsMenu.gameObject.SetActive (true);
        pauseMenu.gameObject.SetActive (false);
    }
    if (!Open) {
        controlsMenu.gameObject.SetActive (false);
        pauseMenu.gameObject.SetActive (true);
    }
}

public void QueHacer (bool Open) {
    if (Open) {
        quehacerMenu.gameObject.SetActive (true);
        pauseMenu.gameObject.SetActive (false);
    }
    if (!Open) {
        quehacerMenu.gameObject.SetActive (false);
    }
}

```

```

        pauseMenu.gameObject.SetActive (true);
    }
}

public void Pause () {
    if (canvas.gameObject.activeInHierarchy == false) {
        canvas.gameObject.SetActive (true);
        Time.timeScale = 0;
        Player.GetComponent<FirstPersonController> ().enabled = false;
        if (pauseMenu.gameObject.activeInHierarchy == false) {
            pauseMenu.gameObject.SetActive (true);
            controlsMenu.gameObject.SetActive (false);
            quehacerMenu.gameObject.SetActive (false);
            salirMenu.gameObject.SetActive (false);
        }
    } else {
        canvas.gameObject.SetActive (false);
        Time.timeScale = 1;
        Player.GetComponent<FirstPersonController> ().enabled = true;
    }
}

public void Salir (bool Open) {
    if (Open) {
        salirMenu.gameObject.SetActive (true);
        pauseMenu.gameObject.SetActive (false);
    }
    if (!Open) {
        salirMenu.gameObject.SetActive (false);
        pauseMenu.gameObject.SetActive (true);
    }
}

```

Cabe destacar que las funciones de subida y bajada de volumen se encuentran recogidas en el **Script Music Volume**, también inserto en el **GameObject Game Controller**:

```

public Slider musicVolume;
public AudioSource myMusic;
public Slider fxVolume;
public AudioSource fxMusic;

void Update () {
    myMusic.volume = musicVolume.value;
    fxMusic.volume = fxVolume.value;
}

```

3.D GITLAB

Una de las mayores problemáticas que presentaba el Trabajo de Fin de Grado conjunto de dos alumnos era el modo de compartir los avances que cada uno realizara de manera individual. Era sencillo asignar a cada uno las tareas, pero no resultaba tan fácil compartir los resultados. Aquí, el tutor José Luis Navarrete Cardero propone como alternativa el trabajo mediante Git.

Esta web permite a las personas abrir una cuenta para subir y actualizar archivos entre usuarios de una carpeta máster. A continuación, se detallan las principales funciones que se han utilizado:

- **Git status.** Enseña los archivos creados o modificados desde la última actualización (en rojo).
- **Git add.** Acompañando este comando de una carpeta, añade todos los archivos que contiene dicha carpeta. En Unity, resulta fundamental añadir en cada actualización Assets y, en casos concretos, ProjectSettings. Si se vuelve a ejecutar Git Status, los archivos pertenecientes a las carpetas añadidas aparecerán en verde
- **Git commit.** Acompañando este comando por *-m* y un nombre entre comillas agrupa todos los archivos añadidos en el paso anterior en una actualización.
- **Git push.** Con este procedimiento, cada usuario sube a la carpeta Máster el/los commit/s.
- **Git pull.** Este comando permite descargar al otro usuario los commits pusheados.

Es importante tener una comunicación fluida entre los usuarios. De otra manera, si dos personas realizan cambios a la vez en una misma escena sin que estén actualizados con Git pull y Git push se producen merge conflicts.

Uno de los principales problemas que se tuvieron mientras se realizaba este proyecto estuvo relacionado con este último apartado. Elaborando la escena de Historia de la Cultura Contemporánea, se produjeron merge conflicts en el proyecto debido a que se realizaron cambios simultáneos dentro de la escena. El resultado fue que tanto ese nivel como la planta baja de la Facultad de Comunicación quedaron eliminados.

Para la resolución de este conflicto, se dedicó más de una semana de trabajo:

- En un primer lugar, se intentó solucionar de forma autodidacta fracasadamente. Tras consultar muchos foros de expertos en este programa, no se encontró el mecanismo de solucionar esos merge conflicts.
- Tras muchos intentos, se acudió a pedir ayuda al profesor José Luis Navarrete. En ese momento, el tutor explicó que Git comprende en su programación la posibilidad de volver a una versión anterior de los denominados commits (los ficheros que contienen los datos de las diferentes subidas del proyecto que se han hecho a la página web).
- Con esta información, se descubrió el uso del comando git reset, una combinación que permite volver a una versión anterior del proyecto, pero de forma local. Esto quería decir que se podía observar que las escenas Historia de la Cultura Contemporánea y planta baja seguían dentro de Git, sin embargo no se tenía acceso de forma online a ellas.

- El uso de diferentes comandos como git reset hard (reinicio forzado), git reset soft (reinicio controlado) y la búsqueda en multitud de páginas web fueron los pasos previos al encuentro de la siguiente información en uno de los foros:

On the same settings page you can also allow developers to push into the protected branches. With this setting on, protection will be limited to rejecting operations requiring `git push --force` (rebase etc.)

Since GitLab 9.3

Go to project: "Settings" -> "Repository" -> "Expand" on "Protected branches"

Protected branch (1)	Last commit	Allowed to merge	Allowed to push
master (default)	b1cbeef47 4 minutes ago	Masters	Masters

- Esta imagen explica que, de forma genérica, Git tiene protegida la raíz máster del proyecto por lo que no se puede volver a versiones anteriores. Para solucionar el problema, lo único que se tuvo que hacer fue desbloquear esta opción, hacer el comando git reset hard para volver al commit (versión) del proyecto que se quería recuperar y hacer un git push hard (una subida forzada) de dicho commit. De esta forma, el proyecto recuperó en Git las escenas Historia de la Cultura Contemporánea y planta baja.

4- CONCLUSIONES

Aunque el desarrollo final del videojuego muestre serias diferencias en relación con el guion original, Virtual-CAV se presenta como un producto que resuelve satisfactoriamente la necesidad básica con la que se crea: proporcionar información de manera adecuada y atractiva a los posibles estudiantes del Grado en Comunicación Audiovisual. El videojuego tiene capacidad para introducirse en la web de la Facultad de Comunicación, tras la obtención de los permisos y requerimientos específicos.

Por otro lado, la programación del videojuego no es cerrada, sino que está abierta a futuras modificaciones que hacen posibles proyectos como Virtual-PER, Virtual-PUB y Virtual-DG, entendiendo estas ideas como la adaptación del formato de Virtual-CAV al resto de titulaciones disponibles en la Facultad de Comunicación: los grados en Periodismo, Publicidad y Relaciones Públicas y el Doble Grado en Periodismo y Comunicación Audiovisual. Existiría la posibilidad de recrear las asignaturas presentes en todos los planes de estudios o aquellas que presenten un grado de similitud considerable. Otra opción sería adaptar la esencia de las escenas de Virtual-CAV (formatos como el escape-room, space shooter o niveles con pérdida de vida con el paso del tiempo) a las materias del resto de grados. En todos estos proyectos resultaría fundamental la constante renovación en función de las modificaciones en los planes de estudio.

Cabe destacar que Virtual-CAV permite, además, exportar exclusivamente las escenas correspondientes a las plantas de la Facultad de Comunicación. De este modo, se puede hacer un recorrido virtual por el edificio. Esta idea se puede aplicar a todas las facultades de la Universidad de Sevilla, lo que resultaría innovador: podría poner a disposición de los futuros estudiantes de sus grados una herramienta muy útil para conocer sus edificios.

5- BIBLIOGRAFÍA

- Pixabay (2018). Maravillosas imágenes gratis. Pixabay.com. Recuperado de <https://pixabay.com/>
- Free 3D (2018). Free 3D Models. Free3d.com. Recuperado de <https://free3d.com/>
- Turbosquid. 3D (2018). Models for professionals. Recuperado de <https://www.turbosquid.com/>
- Archive 3D (2007-2018). Free 3D Models. Archive3D.net. Recuperado de <https://archive3d.net/>
- Dafont (2018). Fuentes añadidas. Dafont.com. Recuperado de <https://www.dafont.com/es/>
- OpenFont (2018). Font Library. Fontlibrary.org. Recuperado de <https://fontlibrary.org/es>
- M. Berengueras, Josep (2018). La industria del videojuego reclama el mismo trato que el cine y la música. El Periódico, Barcelona. Recuperado de <https://www.elperiodico.com/es/ocio-y-cultura/20180827/videojuegos-cine-musica-cultura-comparativa-7001883>
- Facultad de Comunicación (2018). Plan de Estudios de Comunicación Audiovisual. Universidad de Sevilla. Recuperado de http://www.us.es/estudios/grados/plan_192?p=7
- Mayorga, Juan (2004). “*Hamelín*”. Madrid, Ñaque.
- Boadella, Albert (2004). “*El Retablo de las maravillas: cinco variaciones de Cervantes*”. Alicante, Biblioteca Virtual Miguel de Cervantes.
- Mendoza, Eduardo (2012). “*El Enredo de la bolsa y la vida*”. Barcelona, Seix Barral.
- Llamazares, Julio (2013). “*Las lágrimas de San Lorenzo*”. Madrid, Alfaguara.
- Unity, Documentation (2018). Space Shooter tutorial. Unity Technologies. Unity3d.com. Recuperado de <https://unity3d.com/es/learn/tutorials/s/space-shooter-tutorial>
- Unity, Documentation (2016). Controlando GameObjects utilizando Componentes. Unity3d.com. Recuperado de

<https://docs.unity3d.com/es/current/Manual/ControllingGameObjectsComponents.html> Consultado el 4 de octubre de 2018.

- Unity, Documentation (2018). Mathf.PingPong. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/Mathf.PingPong.html> Consultado el 4 de octubre de 2018.
- Unity, Documentation (2016). Mathf.PingPong. Unity3d.com. Recuperado de <https://docs.unity3d.com/540/Documentation/ScriptReference/Mathf.PingPong.html> Consultado el 4 de octubre de 2018.
- Unity, Documentation (2018). Resolving file conflicts. Unity3d.com. Recuperado de <https://docs.unity3d.com/Manual/UnityCollaborateResolvingConflicts.html> Consultado el 8 de octubre de 2018.
- Unity, Documentation (2018). Cursor.visible. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/Cursor-visible.html> Consultado el 22 de octubre de 2018.
- Unity, Documentation (2016). Bloqueo del cursor y modo de pantalla completa en WebGL. Unity3d.com. Recuperado de <https://docs.unity3d.com/es/current/Manual/webgl-cursorfullscreen.html> Consultado el 22 de octubre de 2018.
- Unity, Documentation (2016). Solución de Problemas (Troubleshooting) del Editor. Unity3d.com. Recuperado de <https://docs.unity3d.com/es/current/Manual/TroubleShootingEditor.html> Consultado el 22 de octubre de 2018.
- Unity, Documentation (2016). Administrador del Tiempo y Framerate. Unity3d.com. Recuperado de <https://docs.unity3d.com/es/current/Manual/TimeFrameManagement.html> Consultado el 22 de octubre de 2018.
- Unity, Documentation (2018). Button. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/UI.Button.html> Consultado el 22 de octubre de 2018.
- Unity, Documentation (2018). Cursor.lockState. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/Cursor-lockState.html> Consultado el 22 de octubre de 2018.

- Unity, Documentation (2018). Publishing Builds. Unity3d.com. Recuperado de <https://docs.unity3d.com/Manual/PublishingBuilds.html> Consultado el 24 de octubre de 2018.
- Unity, Documentation (2018). Build Settings. Unity3d.com. Recuperado de <https://docs.unity3d.com/Manual/BuildSettings.html> Consultado el 24 de octubre de 2018.
- Unity, Documentation (2018). Physics.SphereCast. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/Physics.SphereCast.html> Consultado el 25 de octubre de 2018.
- Unity, Documentation (2018). WaitForSeconds. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/ WaitForSeconds.html> Consultado el 5 de noviembre de 2018.
- Unity, Documentation (2018). RectTransform. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/RectTransform.html> Consultado el 14 de noviembre de 2018.
- Unity, Documentation (2018). Mathf.RoundToInt. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/Mathf.RoundToInt.html> Consultado el 14 de noviembre de 2018.
- Unity, Documentation (2018). GameObject.GetComponent. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/GameObject.GetComponent.html> Consultado el 14 de noviembre de 2018.
- Unity, Documentation (2018). GameObject.GetComponent. Unity3d.com. Recuperado de <https://docs.unity3d.com/ScriptReference/GameObject.GetComponent.html> Consultado el 14 de noviembre de 2018.
- Van Dugteren, David (20 de octubre de 2010). How can I recover from an erroneous git push -f origin master? Stackoverflow.com. Recuperado de <https://stackoverflow.com/questions/3973994/how-can-i-recover-from-an-erroneous-git-push-f-origin-master> Consultado el 17 de noviembre de 2018

6- ANEXOS

En el apartado de “anexos” se presentan los scripts escritos, acompañados por la principal función que producen:

Bolas Verdes. Recoge las funciones de la Planta Baja Parte Uno (matrícula y recuperar y perder vida):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class BolasVerdes : MonoBehaviour {

    public GameObject roja1;
    public GameObject roja2;
    //public GameObject verde1;
    //public GameObject verde2;
    public GameObject matriculaCubo;
    public GameObject matriculaCanvas;
    public GameObject informacionCanvas;
    private bool canDestroy;
    public Slider barraSalud;
    public float salud;
    public float damage;
    public bool menosSalud;
    public GameObject muerteObjeto;
    public GameObject masSaludObjeto;
    public GameObject vidaObjeto;
    public GameObject informacionCafeteriaCanvas;
    public bool masSalud;
    public GameObject musicFXCaña;
    public GameObject musicFXConseguir;
    public float speed;

    // Use this for initialization
    void Start () {
        matriculaCubo.gameObject.SetActive (true);
        matriculaCanvas.gameObject.SetActive (false);
        canDestroy = false;
        salud = 1;
        barraSalud.value = salud;
        menosSalud = true;
        muerteObjeto.gameObject.SetActive (true);
        masSaludObjeto.gameObject.SetActive (false);
        vidaObjeto.gameObject.SetActive (true);
        informacionCafeteriaCanvas.gameObject.SetActive (false);
        masSalud = false;
    }

    // Update is called once per frame
```

```

void Update () {
    if (Input.GetKeyDown(KeyCode.E) && canDestroy) {
        Destroy (matriculaCanvas);
        Destroy (roja1.gameObject);
        canDestroy = false;
    }
    if (Input.GetKeyDown(KeyCode.E) && masSalud) {
        Destroy (muerteObjeto.gameObject);
        masSaludObjeto.gameObject.SetActive (true);
        Destroy (roja2.gameObject);
        musicFXCaña.GetComponent< AudioSource > ().Play ();
        Destroy (informacionCafeteriaCanvas.gameObject);
        masSalud = false;
    }
    barraSalud.value = salud;
    if (barraSalud.value == 0) {
        Destroy (gameObject);
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
- 1);
    }
    if (barraSalud.value == 1) {
        masSaludObjeto.gameObject.SetActive (false);
        vidaObjeto.gameObject.SetActive (false);
    }
    if (roja1.gameObject == null && roja2.gameObject == null) {
        SiguienteNivel ();
    }
}

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Matricula")) {
        //Destroy (roja2.gameObject);
        matriculaCubo.gameObject.SetActive (false);
        matriculaCanvas.gameObject.SetActive (true);
        musicFXConseguir.GetComponent< AudioSource > ().Play ();
    }
    if (col.CompareTag ("Buzon")) {
        if (matriculaCanvas.gameObject.activeInHierarchy == true) {
            informacionCanvas.gameObject.SetActive (true);
            canDestroy = true;
        }
    }
    if (col.CompareTag ("Vida")) {
        informacionCafeteriaCanvas.gameObject.SetActive (true);
        masSalud = true;
    }
}

void OnTriggerExit (Collider col) {
    if (col.CompareTag ("Buzon")) {
        informacionCanvas.gameObject.SetActive (false);
        canDestroy = false;
    }
    if (col.CompareTag ("Vida")) {
        informacionCafeteriaCanvas.gameObject.SetActive (false);
    }
}

void OnTriggerStay (Collider col) {

```

```

        if (col.CompareTag ("muerte") && menosSalud) {
            salud = salud - damage * Time.deltaTime;
        }
        if (col.CompareTag ("MasSalud")) {
            menosSalud = false;
            salud = salud + speed * Time.deltaTime;
        }
    }

    public void SiguienteNivel () {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
    );
}
}

```

Boss Eliminado. Recoge funciones relacionadas con el enemigo final de la escena de Teoría de la Publicidad y las Relaciones Públicas:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class BossEliminado : MonoBehaviour {

    public GameObject boss;
    public GameObject victoria;
    public GameObject victoriaSonido;
    public GameObject musicaBackground;

    //intento de temporizador
    public float tiempoInicio;
    public float tiempoMas;
    public float tiempoFinal;
    public bool tiempoMuerte;

    // Use this for initialization
    void Start () {
        tiempoInicio = 0;
        tiempoFinal = 2;
        tiempoMuerte = false;
        tiempoMas = 0;
    }

    // Update is called once per frame
    void Update () {
        if (boss.gameObject == null) {
            musicaBackground.GetComponent< AudioSource > ().Stop ();
            tiempoMas = 1;
            tiempoInicio = tiempoInicio + tiempoMas * Time.deltaTime;
            if (tiempoInicio > tiempoFinal) {
                victoria.gameObject.SetActive (true);
                victoriaSonido.GetComponent< AudioSource > ().Play ();
                Time.timeScale = 0;
            }
        }
    }
}

```

```

    }

    public void Ganaste () {
        Time.timeScale = 1;
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
    );
    }
}

```

Cae Cuadro. Contiene funciones de la escena de Historia de la Cultura

Contemporánea como activar la animación de caída del cuadro y activar la llave:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CaeCuadro : MonoBehaviour {

    public GameObject trigger;
    public GameObject cuadro;
    public GameObject llave;

    Animator CajonAbre;

    // Use this for initialization
    void Start () {
        CajonAbre = cuadro.GetComponent <Animator> ();
        llave.gameObject.SetActive (false);
    }

    void OnTriggerEnter (Collider coll) {
        if (coll.gameObject.tag == "Player") {
        }
    }

    void OnTriggerStay (Collider coll) {
        if (coll.gameObject.tag == "Player" && Input.GetKeyDown (KeyCode.E))
    {
        OpenTheDrawer (true);
        llave.gameObject.SetActive (true);
    }
}

    void OnTriggerExit (Collider coll) {
        if (coll.gameObject.tag == "Player") {
            OpenTheDrawer (false);
        }
    }

    void OpenTheDrawer (bool state) {
        CajonAbre.SetBool ("position", state);
    }

    // Update is called once per frame
    void Update () {

```

```
        }
    }
```

Destroy By Boundary. Destruye los game objects que abandonen el escenario de

Teoría de la Publicidad y las Relaciones Públicas:

```
using UnityEngine;
using System.Collections;

public class DestroyByBoundary : MonoBehaviour {

    void OnTriggerExit(Collider other) {
        Destroy (other.gameObject);
    }
}
```

Destroy By Contact. Destruye los game objects mediante la interacción con otro game object en Teoría de la Publicidad y las Relaciones Públicas:

```
using UnityEngine;
using System.Collections;

public class DestroyByContact : MonoBehaviour {

    public GameObject explosion;
    public GameObject playerExplosion;
    public int scoreValue;
    private GameController gameController;

    void Start () {
        GameObject gameControllerObject = GameObject.FindWithTag ("GameController");
        if (gameControllerObject != null) {
            gameController = gameControllerObject.GetComponent<GameController> ();
        }
        if (gameController == null)
            Debug.Log ("Cannot find 'GameController' script");
    }

    void OnTriggerEnter(Collider other) {
        if (other.tag == "Boundary") {
            return;
        }
        Instantiate(explosion, transform.position, transform.rotation);
        if (other.tag == "Player") {
            Instantiate (playerExplosion, other.transform.position, other.transform.rotation);
            gameController.GameOver ();
        }
        gameController.AddScore (scoreValue);
        Destroy (other.gameObject);
        Destroy (gameObject);
    }
}
```

```
    }
}
```

Destroy By Time. Destruye los game objects mediante el paso del tiempo en Teoría de la Publicidad y las Relaciones Públicas:

```
using UnityEngine;
using System.Collections;

public class DestroyByTime : MonoBehaviour {

    public float lifeTime;

    void Start () {
        Destroy (gameObject, lifeTime);
    }
}
```

Disparar Enemigo. Recoge funciones relacionadas con el enemigo final de la escena de Teoría de la Publicidad y las Relaciones Públicas:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DisparaEnemigo : MonoBehaviour {

    public GameObject shot1;
    public GameObject shot2;
    public Transform boss1;
    public Transform boss2;
    public Transform boss3;
    private AudioSource fxDisparo;
    public float delay1;
    public float fireRate1;
    public float delay2;
    public float fireRate2;

    void Awake () {
        fxDisparo = GetComponent<AudioSource> ();
    }

    // Use this for initialization
    void Start () {
        InvokeRepeating ("Fire1", delay1, fireRate1);
        InvokeRepeating ("Fire2", delay2, fireRate2);
    }

    // Update is called once per frame
    void Update () {

    }

    void Fire1 () {
```

```

        Instantiate (shot1, boss1.position, boss1.rotation);
        Instantiate (shot1, boss2.position, boss2.rotation);
        fxDisparo.Play ();
    }

    void Fire2 () {
        Instantiate (shot2, boss3.position, boss3.rotation);
        fxDisparo.Play ();
    }
}

```

Display UI. Despliega textos:

```

using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class DisplayUI : MonoBehaviour {

    public string myString;
    public Text myText;
    public float fadeTime;
    public bool displayInfo;

    // Use this for initialization
    void Start () {
        myText.color = Color.clear;
    }

    // Update is called once per frame
    void Update () {
        FadeText ();
    }

    void OnTriggerEnter (Collider col) {
        if (col.CompareTag ("Player")) {
            displayInfo = true;
        }
    }

    void OnTriggerExit (Collider col) {
        if (col.CompareTag ("Player")) {
            displayInfo = false;
        }
    }

    void FadeText () {
        if (displayInfo) {
            myText.text = myString;
            myText.color = Color.Lerp (myText.color, Color.white, fadeTime *
Time.deltaTime);
        } else {
            myText.color = Color.Lerp (myText.color, Color.clear, fadeTime *
Time.deltaTime);
        }
    }
}

```

Escaleras. Permite cargar otras plantas de la Facultad de Comunicación:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Escaleras : MonoBehaviour {

    void OnTriggerEnter (Collider other) {
        if (other.tag == "Escaleras") {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
        }
        if (other.tag == "Rampas") {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
- 1);
        }
        if (other.tag == "Ascensores") {
            SceneManager.LoadScene ("_Ascensor");
        }
    }
}
```

Game Controller. Recoge funciones generales en las escenas de Teoría de la Publicidad y las Relaciones Públicas:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using UnityEngine.SceneManagement;

public class GameController : MonoBehaviour {

    public GameObject hazard;
    public Vector3 spawnValues;
    public int hazardCount;
    public float spawnWait;
    public float startWait;
    public float waveWait;
    public Text scoreText;
    public Text restartText;
    public Text gameOverText;
    public Text backText;
    private bool gameOver;
    private bool restart;
    //private bool back;
    private int score;
    private bool win;
    public float ganar;
    public GameObject siguiente;

    //intento de temporizador
```

```

public float tiempoInicio;
public float tiempoMas;
public float tiempoFinal;
public bool tiempoMuerte;

void Start () {
    Time.timeScale = 1;
    gameOver = false;
    restart = false;
    //back = false;
    win = false;
    restartText.text = "";
    gameOverText.text = "";
    backText.text = "";
    score = 0;
    UpdateScore ();
    StartCoroutine (Zapatillas ());
    siguiente.gameObject.SetActive (false);
    tiempoInicio = 0;
    tiempoFinal = 2;
    tiempoMuerte = false;
    tiempoMas = 0;
}

void Update () {
    if (restart) {
        if (Input.GetKeyDown (KeyCode.R)) {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 0);
        }
        //if (Input.GetKeyDown (KeyCode.B)) {
        //    SceneManager.LoadScene ("_PlantaCero");
        //}
    }
    if (win) {
        siguiente.gameObject.SetActive (true);
        Time.timeScale = 0;
    }
    if (tiempoMuerte == true) {
        tiempoMas = 1;
        tiempoInicio = tiempoInicio + tiempoMas * Time.deltaTime;
        if (tiempoInicio > tiempoFinal) {
            restartText.text = "Pulsa R para reintentar";
            //backText.text = "Pulsa B para volver a la transición";
            restart = true;
            //back = true;
            Time.timeScale = 0;
        }
    }
}

public void AddScore (int newScoreValue) {
    score += newScoreValue;
    UpdateScore ();
    if (score == ganar) {
        win = true;
    }
}

```

```

void UpdateScore () {
    scoreText.text = "" + score;
}

public void GameOver () {
    gameOverText.text = "¡Has muerto!";
    gameOver = true;
    tiempoMuerte = true;
}

IEnumerator Zapatillas () {
    yield return new WaitForSeconds (startWait);
    while (true) {
        for (int i = 0; i < hazardCount; i++) {
            Vector3 spawnposition = new Vector3 (Random.Range (-spawnValues.x, spawnValues.x), spawnValues.y, spawnValues.z);
            Quaternion spawnrotation = Quaternion.identity;
            Instantiate (hazard, spawnposition, spawnrotation);
            yield return new WaitForSeconds (spawnWait);
        }
        yield return new WaitForSeconds (waveWait);
        //if (gameOver) {

            //break;
        //}
    }
}

public void SiguienteNivel () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
);
}
}

```

Game Controller Puzzle. Contiene las funciones del nivel de El Retablo de Las Maravillas:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameControllerPuzzle : MonoBehaviour {

    void Start () {
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
    }

    // Update is called once per frame
    void Update () {

    }

    public void Acierto () {

```

```

        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
    );
}

public void Fallo () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex -
1);
}
}

```

Game Pause. Pausa el juego, aunque sus opciones necesitan las funciones **On Click** () de los botones del menú de pausa:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityStandardAssets.Characters.FirstPerson;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GamePause : MonoBehaviour {

    public Transform canvas;
    public Transform Player;
    public Transform pauseMenu;
    public Transform controlsMenu;
    public Transform quehacerMenu;
    public Transform salirMenu;

    void Start () {
        canvas.gameObject.SetActive (false);
        pauseMenu.gameObject.SetActive (false);
        controlsMenu.gameObject.SetActive (false);
        quehacerMenu.gameObject.SetActive (false);
        salirMenu.gameObject.SetActive (false);
        //Cursor.visible = true;
    }

    // Update is called once per frame
    void Update () {
        if (Input.GetKeyDown (KeyCode.P)) {
            Pause ();
        }
    }

    void OnGUI () {
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }

    public void QuitGame () {
        Debug.Log ("SALIR DEL JUEGO");
        Application.Quit ();
    }

    public void Controls (bool Open) {

```

```

        if (Open) {
            controlsMenu.gameObject.SetActive (true);
            pauseMenu.gameObject.SetActive (false);
        }
        if (!Open) {
            controlsMenu.gameObject.SetActive (false);
            pauseMenu.gameObject.SetActive (true);
        }
    }

    public void QueHacer (bool Open) {
        if (Open) {
            quehacerMenu.gameObject.SetActive (true);
            pauseMenu.gameObject.SetActive (false);
        }
        if (!Open) {
            quehacerMenu.gameObject.SetActive (false);
            pauseMenu.gameObject.SetActive (true);
        }
    }

    public void Pause () {
        if (canvas.gameObject.activeInHierarchy == false) {
            canvas.gameObject.SetActive (true);
            Time.timeScale = 0;
            Player.GetComponent<FirstPersonController> ().enabled = false;
            if (pauseMenu.gameObject.activeInHierarchy == false) {
                pauseMenu.gameObject.SetActive (true);
                controlsMenu.gameObject.SetActive (false);
                quehacerMenu.gameObject.SetActive (false);
                salirMenu.gameObject.SetActive (false);
            }
        } else {
            canvas.gameObject.SetActive (false);
            Time.timeScale = 1;
            Player.GetComponent<FirstPersonController> ().enabled = true;
            //Cursor.visible = false;
        }
    }

    public void Salir (bool Open) {
        if (Open) {
            salirMenu.gameObject.SetActive (true);
            pauseMenu.gameObject.SetActive (false);
        }
        if (!Open) {
            salirMenu.gameObject.SetActive (false);
            pauseMenu.gameObject.SetActive (true);
        }
    }

    public void Instrucciones () {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex -
1);
    }
}

```

Informacion Puertas. Activa y desactiva la información sobre las puertas interactivas:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class InformacionPuertas : MonoBehaviour {

    public GameObject informacionPuerta;

    // Use this for initialization
    void Start() {
        informacionPuerta.gameObject.SetActive(false);
    }

    private void OnTriggerEnter(Collider coll) {
        if (coll.gameObject.tag == "Player") {
            informacionPuerta.gameObject.SetActive(true);
        }
    }

    void OnTriggerExit(Collider coll) {
        if (coll.gameObject.tag == "Player") {
            informacionPuerta.gameObject.SetActive(false);
        }
    }
}
```

InteraccionTI. Recoge funciones generales de la escena de Teoría de la Imagen:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class InteraccionTI : MonoBehaviour {

    public GameObject LenteOscura;
    public GameObject LenteClara;
    public GameObject Lente;
    public GameObject ObjetivoOscurro;
    public GameObject ObjetivoClaro;
    public GameObject Objetivo;
    public GameObject CarcasaOscura;
    public GameObject CarcasaClara;
    public GameObject Carcasa;
    public GameObject TodosLosCanvas;
    public GameObject CamaraInicio;
    public GameObject CamaraHecha;
    public GameObject Reloj;
    public GameObject PlataformasContrarreloj;
    public GameObject InformacionCanvas;
```

```

public GameObject sonido;
public GameObject sonidoDestruirCamara;
public GameObject objetosInteractivos;
public GameObject objetivoInicio;
public GameObject objetivoFinal;

//variables relacionadas con el temporizador
public float tiempo = 0;
public float finalTiempo;
public bool inicioTiempo = false;
public GameObject sonidoTicTac;

//variables relacionadas con el temporizador y el canvas
public bool tiempoCanvas;
public float tiempoCanvasInicio = 0;
public float tiempoCanvasFinal;

//variables relacionadas con la salud
public bool canDestroy;
public Slider barraSalud;
public float salud;
public float damage;
public bool menosSalud;
public GameObject muerteObjeto;

// Use this for initialization
void Start () {
    Time.timeScale = 1;
    LenteOscura.gameObject.SetActive (true);
    LenteClara.gameObject.SetActive (false);
    Lente.gameObject.SetActive (true);
    ObjetivoOscurro.gameObject.SetActive (true);
    ObjetivoClaro.gameObject.SetActive (false);
    Objetivo.gameObject.SetActive (true);
    CarcasaOscura.gameObject.SetActive (true);
    CarcasaClara.gameObject.SetActive (false);
    Carcasa.gameObject.SetActive (true);
    TodosLosCanvas.gameObject.SetActive (false);
    CamaraInicio.gameObject.SetActive (true);
    CamaraHecha.gameObject.SetActive (false);
    Reloj.gameObject.SetActive (true);
    PlataformasContrarreloj.gameObject.SetActive (false);
    InformacionCanvas.gameObject.SetActive (false);
    tiempoCanvas = false;
    salud = 1;
    barraSalud.value = salud;
    menosSalud = true;
    objetosInteractivos.gameObject.SetActive (false);
    objetivoInicio.gameObject.SetActive (true);
    objetivoFinal.gameObject.SetActive (false);
}

// Update is called once per frame
void Update () {
    if (LenteOscura.gameObject == null && ObjetivoOscurro.gameObject == nu
11 && CarcasaOscura.gameObject == null) {
        CamaraHecha.gameObject.SetActive (true);
    }
    if (inicioTiempo == true) {
}

```

```

        finalTiempo = 15;
        tiempo = tiempo + Time.deltaTime;
        PlataformasContrarreloj.gameObject.SetActive (true);
        if (tiempo > finalTiempo) {
            PlataformasContrarreloj.gameObject.SetActive (false);
            tiempo = 0;
            inicioTiempo = false;
            Reloj.gameObject.SetActive (true);
            sonidoTicTac.GetComponent< AudioSource > ().Stop ();
        }
    }
    if (tiempoCanvas == true) {
        InformacionCanvas.gameObject.SetActive (true);
        tiempoCanvasFinal = 5;
        tiempoCanvasInicio = tiempoCanvasInicio + Time.deltaTime;
        if (tiempoCanvasInicio > tiempoCanvasFinal) {
            tiempoCanvas = false;
        }
    } else if (tiempoCanvas == false) {
        InformacionCanvas.gameObject.SetActive (false);
    }
    barraSalud.value = salud;
    if (barraSalud.value == 0) {
        Destroy (gameObject);
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
- 1);
    }
}

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Lente")) {
        Destroy (LenteOscura.gameObject);
        LenteClara.gameObject.SetActive (true);
        Lente.gameObject.SetActive (false);
        sonido.GetComponent< AudioSource > ().Play ();
    }
    if (col.CompareTag ("Objetivo")) {
        Destroy (ObjetivoOscuro.gameObject);
        ObjetivoClaro.gameObject.SetActive (true);
        Objetivo.gameObject.SetActive (false);
        sonido.GetComponent< AudioSource > ().Play ();
    }
    if (col.CompareTag ("Carcasa")) {
        Destroy (CarcasaOscura.gameObject);
        CarcasaClara.gameObject.SetActive (true);
        Carcasa.gameObject.SetActive (false);
        sonido.GetComponent< AudioSource > ().Play ();
    }
    if (col.CompareTag ("Reloj")) {
        inicioTiempo = true;
        Reloj.gameObject.SetActive (false);
        sonidoTicTac.GetComponent< AudioSource > ().Play ();
    }
    if (col.CompareTag ("CamaraHecha")) {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
    }
    if (col.CompareTag ("CamaraInicio")) {
        tiempoCanvas = true;
    }
}

```

```

        CamaraInicio.gameObject.SetActive (false);
        TodosLosCanvas.gameObject.SetActive (true);
        sonidoDestruirCamara.GetComponent< AudioSource > ().Play ();
        objetosInteractivos.gameObject.SetActive (true);
        objetivoInicio.gameObject.SetActive (false);
        objetivoFinal.gameObject.SetActive (true);
    }
}

void OnTriggerEnterStay (Collider col) {
    if (col.CompareTag ("muerte") && menosSalud) {
        salud = salud - damage * Time.deltaTime;
    }
}
}
}

```

Lagrimas Tiempo. Carga la siguiente escena tras diez segundos:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class LagrimasTiempo : MonoBehaviour {

    public float tiempo;
    public float tiempoInicio;
    public float tiempoFinal;

    // Use this for initialization
    void Start () {
        tiempoInicio = 0;
        tiempo = 1;
        tiempoFinal = 10;
    }

    // Update is called once per frame
    void Update () {
        tiempoInicio = tiempoInicio + tiempo * Time.deltaTime;
        if (tiempoInicio > tiempoFinal) {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
        }
    }
}

```

Mover. Permite el movimiento de los hazards del nivel de Teoría de la Publicidad y las Relaciones Públicas:

```

using UnityEngine;
using System.Collections;

public class Mover : MonoBehaviour {

```

```

private Rigidbody rb;
public float speed;

void Start () {
    rb = GetComponent<Rigidbody> ();
    rb.velocity = transform.forward * speed;
}
}

```

Main Menu. Recoge las funciones de la primera escena:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour {

    public void PlayGame () {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
    );
    }

    public void QuitGame () {
        Debug.Log ("QUIT!");
        Application.Quit ();
    }
}

```

Music Volume. Vincula el volumen de la música de fondo y de los efectos del player (caminar y saltar) con sliders del menú de pausa:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class MusicVolume : MonoBehaviour {

    public Slider musicVolume;
    public AudioSource myMusic;
    public Slider fxVolume;
    public AudioSource fxMusic;

    // Update is called once per frame
    void Update () {
        myMusic.volume = musicVolume.value;
        fxMusic.volume = fxVolume.value;
    }
}

```

Music Volume. Vincula el volumen de la música de fondo con un slider:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class MusicVolumeInicio : MonoBehaviour {

    public Slider musicVolume;
    public AudioSource myMusic;

    // Update is called once per frame
    void Update () {
        myMusic.volume = musicVolume.value;
    }
}

```

Music Volume TTLL. Debido a que en la última fase de “Tendencias Literarias en la Cultura Contemporánea” se reproducen dos canciones como música de fondo, hay un script propio que activa una y otra canción en función del libro que se esté leyendo:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class musicVolumeTTLL : MonoBehaviour {

    public Slider musicVolume;
    public Slider musicVolume2;
    public AudioSource myMusic;
    public AudioSource myMusic2;
    public bool hamelin;
    public bool enredo;
    public GameObject hamelinObjeto;
    public GameObject enredoObjeto;

    // Use this for initialization
    void Start () {
        hamelin = false;
        enredo = false;
        musicVolume.gameObject.SetActive (false);
        musicVolume2.gameObject.SetActive (false);
    }

    // Update is called once per frame
    void Update () {
        if (hamelinObjeto.gameObject.activeInHierarchy == true) {
            hamelin = true;
            musicVolume.gameObject.SetActive (true);
        }
        if (hamelinObjeto.gameObject.activeInHierarchy == false) {
            hamelin = false;
            musicVolume.gameObject.SetActive (false);
            musicVolume2.gameObject.SetActive (false);
        }
    }
}

```

```

        }
        if (enredoObjeto.gameObject.activeInHierarchy == true) {
            enredo = true;
            musicVolume2.gameObject.SetActive (true);
            musicVolume.gameObject.SetActive (false);
        }
        if (enredoObjeto.gameObject.activeInHierarchy == false) {
            enredo = false;
            musicVolume2.gameObject.SetActive (false);
        }
        if (enredo == true) {
            myMusic2.volume = musicVolume2.value;
        }
        if (hamelin == true) {
            myMusic.volume = musicVolume.value;
        }
    }
}

```

No Puedes Subir. Recoge las funciones de la planta dos (no poder subir a la planta tres y mostrar información sobre cómo acceder a las asignaturas):

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class NoPuedesSubir : MonoBehaviour {

    public GameObject avisoCanvas;
    public GameObject pizarraCanvas;
    public GameObject pizarraCanvas2;

    // Use this for initialization
    void Start () {
        avisoCanvas.gameObject.SetActive (false);
        pizarraCanvas.gameObject.SetActive (false);
        pizarraCanvas2.gameObject.SetActive (true);
    }

    // Update is called once per frame
    void Update () {
        if (pizarraCanvas.gameObject.activeInHierarchy == true && Input.GetKeyDown (KeyCode.E)) {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1);
        }
    }

    void OnTriggerEnter (Collider col) {
        if (col.CompareTag ("Aviso")) {
            avisoCanvas.gameObject.SetActive (true);
        }
        if (col.CompareTag ("Fire")) {
            pizarraCanvas.gameObject.SetActive (true);
            pizarraCanvas2.gameObject.SetActive (false);
        }
    }
}

```

```

        }

    }

    void OnTriggerExit (Collider col) {
        if (col.CompareTag ("Aviso")) {
            avisoCanvas.gameObject.SetActive (false);
        }
        if (col.CompareTag ("Fire")) {
            pizarraCanvas.gameObject.SetActive (false);
            pizarraCanvas2.gameObject.SetActive (true);
        }
    }
}

```

Open Drawer. Activa la animación de apertura y cierre de cajones en la escena de Historia de la Cultura Contemporánea:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class OpenDrawer : MonoBehaviour {

    public GameObject trigger;
    public GameObject drawer;
    public GameObject informacionCajon;

    Animator CajonAbre;

    // Use this for initialization
    void Start () {
        CajonAbre = drawer.GetComponent <Animator> ();
        informacionCajon.gameObject.SetActive (false);
    }

    void OnTriggerEnter (Collider coll) {
        if (coll.gameObject.tag == "Player") {
            informacionCajon.gameObject.SetActive (true);
        }
    }

    void OnTriggerStay (Collider coll) {
        if (coll.gameObject.tag == "Player" && Input.GetKeyDown (KeyCode.E))
    {
        OpenTheDrawer (true);
        informacionCajon.gameObject.SetActive (false);
    }
}

    void OnTriggerExit (Collider coll) {
        if (coll.gameObject.tag == "Player") {
            OpenTheDrawer (false);
            informacionCajon.gameObject.SetActive (false);
        }
    }
}

```

```

    void OpenTheDrawer (bool state) {
        CajonAbre.SetBool ("position", state);
    }

    // Update is called once per frame
    void Update () {

    }
}

```

Plataforma Ping Pong. Contiene las funciones del movimiento de plataformas, incluida la de vincular la posición del player:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlataformaPingPong : MonoBehaviour {

    private Transform plataforma;
    public GameObject Player;

    // Use this for initialization
    void Start () {
        plataforma = GetComponent<Transform> ();
        //plataforma.position = posicionOriginal;
    }

    // Update is called once per frame
    void Update () {

    }

    void OnTriggerEnter (Collider other) {
        if (other.gameObject == Player) {
            Player.transform.parent = transform;
        }
    }

    void OnTriggerExit (Collider other) {
        if (other.gameObject == Player) {
            Player.transform.parent = null;
        }
    }
}

```

Player Controller. Recoge funciones del player en las escenas de Teoría de la Publicidad y las Relaciones Públicas:

```

using UnityEngine;
using System.Collections;

```

```

[System.Serializable]
public class Boundary {

    public float xMin, xMax, zMin, zMax;

}

public class PlayerController : MonoBehaviour {

    private Rigidbody rb;
    public float speed;
    public float tilt;
    public Boundary boundary;

    public GameObject shot;
    public Transform shotSpawn;
    public float fireRate;
    private float nextFire;
    private AudioSource audio;

    void Start () {
        rb = GetComponent<Rigidbody> ();
        audio = GetComponent<

```

Player HCC. Contiene funciones de la escena de Historia de la Cultura

Contemporánea:

```

using System.Collections;
using System.Collections.Generic;

```

```

using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class PlayerHCC : MonoBehaviour {

    public GameObject llaveObjeto;
    public GameObject llaveCanvas;
    public GameObject llaveSonido;
    public Slider barraSalud;
    public float salud;
    public float damage;
    public GameObject sonidoPasarNivel;
    public GameObject puertaParejo;
    public bool llaveB;
    public GameObject llaveImaginaria;
    public GameObject puertaCerrada;
    public GameObject puertaAbierta;

    // Use this for initialization
    void Start () {
        llaveObjeto.gameObject.SetActive (true);
        llaveCanvas.gameObject.SetActive (false);
        salud = 1;
        barraSalud.value = salud;
        puertaParejo.gameObject.SetActive (false);
        llaveB = false;
        llaveImaginaria.gameObject.SetActive (false);
        puertaCerrada.gameObject.SetActive(true);
        puertaAbierta.gameObject.SetActive(false);

    }

    // Update is called once per frame
    void Update () {
        barraSalud.value = salud;
        if (barraSalud.value == 0) {
            Destroy (gameObject);
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
- 1);
        }
    }

    void OnTriggerEnter (Collider col) {
        if (col.CompareTag ("Llave")) {
            puertaCerrada.gameObject.SetActive(false);
            puertaAbierta.gameObject.SetActive(true);
            llaveSonido.GetComponent<AudioSource>().Play();
            llaveObjeto.gameObject.SetActive(false);
            llaveCanvas.gameObject.SetActive(true);
            llaveImaginaria.gameObject.SetActive(true);
        }
        if (col.CompareTag ("Cofre")) {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
        }
        if (col.CompareTag ("agua")) {
            sonidoPasarNivel.GetComponent<AudioSource> ().Play ();
        }
    }
}

```

```

        }

        void OnTriggerStay (Collider col) {
            if (col.CompareTag ("muerte")) {
                salud = salud - damage * Time.deltaTime;
            }
            if (col.CompareTag ("Door")) {
                puertaParejo.gameObject.SetActive (true);
            }
        }

        void OnTriggerExit (Collider col) {
            if (col.CompareTag ("Door")) {
                puertaParejo.gameObject.SetActive (false);
            }
        }
    }
}

```

Player Planta 4. Cargar la siguiente escena al llegar al despacho:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class PlayerPlanta4 : MonoBehaviour {

    void OnTriggerEnter (Collider col) {
        if (col.CompareTag ("Fire")) {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
        }
    }
}

```

Player TDIE. Recoge funciones generales de la escena de Teoría de la Información

Escrita:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityStandardAssets.Characters.FirstPerson;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PlayerTDIE : MonoBehaviour {

    public Transform canvasOrdenador;
    public Transform Player;
    public bool ordenadorB;

    //bolas Canvas ordenador
    public GameObject incompleta1;
    public GameObject completa1;
}

```

```

public GameObject incompleta2;
public GameObject completa2;
public GameObject incompleta3;
public GameObject completa3;

//libros
public GameObject Libro1;
public GameObject Libro2;
public GameObject Libro3;
public bool Libro1B;
public bool Libro2B;
public bool Libro3B;
public GameObject sonidoLibro;

//siguiente nivel
public bool pedirLibros;
public GameObject canvasPulsaE;

//variables relacionadas con la salud
public Slider barraSalud;
public float salud;
public float damage;

//variables relacionadas con los canvas de libros
public GameObject libro1Canvas;
public GameObject libro2Canvas;
public GameObject libro3Canvas;

void Start () {
    canvasOrdenador.gameObject.SetActive (false);
    ordenadorB = false;
    //bolas Canvas ordenador
    incompleta1.gameObject.SetActive (true);
    incompleta2.gameObject.SetActive (true);
    incompleta3.gameObject.SetActive (true);
    completa1.gameObject.SetActive (false);
    completa2.gameObject.SetActive (false);
    completa3.gameObject.SetActive (false);
    Libro1B = false;
    Libro2B = false;
    Libro3B = false;
    canvasPulsaE.gameObject.SetActive (false);
    barraSalud.value = salud;
    salud = 1;
    libro1Canvas.gameObject.SetActive (false);
    libro2Canvas.gameObject.SetActive (false);
    libro3Canvas.gameObject.SetActive (false);
}

void Update () {
    barraSalud.value = salud;
    if (Input.GetKeyDown (KeyCode.E) && ordenadorB) {
        Ordenador ();
    }
    if (Input.GetKeyDown (KeyCode.E) && Libro1B) {
        completa1.gameObject.SetActive (true);
        Destroy (incompleta1.gameObject);
        Destroy (Libro1.gameObject);
        sonidoLibro.GetComponent<

```

```

        Libro1B = false;
        libro1Canvas.gameObject.SetActive (true);
    }
if (Input.GetKeyDown (KeyCode.E) && Libro2B) {
    completa2.gameObject.SetActive (true);
    Destroy (incompleta2.gameObject);
    Destroy (Libro2.gameObject);
    sonidoLibro.GetComponent< AudioSource> ().Play ();
    Libro2B = false;
    libro2Canvas.gameObject.SetActive (true);
}
if (Input.GetKeyDown (KeyCode.E) && Libro3B) {
    completa3.gameObject.SetActive (true);
    Destroy (incompleta3.gameObject);
    Destroy (Libro3.gameObject);
    sonidoLibro.GetComponent< AudioSource> ().Play ();
    Libro3B = false;
    libro3Canvas.gameObject.SetActive (true);
}
if (incompleta1.gameObject == null && incompleta2.gameObject == null
&& incompleta3.gameObject == null && pedirLibros && Input.GetKeyDown (KeyCode
.E)) {
    SiguienteNivel ();
}
if (salud < 0) {
    Destroy (gameObject);
    SceneManager.LoadScene (SceneManager.GetActiveScene () .buildIndex
- 1);
}
}

public void Ordenador () {
    if (canvasOrdenador.gameObject.activeInHierarchy == false) {
        canvasOrdenador.gameObject.SetActive (true);
        Time.timeScale = 0;
        Player.GetComponent<FirstPersonController> ().enabled = false;
    } else {
        canvasOrdenador.gameObject.SetActive (false);
        Time.timeScale = 1;
        Player.GetComponent<FirstPersonController> ().enabled = true;
        //Cursor.visible = false;
    }
}

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Ordenador")) {
        ordenadorB = true;
    }
    if (col.CompareTag ("Libro1")) {
        Libro1B = true;
    }
    if (col.CompareTag ("Libro2")) {
        Libro2B = true;
    }
    if (col.CompareTag ("Libro3")) {
        Libro3B = true;
    }
    if (col.CompareTag ("Finish")) {
        SiguienteNivel ();
    }
}

```

```

        }
        if (col.CompareTag ("Fire") && incompleta1.gameObject == null && incompleta2.gameObject == null && incompleta3.gameObject == null) {
            pedirLibros = true;
            canvasPulsaE.gameObject.SetActive (true);
        }
    }

    void OnTriggerExit (Collider col) {
        if (col.CompareTag ("Ordenador")) {
            ordenadorB = false;
        }
        if (col.CompareTag ("Libro1")) {
            Libro1B = false;
        }
        if (col.CompareTag ("Libro2")) {
            Libro2B = false;
        }
        if (col.CompareTag ("Libro3")) {
            Libro3B = false;
        }
        if (col.CompareTag ("Fire") && incompleta1.gameObject == null && incompleta2.gameObject == null && incompleta3.gameObject == null) {
            pedirLibros = false;
            canvasPulsaE.gameObject.SetActive (false);
        }
    }

    void OnTriggerStay (Collider col) {
        if (col.CompareTag ("muerte")) {
            salud = salud - damage * Time.deltaTime;
        }
    }

    public void SiguienteNivel () {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1);
    }
}

```

Puerta Parejo. Contiene funciones de la escena de Historia de la Cultura

Contemporánea:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PuertaParejo : MonoBehaviour {

    public GameObject trigger;
    public GameObject doorR;
    public GameObject llave;
    public GameObject puertaSonido;
    public GameObject llaveImaginaria;
    public GameObject llaveCanvas;

```

```

    Animator rotation;

    public bool llaveCogida;

    // Use this for initialization
    void Start () {
        rotation = doorR.GetComponent <Animator> ();
        llaveCogida = false;
        llave.gameObject.SetActive (true);
        llave.gameObject.SetActive (false);
        llaveImaginaria.gameObject.SetActive (false);
    }

    void Update () {
        if (llave.gameObject.activeInHierarchy == true) {
            llaveCogida = true;
        }
    }

    void OnTriggerEnter (Collider coll) {
        if (coll.gameObject.tag == "Player" && llaveImaginaria == true) {
            if (Input.GetKeyDown (KeyCode.E)) {
                RotateDoors (true);
                puertaSonido.GetComponent<

```

Puntuacion. Recoge la información sobre la puntuación del examen final:

```

using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;

public class Puntuacion : MonoBehaviour {

    public Text puntosText;
    public int puntos;
    public float acierto = 0;
    public GameObject pizarra1;
    public GameObject pizarra4;
    public GameObject siguienteAprobado;
    public GameObject siguienteSuspensos;

```

```

// Use this for initialization
void Start () {
    pizarra1.gameObject.SetActive (true);
    UpdateScore ();
    puntos = 0;
    puntosText.text = "0";
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
    siguienteAprobado.gameObject.SetActive (false);
    siguienteSuspensivo.gameObject.SetActive (false);
    pizarra4.gameObject.SetActive (false);
}

// Update is called once per frame
void Update () {
    if (puntosText.text == "10" || puntosText.text == "9" || puntosText.text == "8" || puntosText.text == "7" && pizarra4.gameObject.activeInHierarchy == true) {
        siguienteAprobado.gameObject.SetActive (true);
        siguienteSuspensivo.gameObject.SetActive (false);
    }
    if (puntosText.text == "0" || puntosText.text == "1" || puntosText.text == "2" || puntosText.text == "3" && pizarra4.gameObject.activeInHierarchy == true) {
        siguienteSuspensivo.gameObject.SetActive (true);
        siguienteAprobado.gameObject.SetActive (false);
    }
    if (puntosText.text == "4" || puntosText.text == "5" || puntosText.text == "6" && pizarra4.gameObject.activeInHierarchy == true) {
        siguienteSuspensivo.gameObject.SetActive (true);
        siguienteAprobado.gameObject.SetActive (false);
    }
}
public void AddScore (int newScoreValue) {
    puntos += newScoreValue;
    UpdateScore ();
}

public void UpdateScore () {
    puntosText.text = "" + puntos;
}

public void SalirDelJuego () {
    Application.Quit ();
    Debug.Log ("QUIT!");
}

public void Victoria () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1);
}

public void Examen () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 0);
}

```

```

        public void Asignaturas () {
            SceneManager.LoadScene ("_TransicionTIE");
        }
    }
}

```

Puntuacion Texto. Funciones del texto de la puntuación del examen final:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class puntuacionTexto : MonoBehaviour {

    private RectTransform rt;
    public GameObject pizarra4;
    public RectTransform nuevo;
    public Vector3 posicionFinal;
    //public Vector3 escalaFinal;
    public Text texto;

    // Use this for initialization
    void Start () {
        rt = GetComponent<RectTransform> ();
        texto = GetComponent<Text> ();
    }

    // Update is called once per frame
    void Update () {
        if (pizarra4.gameObject.activeInHierarchy == true) {
            rt.localPosition = posicionFinal;
            texto.fontSize = 50;
        }
    }
}

```

Random Rotator. Genera el movimiento aleatorio de los hazards en las escenas de Teoría de la Publicidad y las Relaciones Públicas:

```

using UnityEngine;
using System.Collections;

public class RandomRotator : MonoBehaviour {

    public float tumble;
    private Rigidbody rb;

    void Start () {
        rb = GetComponent<Rigidbody> ();
        rb.angularVelocity = Random.insideUnitSphere * tumble;
    }
}

```

Rotation_Doors. Permite el movimiento de apertura (con ayuda de la tecla E) y clausura (de manera automática) de las puertas:

```
using UnityEngine;

public class Rotation_Doors : MonoBehaviour {

    public GameObject trigger;
    public GameObject doorR;
    public GameObject informacionCajon;
    public GameObject puertaSonido;

    Animator rotation;

    // Use this for initialization
    void Start () {
        rotation = doorR.GetComponent <Animator> ();
        informacionCajon.gameObject.SetActive (false);
    }

    private void OnTriggerEnter(Collider coll) {
        if (coll.gameObject.tag == "Player") {
            informacionCajon.gameObject.SetActive (true);
        }
    }

    void OnTriggerStay (Collider coll) {
        if (coll.gameObject.tag == "Player" && Input.GetKeyDown (KeyCode.E)) {
            RotateDoors (true);
            informacionCajon.gameObject.SetActive (false);
            puertaSonido.GetComponent< AudioSource > ().Play ();
        }
    }

    void OnTriggerExit(Collider coll) {
        if (coll.gameObject.tag == "Player") {
            RotateDoors (false);
            informacionCajon.gameObject.SetActive (false);
        }
    }

    void RotateDoors (bool state) {
        rotation.SetBool ("rotate", state);
    }

    // Update is called once per frame
    void Update () {
    }
}
```

Salud Enemigo. Recoge funciones relacionadas con la salud del enemigo final de la escena de Teoría de la Publicidad y las Relaciones Públicas:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class SaludEnemigo : MonoBehaviour {

    public Slider barraSalud;
    public float salud;
    public float damage;
    public GameObject victoria;
    public GameObject boss;
    public GameObject bossSonido;
    public GameObject explosionFinal;

    // Use this for initialization
    void Start () {
        salud = 1;
        barraSalud.value = salud;
        victoria.gameObject.SetActive (false);
    }

    // Update is called once per frame
    void Update () {
        barraSalud.value = salud;
        if (salud < 0) {
            Instantiate (explosionFinal, boss.transform.position, boss.transform.rotation);
            bossSonido.GetComponent<AudioSource> ().Play ();
            Destroy (gameObject);
        }
    }

    void OnTriggerEnter (Collider col) {
        if (col.CompareTag ("Disparo")) {
            salud = salud - damage;
        }
    }

    //void Destruir () {
    //if (salud == 0) {
    //tiempoMuerte = true;
    //}
    //}
}

}

```

Screen Roll. Permite hacer bajar los créditos y activar el objeto “fin”.

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class ScreenRoll : MonoBehaviour {

    public RectTransform rt;

```

```

public Vector3 posicionFinal;
public Vector3 posicionInicio;
public bool bajar;
public GameObject fin;

// Use this for initialization
void Start () {
    rt = GetComponent<RectTransform> ();
    bajar = true;
    rt.transform.localPosition = posicionInicio;
    fin.gameObject.SetActive (false);
}

// Update is called once per frame
void Update () {
    if (bajar == true) {
        rt.transform.localPosition += Vector3.down;
        if (rt.localPosition == posicionFinal) {
            bajar = false;
            fin.gameObject.SetActive (true);
        }
    }
}
}

```

screen Roll 2. Permite hacer bajar la palabra “fin” y finalizar el juego automáticamente tras cinco segundos.

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class screenRoll2 : MonoBehaviour {

    public RectTransform rt;
    public Vector3 posicionFinal;
    public Vector3 posicionInicio;
    public bool bajar;

    //variables relacionadas con el tiempo
    public float tiempo;
    public float tiempoFinal;
    public float masTiempo;
    public bool tiempoCorre;

    // Use this for initialization
    void Start () {
        rt = GetComponent<RectTransform> ();
        bajar = true;
        tiempo = 0;
        tiempoFinal = 5;
        masTiempo = 1;
        tiempoCorre = false;
        rt.transform.localPosition = posicionInicio;
    }
}

```

```

// Update is called once per frame
void Update () {
    if (bajar == true) {
        rt.transform.localPosition += Vector3.down;
        if (rt.localPosition == posicionFinal) {
            bajar = false;
            tiempoCorre = true;
        }
    }
    if (tiempoCorre == true) {
        tiempo = tiempo + masTiempo * Time.deltaTime;
        if (tiempo > tiempoFinal) {
            Debug.Log ("FINAL");
            Application.Quit ();
            tiempoCorre = false;
        }
    }
}
}

```

Sliding_Doors_Script. Permite el movimiento de apertura y clausura de las puertas (automáticas y ascensor):

```

using UnityEngine;

public class Sliding_Doors_Script : MonoBehaviour {

    public GameObject trigger;
    public GameObject leftDoor;
    public GameObject rightDoor;
    public GameObject sonidoPuerta;

    Animator leftAnim;
    Animator rightAnim;

    // Use this for initialization
    void Start () {
        leftAnim = leftDoor.GetComponent <Animator> ();
        rightAnim = rightDoor.GetComponent <Animator> ();

    }

    //Estos métodos funcionan para las puertas automáticas

    void OnTriggerEnter(Collider coll) {
        if (coll.gameObject.tag == "Player") {
            SlideDoors(true);
            sonidoPuerta.GetComponent< AudioSource >().Play();
        }
    }

    void OnTriggerExit (Collider coll) {
        if (coll.gameObject.tag == "Player")
            SlideDoors (false);
    }
}

```

```

void SlideDoors (bool state) {
    leftAnim.SetBool ("slide", state);
    rightAnim.SetBool ("slide", state);
}

//Este método funciona para abrir las puertas cuando se quiera

// Update is called once per frame
void Update () {
    //if (Input.GetKeyDown (KeyCode.E)) {
        //SlideDoors (true);
    //}
    //if (Input.GetKeyDown (KeyCode.R)) {
        //SlideDoors (false);
    //}
}
//Este método funciona para abrir y cerrar las puertas cuando se está cerca
a

//void OnTriggerStay (Collider coll) {
    //if (coll.gameObject.tag == "Player") {
        //if (Input.GetKeyDown (KeyCode.E)) {
            //SlideDoors (true);
        //}
    //}
    //if (coll.gameObject.tag == "Player") {
        //if (Input.GetKeyDown (KeyCode.R)) {
            //SlideDoors (false);
        //}
    //}
}
//}
}

```

Sumar Enredo. Actualiza la puntuación de objetos interactivos encontrados en el libro El Enredo del último nivel de Tendencias Literarias en la Cultura Contemporánea:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class sumarEnredo : MonoBehaviour {

    public Text puntuacion5;
    private float puntuacion;
    private float puntuacion0;
    private float puntuacion1;
    private float puntuacion2;
    private float puntuacion3;
    private float puntuacion4;
    public bool añadirPuntuacion;
    public GameObject uno;
    public GameObject dos;
    public GameObject tres;
}

```

```

public GameObject cuatro;
public GameObject cinco;
public GameObject sonido;
public GameObject sonido2;
public GameObject puertas;
public GameObject canvasVictoria;

public float tiempo;
public float tiempoFinal;
public float masTiempo;
public bool tiempoCanvas;

// Use this for initialization
void Start () {
    puntuacion = 0;
    puntuacion5.text = puntuacion + "/5";
    añadirPuntuacion = false;
    canvasVictoria.gameObject.SetActive (false);
    tiempoCanvas = false;
    masTiempo = 1;
    tiempoFinal = 5;
}

// Update is called once per frame
void Update () {
    puntuacion5.text = puntuacion + puntuacion0 + puntuacion1 + puntuacion
2 + puntuacion3 + puntuacion4 + "/5";
    if (puntuacion5.text == "5/5") {
        Destroy (puertas.gameObject);
        sonido2.GetComponent< AudioSource > ().Play ();
        tiempoCanvas = true;
    }
    if (tiempoCanvas == true) {
        tiempo = tiempo + masTiempo * Time.deltaTime;
        canvasVictoria.gameObject.SetActive (true);
        if (tiempo > tiempoFinal) {
            tiempoCanvas = false;
            Destroy (canvasVictoria.gameObject);
            tiempo = 0;
        }
    }
}

void OnTriggerEnter (Collider col) {
    if (col.CompareTag ("Uno")) {
        añadirPuntuacion = true;
        Destroy (uno.gameObject);
        puntuacion0 = 1;
        sonido.GetComponent< AudioSource > ().Play ();
    }
    if (col.CompareTag ("Dos")) {
        añadirPuntuacion = true;
        Destroy (dos.gameObject);
        puntuacion1 = 1;
        sonido.GetComponent< AudioSource > ().Play ();
    }
    if (col.CompareTag ("Tres")) {
        añadirPuntuacion = true;
        Destroy (tres.gameObject);
    }
}

```

```

        puntuacion2 = 1;
        sonido.GetComponent<AudioSource> ().Play ();
    }
    if (col.CompareTag ("Cuatro")) {
        añadirPuntuacion = true;
        Destroy (cuatro.gameObject);
        puntuacion3 = 1;
        sonido.GetComponent<AudioSource> ().Play ();
    }
    if (col.CompareTag ("Cinco")) {
        añadirPuntuacion = true;
        Destroy (cinco.gameObject);
        puntuacion4 = 1;
        sonido.GetComponent<AudioSource> ().Play ();
    }
}
}
}

```

Transicion Asignaturas. Carga el siguiente nivel (pero para poder ejecutarse, es necesaria la función **On Click ()** de un botón):

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class TransicionAsignaturas : MonoBehaviour {

    void Start () {
        Cursor.visible = true;
    }

    public void PlayGame () {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 1
    );
    }
}

```

Trigger Parejo. Contiene funciones de la escena de Historia de la Cultura Contemporánea:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TriggerParejo : MonoBehaviour {

    public GameObject trigger;
    public GameObject drawer;
    public GameObject informacionCajon;
    public GameObject sonidoCajon;
    public GameObject pared1;
}

```

```

public GameObject pared2;
public GameObject luzPared;
public bool paredes;

Animator CajonAbre;

// Use this for initialization
void Start () {
    CajonAbre = drawer.GetComponent <Animator> ();
    informacionCajon.gameObject.SetActive (false);
    pared1.gameObject.SetActive (true);
    pared2.gameObject.SetActive (false);
    luzPared.gameObject.SetActive (false);
}

void OnTriggerEnter (Collider coll) {
    if (coll.gameObject.tag == "Player") {
        informacionCajon.gameObject.SetActive (true);
    }
}

void OnTriggerStay (Collider coll) {
    if (coll.gameObject.tag == "Player" && Input.GetKeyDown (KeyCode.E))
    {
        OpenTheDrawer (true);
        informacionCajon.gameObject.SetActive (false);
        paredes = true;
        sonidoCajon.GetComponent< AudioSource > ().Play ();
    }
}

void OnTriggerExit (Collider coll) {
    if (coll.gameObject.tag == "Player") {
        OpenTheDrawer (false);
        informacionCajon.gameObject.SetActive (false);
    }
}

void OpenTheDrawer (bool state) {
    CajonAbre.SetBool ("position", state);
}

// Update is called once per frame
void Update () {
    if (paredes == true) {
        pared1.gameObject.SetActive (false);
        pared2.gameObject.SetActive (true);
        luzPared.gameObject.SetActive (true);
    }
}
}

```

TTLL Enredo. Recoge las funciones generales del último nivel de Tendencias

Literarias en la Cultura Contemporánea:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class TTLEnredo : MonoBehaviour {

    public GameObject waterSplash;
    private Vector3 inicio = Vector3.zero;
    private CharacterController move;
    public GameObject Enredo;
    public GameObject Hamelin;
    public GameObject Gira;
    public GameObject malvado;
    public GameObject Explosion;
    public GameObject ObjetoExplosion;
    public GameObject ObjetoExplosionEnredo;
    public GameObject ObjetoExplosionHamelin;
    public GameObject ParedHamelin;
    public GameObject ParedEnredo;
    public GameObject canvasEnredo;
    public GameObject sonido2;
    public GameObject libroEnredo;
    public GameObject libroHamelin;
    public GameObject canvasInicio;

    public bool EnredoB;
    public bool HamelinB;
    public bool EnredoBien;

    //variables relacionadas con la salud
    public Slider barraSalud;
    public float salud;
    public float damageHamelin;
    public float damageEnredo;

    // Use this for initialization
    void Start () {
        move = GetComponent<CharacterController> ();
        Enredo.SetActive (false);
        Hamelin.SetActive (false);
        Gira.SetActive (false);
        malvado.gameObject.SetActive (false);
        ParedEnredo.gameObject.SetActive(true);
        ParedHamelin.gameObject.SetActive(true);
        EnredoB = false;
        HamelinB = false;
        salud = 1;
        barraSalud.value = salud;
        canvasEnredo.gameObject.SetActive (false);
        EnredoBien = false;
        libroEnredo.gameObject.SetActive (false);
        libroHamelin.gameObject.SetActive (true);
        canvasInicio.gameObject.SetActive (true);
    }

    // Update is called once per frame
    void Update () {

```

```

        if (Input.GetKeyDown (KeyCode.E) && EnredoB == true && EnredoBien ==
true) {
            Enredo.SetActive (true);
            ParedEnredo.gameObject.SetActive (false);
            Instantiate (Explosion, ObjetoExplosionEnredo.transform.position,
ObjetoExplosionEnredo.transform.rotation);
            EnredoB = false;
            canvasEnredo.gameObject.SetActive (true);
            Hamelin.GetComponent< AudioSource > ().Stop ();
        }
        if (Input.GetKeyDown (KeyCode.E) && HamelinB == true) {
            Hamelin.SetActive (true);
            ParedHamelin.gameObject.SetActive (false);
            Instantiate (Explosion, ObjetoExplosionHamelin.transform.position
, ObjetoExplosionHamelin.transform.rotation);
            HamelinB = false;
            canvasInicio.gameObject.SetActive (false);
            //sonidoBackground.GetComponent< AudioSource > ().Stop ();
        }
        if (salud < 0) {
            Muerte ();
        }
        barraSalud.value = salud;
    }

    void OnTriggerEnter (Collider Col) {
        if (Col.CompareTag ("Enredo")) {
            EnredoB = true;
        }
        if (Col.CompareTag ("Hamelin")) {
            HamelinB = true;
        }
        if (Col.CompareTag ("agua")) {
            waterSplash.GetComponent< AudioSource > ().Play ();
            move.transform.position = inicio;
        }
        if (Col.CompareTag ("MesaExamen")) {
            move.transform.position = inicio;
            EnredoBien = true;
            salud = 1;
            sonido2.GetComponent< AudioSource > ().Play ();
            libroEnredo.gameObject.SetActive (true);
            libroHamelin.gameObject.SetActive (false);
            ParedHamelin.gameObject.SetActive (true);
        }
        if (Col.CompareTag ("VGira")) {
            Gira.SetActive (true);
            Instantiate (Explosion, ObjetoExplosion.transform.position, Obj
oExplosion.transform.rotation);
            malvado.gameObject.SetActive (true);
        }
        if (Col.CompareTag ("Finish")) {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
        }
    }

    void OnTriggerExit (Collider Col) {
        if (Col.CompareTag ("Enredo")) {

```

```

        EnredoB = false;
    }
    if (Col.CompareTag ("Hamelin")) {
        HamelinB = false;
    }
}

void OnTriggerStay (Collider col) {
    if (col.CompareTag ("Llamas")) {
        salud = salud - damageHamelin * Time.deltaTime;
    }
    if (col.CompareTag ("Escaleras")) {
        salud = salud - damageEnredo * Time.deltaTime;
    }
}

public void Muerte () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 0
);
}
}
}

```

TTLL Lagrimas. Recoge las funciones generales del segundo nivel de Tendencias Literarias en la Cultura Contemporánea:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class TTLLagrimas : MonoBehaviour {

    public GameObject waterSplash;
    private Vector3 inicio = Vector3.zero;
    private CharacterController move;
    public GameObject Lagrimas;
    public GameObject Explosion;
    public GameObject ObjetoExplosionLagrimas;
    public GameObject ParedLagrimas;
    public GameObject sonidoBackground;

    public bool LagrimasB;

    //variables relacionadas con la salud
    public Slider barraSalud;
    public float salud;
    public float damageFuego;
    public float damageGolpe;

    // Use this for initialization
    void Start () {
        move = GetComponent<CharacterController> ();
        Lagrimas.SetActive (false);
        ParedLagrimas.gameObject.SetActive(true);
        LagrimasB = false;
        salud = 1;
    }
}

```

```

        barraSalud.value = salud;
    }

// Update is called once per frame
void Update () {
    if (Input.GetKeyDown (KeyCode.E) && LagrimasB == true) {
        Lagrimas.SetActive (true);
        ParedLagrimas.gameObject.SetActive(false);
        Instantiate (Explosion, ObjetoExplosionLagrimas.transform.position,
                    ObjetoExplosionLagrimas.transform.rotation);
        LagrimasB = false;
        sonidoBackground.GetComponent<if (salud < 0) {
        Muerte ();
    }
    barraSalud.value = salud;
}

void OnTriggerEnter (Collider Col) {
    if (Col.CompareTag ("Lagrimas")) {
        LagrimasB = true;
    }
    if (Col.CompareTag ("agua")) {
        waterSplash.GetComponent< AudioSource > () .Play ();
        move.transform.position = inicio;
    }
    if (Col.CompareTag ("MesaExamen")) {
        SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
    }
}

void OnTriggerExit (Collider Col) {
    if (Col.CompareTag ("Lagrimas")) {
        LagrimasB = false;
    }
}

void OnTriggerStay (Collider col) {
    if (col.CompareTag ("Llamas")) {
        salud = salud - damageFuego;
    }
    if (col.CompareTag ("Golpe")) {
        salud = salud - damageGolpe;
    }
}

public void Muerte () {
    SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex + 0
);
}
}

```

TTLL Retablo. Recoge las funciones generales del primer nivel de Tendencias

Literarias en la Cultura Contemporánea:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class TTLLRetablo : MonoBehaviour {

    public GameObject waterSplash;
    private Vector3 inicio = Vector3.zero;
    private CharacterController move;
    public GameObject Retablo;
    public GameObject Explosion;
    public GameObject ObjetoExplosionRetablo;
    public GameObject ParedRetablo;
    public GameObject sonidoBackground;

    public bool RetabloB;

    //variables relacionadas con la salud
    public Slider barraSalud;
    public float salud;

    // Use this for initialization
    void Start () {
        move = GetComponent<CharacterController> ();
        Retablo.SetActive (false);
        ParedRetablo.gameObject.SetActive(true);
        RetabloB = false;
        salud = 1;
        barraSalud.value = salud;
    }

    // Update is called once per frame
    void Update () {
        if (Input.GetKeyDown (KeyCode.E) && RetabloB == true) {
            Retablo.SetActive (true);
            ParedRetablo.gameObject.SetActive(false);
            Instantiate (Explosion, ObjetoExplosionRetablo.transform.position
, ObjetoExplosionRetablo.transform.rotation);
            RetabloB = false;
        }
        if (salud < 0) {
            Muerte ();
        }
        barraSalud.value = salud;
    }

    void OnTriggerEnter (Collider Col) {
        if (Col.CompareTag ("Retablo")) {
            RetabloB = true;
        }
        if (Col.CompareTag ("agua")) {
            waterSplash.GetComponent< AudioSource > ().Play ();
            move.transform.position = inicio;
        }
        if (Col.CompareTag ("MesaExamen")) {
            move.transform.position = inicio;
            //SceneManager.LoadScene ("_ExamenTTLL");
        }
    }
}

```

```
        }
        if (Col.CompareTag ("Escaleras")) {
            SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex
+ 1);
        }
    }

    void OnTriggerExit (Collider Col) {
        if (Col.CompareTag ("Retablo")) {
            RetabloB = false;
        }
    }

    public void Muerte () {
        salud = 1;
        move.transform.position = inicio;
    }
}
```