Universitat Oberta de Catalunya

uoc.edu

Estudio y desarrollo de un serious game

Joaquín Campos Colmenarejo
Máster Universitario en Diseño y Programación de Videojuegos
Trabajo Final de Máster (TFM)

Helio Tejedor Navarro
Joan Arnedo Moreno

16 de Junio, 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Estudio y desarrollo de un serious game*
Nombre del autor:	*Joaquín Campos Colmenarejo*
Nombre del consultor/a:	*Helio Tejedor Navarro*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*06/2020*
Titulación o programa:	*Máster Universitario en Diseño y Programación de Videojuegos*
Área del Trabajo Final:	*Trabajo Final de Máster (TFM)*
Idioma del trabajo:	*Castellano*
Palabras clave	*Serious games, Game design, Custom game engine*

Resumen del Trabajo (máximo 250 palabras):

El Trabajo presenta, como primera parte, un breve estudio del videojuego The McDonald's Videogame, un serious game reivindicativo que fue todo un éxito a mediados de la primera década de este siglo, con la intención de extraer mecánicas que me permitiese elaborar un serious game de temática propia. En la segunda parte, el lector encontrará la conceptualización de dicho juego de temática propia, el desarrollo de un motor de videojuegos propio y una demo técnica.

El videojuego conceptualizado en el presente Trabajo es de corte reivindicativo y su narrativa orbita alrededor de la situación de las universidades en plena crisis financiera. El jugador es persuadido a tomar decisiones moralmente incorrectas con tal de obtener la victoria, siendo esta dinámica un pilar central del diseño del videojuego, dinámica que comparte con The McDonald's Videogame.

Abstract (in English, 250 words or less):

This work presents, in its first part, a brief study on the videogame The McDonald's Videogame, a reivindicative serious game that was a huge success in the middle of the first decade of this century, with the aim of drawing the mechanics that would let me to build a serious game with own theme. In its second part, the reader will find the concept of the game, the development of a self-made videogames engine and a technical demo.

The game conceptualized in the present work is reivindicative and its narrative moves around the situation of the universities in times of a financial crisis. The player is persuaded to take morally incorrect decisions in order to get the victory. This dynamic is a central pillar of this game's design and it is shared with The McDonald's Videogame.

Índice

Lista de figuras

Tablas

Ilustraciones

1. Introducción

1.1 Contexto y justificación del Trabajo

El presente Trabajo se enmarca dentro de la asignatura del Trabajo Final de Máster (TFM) del Máster Universitario en Diseño y Programación de Videojuegos, cuya finalidad es la síntesis de los conocimientos adquiridos en todo el programa educativo y que cubre todas las competencias correspondientes.

Con respecto a la temática del Trabajo, el estudio y desarrollo de un juego político, nace de mi interés en los *serious games* y en la capacidad que tiene el medio de los videojuegos como transmisor de conocimiento, de diseñador a jugador, de productor a consumidor, más allá del entretenimiento que el videojuego debe ofrecer de forma inherente. Así pues, el Trabajo conforma mi plataforma para el estudio de un videojuego, a propósito, político, The McDonald's Videogame, para posteriormente proceder al desarrollo de un videojuego político de temática propia.

El resultado que pretendo obtener de este Trabajo, entonces, no es más que la adquisición y práctica de mecánicas que favorezcan la transmisión de conocimiento, que tomarán vida en dicho videojuego propagandístico de temática propia.

1.2 Objetivos del Trabajo

- Estudiar y poner en práctica mecánicas de *feedback negativo* para crear dinámicas persuasivas.
- Desarrollo de un motor propio de videojuegos web que garantice la accesibilidad al producto del Trabajo.

1.3 Enfoque y método

Las posibles estrategias eran principalmente dos: desarrollar el videojuego con un motor de videojuegos completo, que ofreciese una vía rápida para el prototipado de mecánicas; o desarrollar el videojuego con un motor de desarrollo propio, asegurando la accesibilidad del producto del Trabajo pero arriesgando la propia viabilidad del proyecto.

La estrategia escogida sería la segunda, pero sería escogida forzosamente en un estadio avanzado del proyecto, haciendo inevitable la marcha atrás. Luego, la estrategia escogida no fue la más apropiada y la planificación siguió su curso en la medida de lo posible.

Esta estrategia se desarrollaría bajo una metodología ágil, con una división del tiempo de desarrollo por sprints, de manera que los riesgos se pudiesen resolver de forma rápida y sencilla. No obstante, y cómo se observa a continuación, la metodología no se pudo aplicar siquiera debido a los problemas surgidos.

1.4 Planificación del Trabajo

La planificación se planteó inicialmente en 4 fases diferenciadas de desarrollo: 2 fases de producción, 1 fase de test o QA y 1 última fase de ajuste. En total hubiesen resultado 14 sprints de 1 semana cada uno.

Las dos fases de producción habían sido separadas de este modo de manera que al finalizar la primera fase de producción, el producto estuviese en su fase MVP (Minimum Viable Product), es decir, el juego en su expresión más básica. Luego, se podían iniciar sesiones de playtest con ese producto MVP para comenzar a probar las diferentes funcionalidades del juego y poder corregir a tiempo, en concordancia

En las siguientes tablas (Tabla 1 y 2) se observan las diferentes tareas y los tiempos en los que se planificó su desarrollo:

Tabla 1. Primer fragmento de la planificación.

		PROD I					PROD II					QA		TWEAK	
		02-Mar	09-Mar	16-Mar	23-Mar	30-Mar	06-Apr	13-Apr	20-Apr	27-Apr	04-May	11-May	18-May	25-May	01-Jun
Bloque	**Tareas**					ENTREGA							ENTREGA		ENTREGA
Menús	Splash Screen														
	Menú principal														
	Créditos														
	Configuración														
	Pausa														
	Game Over														
Lógica	Economía del juego														
	Sistema de tiempo														
	Sistema de eventos														
	Sistema de tilemap														
	Pathfinding (?)														
	Animaciones														
	Sistema de compra/venta														
	Empleados (habilidades)														
	Cursos (gestión)														
	NPCs														
	Niveles (procedural?)														
	Alumnos (ánimo y opinión)														
	Tutorial (in-game)														

Tabla 2. Segundo fragmento de la planificación.

		PROD I					PROD II					QA		TWEAK	
		02-Mar	09-Mar	16-Mar	23-Mar	30-Mar	06-Apr	13-Apr	20-Apr	27-Apr	04-May	11-May	18-May	25-May	01-Jun
Bloque	**Tareas**					ENTREGA							ENTREGA		ENTREGA
UI	Caja de diálogo														
	Sistema de ventanas														
	Tablas y gráficos														
	Listas														
	HUD														
	Fuentes														
	Opciones para discapacitados														
	Ayuda														
Finetuning	Excel to game (?)														
	Playtests						*								
	Finetuning														
	Bugfixing														
Sonido	Música														
	SFX														
Publicación	Licencia														
	Manual de usuario														
	Preparar Releases plataformas														
	Trailer														
	Marketing (otros)														
Narrativa	Desarrollar historia principal														
	Desarrollar eventos														
	Desarrollar opiniones alumnos														
Sistema trad.	Sistema trad.														

La planificación sería más adelante un fracaso por la sorpresa de un motor con falta de funciones y la subestimación de la carga de trabajo que tendría el desarrollo de un motor propio.

Así pues, los primeros cambios que se darían en la planificación presentada en las tablas anteriores serían la eliminación de las dos últimas fases, pues no eran estrictamente requeridas para la evaluación del Programa de estudio. Luego, se descartarían tareas de la misma planificación, como todas aquellas agrupadas en “Finetuning”, “Publicación” o “Sistema de traducciones”. Por último, se descartarían características del videojuego que ya no serían posibles de realizar, como la representación gráfica del nivel, las opciones para cubrir discapacidades o diferentes mecánicas que se pretendían implementar.

1.5 Breve sumario de productos obtenidos

El resultado del Trabajo es una demo técnica del motor desarrollado, disponible desde cualquier navegador web compatible, con ciertas funciones del videojuego implementadas, pero no siendo éste un videojuego que ofrezca una experiencia completa.

1.6 Breve descripción del resto de capítulos de la memória

A continuación enumero los diferentes capítulos que conforman este documento y los describo brevemente:

- Estado del arte: un estudio superficial sobre el presente estado del arte en el contexto de interés. Análisis del caso de estudio The McDonald’s Game.
- Definición del videojuego: desarrollo de la idea del videojuego de creación propia.
- Diseño técnico: descripción del motor de desarrollo escogido y arquitectura del videojuego desarrollado.
- Conclusiones: descripción de las conclusiones del Trabajo.

2. Estado del arte

2.1 Los serious games y el género de la simulación

Los serious games son juegos diseñados con una intención más allá que la de entretener. Entre todos ellos, hay un subgénero de videojuegos con una intención claramente política y que pretenden persuadir al jugador con una temática ideológica o un programa político concretos.

El género de la simulación en los serious games es bastante frecuente, dado que permite al jugador representar una posición de actor último de todo lo que ocurre en el mundo del juego, muchas veces con capacidades divinas. Es, a su vez, el género que ofrece una visión más detallada del contexto del juego, y es por esto que se suelen ver tantos títulos que pertenecen al grupo de los serious games y como género principal tienen la simulación.

Títulos ejemplo de este hecho son los clásicos SimCity, Roller Coaster Tycoon, o juegos más novedosos como Game Dev Tycoon. Éstos no solo entretienen al jugador con un conjunto de mecánicas que permiten la aparición de dinámicas emergentes, también tienen un propósito educativo en cuanto a que muestran, de forma caricaturizada, los procesos de desarrollo de un ámbito en concreto, como la construcción de una ciudad o la evolución de un parque de atracciones.

2.2 Caso de estudio: The McDonald’s Videogame

The McDonald’s Videogame es un videojuego desarrollado por Molleindustria en 2005 que tuvo un gran éxito en los diferentes portales de videojuegos flash de la época. La sinopsis del juego, en la propia página web del mismo, dice así:

"Making money in a corporation like McDonald's is not easy! Behind every burger there is a complex process you need to master: from pastures to the slaughterhouse, from restaurants to branding. You'll discover all the dirty secrets that made us one of the biggest company in the world."

The McDonald's Videogame es un serious game del género de la simulación y tiene una evidente intención persuasiva y presenta la cruda realidad de las cadenas de comida rápida como McDonald's, desde el punto de vista de un gestor de la compañía. Como gestor de la compañía tienes el objetivo de hacer crecer el negocio mediante el correcto control de la larga cadena de producción, que comienza en Brasil con las plantaciones de soja y las granjas de vacuno, pasando por la producción y venta del producto final y termina en las oficinas de McDonald's, dónde se toman las decisiones ejecutivas y se construye la campaña de mercadotecnia.

El juego es aderezado con un tono sarcástico que visibiliza los problemas de la cadena de producción, o "dirty secrets" como aparece en la sinopsis, de calado capitalista e imperialista. Así pues, por ejemplo, en el eslavón de la cadena dedicada a la plantación de soja, el jugador se enfrenta a la decisión de arrasar el cultivo para la alimentación de los nativos, así como la destrucción de selva para aumentar la producción. En el eslavón de la producción cárnica, el jugador tiene que añadir componentes de dudosa calidad en la comida del vacuno, así como hormonas para aumentar la velocidad de crecimiento, y sacrificar a las vacas que enferman debido a las condiciones higiénicas y alimenticias. En el resto de eslavones el jugador se ve en la tesitura de castigar a empleados y despedirlos como simple mercancia, o sobornar políticos para favorecer el branding de la empresa.

Todas estas acciones, que son opcionales en el juego, son forzadas mediante un *feedback negativo*, que obliga al jugador, mediante una reducción drástica de las ganancias y, por lo tanto, de una aproximación a la derrota en el juego, a

tener que tomar esas acciones, que son dañinas para los colectivos en cuestión. De esta manera, el juego pretende hacer ver al jugador que el crecimiento del capital requiere de la explotación de todo aquello que entre en contacto con su cadena de producción y que las grandes corporaciones como McDonald's tienen una naturaleza destructiva.

Ilustración 1. Captura de pantalla del videojuego The McDonald's Game

Debido a la naturaleza interactiva del videojuego, el mensaje tiene una mayor probabilidad de ser asumido por el jugador, pues es quién está tomando acción y observa las consecuencias en tiempo real, en una escala reducida y fácil de comprender.

Por lo tanto, la aplicación del *feedback negativo* como mecánica de juego es el aspecto en el que se centra este Trabajo y sobre el que se proyecta el videojuego de temática propia que se definirá en los siguientes apartados.

Otro aspecto que se puede extraer de The McDonald's Game, que marcó su éxito y que se plantea para el Trabajo, es la distribución libre y gratuita en la web. Si bien, el contexto es distinto y mediados de los 2000 era un caldo de cultivo para el desarrollo de serious games de corte político, ahora el panorama

es distinto y la plataforma más parecida a la habida entonces son mercados como itch.io.

3. Definición del videojuego

3.1 Concepto

El concepto del videojuego parte de la misma idea que The McDonald's Videogame. El jugador toma el rol de rector de la universidad, pero éste lo hace en plena crisis financiera, caracterizada por un déficit económico grave y una campaña de recortes en los departamentos de educación. El videojuego, entonces, pone al jugador en la tesitura de tener que ajustar los gastos de la universidad, como una empresa más, frente a unos stakeholders que exigen unos beneficios y una comunidad educativa que exige una calidad en los trabajos y en los servicios ofrecidos.

El juego tiene una intención principalmente educativa, mostrando la cara más cruda de la educación en los países desarrollados, resultado de un empobrecimiento del sistema y la aplicación de reformas que no han sido satisfactorias. También, la decadencia de la escuela pública, debido a la burocratización de la misma y la aplicación de regulaciones educativas como el Plan Bolonia.

Tomando el ejemplo de The McDonald's Videogame, como muchos otros, el juego será del género de la simulación, poniendo al jugador al cargo de todas las funciones. El juego estará repleto de sarcasmo y situaciones exageradas, pues no se pretende desarrollar un juego hiperrealista, sino más bien un juego reivindicativo.

3.2 Gameplay y mecánicas

El gameplay se caracterizaría por un uso mayoritario de la interfaz gráfica, mediante ventanas y botones. Conceptualmente, el juego se presenta al jugador como el campus virtual de la universidad. Por ejemplo, la pantalla del título del juego será un login a dicho campus virtual, mientras que el resto el juego sería una interpretación de un simple escritorio de ordenador. Ésto, como podrá ver en la demo técnica desarrollada, no ha sido viable de realizar en el

periodo disponible, por ello quiero que quede constancia en el presente documento.

Los objetivos que el jugador tendría que asumir, serían los siguientes:

1. Conseguir que la universidad obtenga ganancias. Al final de año, la universidad debe tener un balance positivo. Este objetivo es una referencia al corporativismo y la privatización (aún subjetiva) de las universidades.
2. Mantener un buen ratio de alumnos y hacer de la universidad una institución reconocida.
3. Obtener buenas opiniones de los estudiantes.

El juego permitiría alcanzar estos objetivos mediante acciones que consideraríamos buenas y otras que consideraríamos malas. Pero debido a la intención persuasiva del juego, las acciones malas tendrán más relevancia para la consecución de estos objetivos. Este aspecto tiene una razón de ser, que es la demostración de la decadencia de los sistemas educativos, pues implica que el jugador reconoce que está tomando malas decisiones para ganar la partida de una forma sencilla. El jugador más avanzado, que es sin duda el que más nos interesa, reconocería la capacidad que tiene para alcanzar dichos objetivos tomando acciones buenas, con lo cual el aprendizaje sería mucho más valioso.

A propósito, para enumerar algunas de las acciones, cuando hablamos de acciones buenas hablamos de ofrecer buenas condiciones al trabajador de la universidad, escuchar a los estudiantes para ofrecerles el mejor servicio, construir una comunidad educativa fuerte y de valor, etcétera; mientras que las acciones malas serían, por ejemplo, la división cada vez mayor de los roles que participan en la educación para abaratar los costes, aumentar la oferta de programas sin desarrollarlos cualitativamente al nivel que se espera de una institución universitaria o utilizar mecanismos políticos para obtener una mayor presencia mediática.

En cuanto a las mecánicas, podemos distinguir las siguientes:

- Fondos de la universidad: los fondos de la universidad es lo que define la vida en el mundo de juego. Si los fondos son negativos y finaliza el periodo fiscal, la universidad destituye al rector y, por lo tanto, la partida se termina; si los fondos son positivos, recompensas equivalentes a las ganancias obtenidas son otorgadas al jugador. Una de las características del juego será la escasez de fondos, justificado por unos stakeholders que no ven mucha viabilidad en el proyecto.
- Límite temporal: el juego transcurre en periodos mensuales en tiempo ficticio, es decir, que unos segundo del tiempo real suponen un mes del tiempo de juego. Cuando transcurre un mes, ciertos gastos son aplicados a los fondos de la universidad, implicando presión y estrés al jugador. Los ingresos a los fondos se dan en determinados meses, por ejemplo, de forma semestral en las matrículas de los alumnos, aunque pueden ser acelerados obteniendo mejoras en departamentos de la universidad como, por ejemplo, capacidad de financiar el alumnado y recibir pagos mensuales. Además, se establece un periodo anual, en tiempo de juego, que determina la continuidad o no de la partida.
- Mejora de capacidades: la universidad se compone de diferentes departamentos y éstos pueden recibir mejoras para beneficiar al jugador. En este sentido, las mejoras no son necesariamente acciones buenas en el contexto del mundo de juego, como ya he indicado previamente. Un ejemplo sería la investigación en nuevos roles del profesorado, para permitir a la universidad contratar a trabajadores con peores condiciones para cometer el rol del profesor, pero cobrando mucho menos. No hace falta decir que estas mejoras tendrían un coste en fondos y que, por lo tanto, tienen un riesgo.
- Contratación y despedida de profesores: es la mecánica central del videojuego, pues conceptualmente los profesores representan el verdadero valor de la calidad de una universidad. El jugador debe contratar a los profesores que más le convengan, teniendo en cuenta su coste y sus habilidades. Estas contrataciones se materializarán en la bolsa de profesores, con perfiles que aparecerán temporalmente.

También deberá despedir si pretende aliviar en finanzas. No obstante, estas acciones tendrán un impacto en las valoraciones de los cursos y, por lo tanto, en el número de alumnos y el ingreso en los fondos de la universidad. También tendrán un impacto en la comunidad sindicalista de trabajadores de educación.

- Valoraciones de los alumnos: las valoraciones de los alumnos se podrían entender como la puntuación del juego y dependerán de, como he mencionado, la calidad del profesorado y de los programas de estudios, y tendrán un efecto en la cantidad de nuevas matrículas, la captación de profesorado de nivel y más.
- Planes de marketing: el jugador deberá invertir en planes de marketing para aumentar la capacidad de la universidad para captar alumnos. Por supuesto, estos planes de marketing podrán contener publicidad engañosa, siguiendo con la temática sarcástica del videojuego.
- Narrativa guiada por eventos: el juego en su transcurso ejecutará diferentes eventos que tendrán un impacto relevante en las decisiones del jugador. Estos eventos pueden ser, por ejemplo, publicaciones negativas en la prensa que requerirían que el jugador efectuase nuevos planes de marketing.

Todas estas mecánicas promueven dinámicas circulares que obligan al jugador a tomar acciones, que tienen como efecto una reducción de ganancias que obligan a su vez volver a tomar acciones, retroalimentándose constantemente. Estas dinámicas acabarían conformando el feedback negativo que mencionaba anteriormente, el cual acabaría apoyando las acciones negativas del usuario.

El juego presenta, por lo tanto, varios niveles de dificultad, que dependen del tipo de acciones que toma el jugador. Sin duda, las acciones malas a las que es empujado el jugador, no sólo mediante las mecánicas, sino también mediante la narrativa, serían el modo más fácil de avanzar en el juego, aunque el resultado final de la partida ofrecería una puntuación pobre. En cambio, si el jugador se dispone a realizar acciones buenas, el juego será muchísimo más difícil, pero el resultado final será mucho más positivo para el usuario. Este

aspecto se puede observar en juegos como This War of Mine. Además, tiene un efecto en la rejugabilidad del videojuego.

3.3 Historia y personajes

La historia es la ya descrita y es muy simple: eres el rector de la escuela en tiempos de crisis y debes mantenerla en buen estado financiero. La narrativa, enfocada en el componente persuasivo y educativo del videojuego, tiene más interés en la elaboración de una narrativa emergente, nacida de diferentes eventos que se dan de forma aleatoria a lo largo de la partida, que en el establecimiento de una narrativa lineal.

Más importante que la historia son los personajes implicados, que son 3 principales:

- El rector: el jugador, personaje principal del videojuego. Carece de personalidad y tiene un aspecto neutro. Se presenta como una figura masculina para representar la dificultad de las mujeres para acceder a ciertas posiciones, por la existencia de los techos de cristal.
- Los stakeholders: son los diferentes actores que tienen poder directo sobre los fondos de la universidad. Exigen ganancias de la universidad y pueden tomar decisiones arbitrarias sobre el funcionamiento de la institución sin el consentimiento del jugador.
- El sindicato: representa al conjunto de trabajadores de la universidad y tiene un gran peso con respecto a la imagen de la universidad y, por lo tanto, la capacidad de atracción de estudiantes que ésta tiene. Es antagonista de los stakeholders.

Si bien estos son los personajes más importantes, hay que tener en cuenta que hay muchos otros eventos que podrían representar gobiernos, medios informativos, consejos de estudiantes, etcétera.

Otro elemento característico del videojuego serían los comentarios de los estudiantes, algo así como en Roller Coaster Tycoon, que mostraba las opiniones de los visitantes del parque de atracciones, bastante innovador en su

momento con respecto al resto de juegos de simulación, y que supondría una fuente de información bastante amplia para la toma de decisiones del jugador. Estos comentarios vendrían dados cada semestre tal y como ocurre en las universidades reales.

Por último, insistir en que la apariencia del juego, así como el mundo de juego construido alrededor de la universidad, pretenden ser lo más cercanos a la realidad, de manera que el jugador pudiese hacer una intepretación realista del mundo de juego. No obstante, y para potenciar esa capacidad persuasiva, la narrativa estaría exagerada, pero sin romper con este marco realista.

4. Diseño técnico

4.1 Elección de motor

Para la elección del motor se tuvieron en cuenta principalmente las opciones de accesibilidad que ofrecería el producto resultante del trabajo en el motor en cuestión. Para que fuese lo más accesible posible, decidí que el juego debía operar en navegadores web y que su exportación a tiendas de aplicaciones móviles fuese posible. Así mismo, el videojuego debía ser ligero, para que conexiones de internet modestas pudiesen descargarlo con cierta soltura, y debía funcionar en dispositivos de capacidades ajustadas.

Los 4 motores que se disputaron la posición fueron: Unity, Three.js, Babylon.js y PlayCanvas. A priori, el más adecuado para desarrollar el videojuego fue Babylon.js; posteriormente encontraría problemas que no supe avanzar y que afectaron al normal desarrollo del videojuego.

La principal razón por la que escogería Babylon.js fue, como ya he mencionado arriba, la accesibilidad del producto resultante, y es que Babylon.js es un motor web, es decir, el producto podría ejecutarse sin dificultades en la mayoría de navegadores web, aunque también podría ser distribuido en entornos móviles gracias a su extensión Babylon.js Native, una muy prometedora característica del motor que permite la distribución en prácticamente la mayoría de plataformas con la elaboración de un sólo código fuente. Por otro lado, el desarrollo open-source del motor y una gran comunidad detrás del mismo eran una garantía en cuanto a la longevidad del proyecto, que no dependería de que la empresa privada de turno mantuviese el software actualizado una vez pasados unos cuantos años, reduciendo así, también, el coste de mantenimiento.

El otro motor que más números tenía era Unity, porque ya tenía experiencia en la herramienta, pues el Programa de estudio se centraba en él. No obstante, no cumplía con el requisito de la accesibilidad pues, si bien Unity puede compilar para varias plataformas, la compilación para los navegadores web no es

eficiente, requiriendo un programa intermedio, el Unity Web Player. Unity está trabajando en este aspecto, desarrollando el Project Tiny, que permitirá la exportación de un videojuego en formato HTML5 con un resultado optimizado, pero el desarrollo de este componente está en una fase muy temprana y quedaba descartado para el Trabajo.

Luego, el principal problema que encontré en Babylon.js fue la naturaleza genérica del mismo. El motor está diseñado principalmente para servir las necesidades gráficas del software que se va a desarrollar y, en este sentido, no ofrece componentes que se dan por hecho en otros motores y que sirven para alojar la lógica del programa, las mecánicas del juego y más. Además, desconocía Babylon.js y su API, de modo que la mayor parte del Trabajo se dedica a estudiar el motor y desarrollar esos componentes asociados a la lógica del videojuego, es decir, al desarrollo del motor propio.

4.2 Software adicional y configuración

El desarrollo se ha llevado a cabo con la ayuda de unas pocas herramientas adicionales y una configuración específica.

Como herramientas adicionales destacaré Sublime Text 3, que ha sido usado como editor de texto, y Git, para el control de versiones del código.

La configuración del proyecto sigue unas especificaciones concretas debido, principalmente, al uso del último estándar de Javascript, ES6, para poder hacer uso del sistema de paquetes que este estándar ofrece. Dado que los navegadores no soportan completamente esta función del lenguaje, se requiere de una “compilación” que traduzca el código fuente a un código que el navegador sepa interpretar. Para esta compilación se ha hecho uso de Webpack. Adicionalmente, Webpack también preparará los assets del proyecto para que estos puedan ser accedidos posteriormente, y empaquetará el proyecto para que se concentre en un sólo fichero, reduciendo así el número de descargas que tendrá que efectuar el navegador del usuario.

4.3 Arquitectura del motor

El motor depende de Babylon.js como motor de renderizado y también de sus estructuras de datos más básicas. En este sentido, el motor desarrollado tiene una plena dependencia con Babylon.js, lo cual impediría, en cierta manera, un uso más generalista del motor. Más adelante se describirán dichas dependencias.

El motor está conformado por varios módulos, uno principal o “core” y otros auxiliares. Éstos ofrecen unas clases con las que elaborar nuestro videojuego.

4.3.1 El módulo principal

El módulo principal está diseñado con la intención de ser el núcleo de cualquier juego que sea desarrollado con el motor y contiene las estructuras de datos y los algoritmos básicos para el funcionamiento de cualquier videojuego. El módulo ofrece las siguientes clases:

- System: es el núcleo del juego y funciona como un “hub” que permite acceder al resto de componentes necesarios para la ejecución del juego, los que se enumeran posteriormente. Es el encargado de iniciar el juego.
- Time: es el componente encargado de medir el tiempo de ejecución del juego, así como de ofrecer el tiempo delta al resto de componentes para su actualización.
- GameScene: el componente que define la escena del juego. Contiene el mundo de la escena, los objetos de ese mundo, la lógica que aplica a ese mundo, etcétera. Tiene una dependencia con la estructura Scene de Babylon.js, que tiene la misma función y que es requerida por el motor para la actualización y renderizado de los elementos del juego. Luego, para evitar esta dependencia y, por lo tanto, tener una GameScene más genérica, que posteriormente pudiese funcionar con, por ejemplo, un motor de renderizado 2D, se requeriría de la elaboración de un puente, una interfaz, para instanciar una Scene de Babylon.js a partir de la GameScene del motor propio.

- SceneManager: un gestor de escenas, encargado de preparar, iniciar, actualizar y terminar las escenas del juego. El gestor es multiescena, es decir, puede ejecutar más de una escena a la vez. Esta característica es útil para poder tener escenas con claras diferencias de comportamiento: una para la representación del mundo de juego o nivel, otra para el HUD, otra para los elementos interactivos de la interfaz gráfica, etcétera. Por otro lado, el SceneManager gestiona la carga (y descarga) de assets antes de iniciar las escenas.
- Loader: el loader es un gestor de carga y descarga de assets. Depende, de nuevo, del AssetsManager de Babylon.js, el cual requiere para proceder con la descarga y preparación de assets para su posterior uso en el juego. El componente también tiene la función de ser un servidor centralizado de assets, de modo que los assets sean entregados a petición de las diferentes escenas de un modo sencillo. Si bien el componente permite una descarga asíncrona de assets, sólo se ha implementado para servir de forma síncrona, y se comunica con el SceneManager para confirmar que la descarga y procesado se ha realizado con éxito, de manera que éste sea capaz de iniciar las escenas con seguridad.
- GameEvent: es una clase abstracta que sirve como estructura para definir un evento de juego, que no deja de ser un mensaje que se transporta de un componente a otro para tener un efecto concreto en el juego. Por ejemplo, tras pulsar un botón en el HUD, se debe sustraer un valor de una variable de otro componente del cual no se tiene scope; en cuyo caso, se utilizaría un evento.
- EventBus: para enviar el evento descrito anteriormente, se requiere de un bus de eventos. En este bus de eventos, los diferentes componentes del juego se suscriben a un evento en concreto y cuando el evento es lanzado, todos aquellos eventos suscritos serán capaces de escuchar el evento y de procesarlo. Es de esta manera que el SceneManager es capaz de escuchar cuándo los assets ya han sido descargados: éste está suscrito al evento del Loader que marca cuándo está todo listo y, cuando el propio Loader lanza el evento, el SceneManager continúa con la ejecución de las escenas. No tiene ningún nivel de seguridad

implementado, de modo que cualquier componente podría lanzar el evento del Loader sin autorización y el SceneManager interpretaría el mensaje como si hubiese lanzado por el Loader; sería una de las mejoras que se podrían realizar en el sistema.

4.3.2 Landmark System

El Landmark System cubre las necesidad del juego de una narrativa guiada por eventos. Describiré la función de este módulo mediante sus 3 componentes diferenciados: el landmark, la condición y la acción.

El landmark es el evento en la narrativa. Representa un momento en el espacio y tiempo del mundo de juego, como puede ser el inicio del juego o, en el caso del videojuego desarrollado, el momento en que la cuenta bancaria está en números rojos. Las condiciones son usadas para analizar el estado del juego y evaluar si se ha alcanzado el landmark descrito, mientras que las acciones son los cambios que se ejecutarán sobre el estado del juego si se ha alcanzado dicho landmark.

Luego, el LandmarkSystem es un componente que centraliza la revisión de las condiciones de los diferentes landmark y la consiguiente ejecución de las acciones, si las condiciones requeridas se cumplen. Éste se inicializa con la infromación de los diferentes eventos o landmarks, sus condiciones y acciones, que pueden ser cargados de un fichero externo. Entonces, el sistema utiliza las diferentes estructuras de datos para gestionar el flujo mencionado.

Es importante mencionar que las condiciones y las acciones son definidas por identificadores de texto. Luego, la implementación es inyectada mediante scripts, favoreciendo así que el diseñador del juego pueda trabajar sin tener que enfrentarse a código más complejo. He aquí un ejemplo de la instanciación de un landmark:

```
{
	"id": "LANDMARK_OUT_OF_CASH",
	"conditions": [
		["CONDITION_VARIABLE_LESS_THAN", "VARIABLE_CASH", 0]
	],
	"actions": [
		["ACTION_SHOW_MESSAGE", "TEXT_OUT_OF_CASH"],
		"ACTION_UNSUBSCRIBE_LANDMARK"
	]
}
```

Como se puede observar, el landmark es identificado con el texto “LANDMARK_OUT_OF_CASH”. Las condiciones que se tienen que comprobar son “CONDITION_VARIABLE_LESS_THAN”, y la variable a comprobar en cuestión es “VARIABLE_CASH” y el valor es 0. Es decir, si la variable “cash” es menor que cero, entonces se habrá alcanzado el landmark y las acciones marcadas se realizarán. En este caso son dos acciones: “ACTION_SHOW_MESSAGE”, que muestra un mensaje en pantalla con el texto “TEXT_OUT_OF_CASH”, que se obtiene de un repositorio de textos, preparando así el juego para un sistema de traducciones; y “ACTION_UNSUBSCRIBE_LANDMARK” que desuscribe el landmark del sistema, haciéndolo de única iteración.

Las condiciones y acciones se implementan en sus respectivas factorías, que se inyectan en el LandmarkSystem de manera que haya desacoplamiento de las clases, pues el LandmarkSystem no necesita saber cuáles son las condiciones y las acciones que pertenecen al juego en concreto, pues su naturaleza es genérica. Luego, el LandmarkSystem acude a estas factorías para obtener la condición o la acción determinada por sus identificadores. Para la traducción de identificadores de variables a las propias variables del juego, se implementa un ParamsParser, que es básicamente un diccionario de identificadores que devuelve la correspondiente variable en el contexto del mundo de juego.

El sistema cuenta además con una optimización que recuerda, en cierta manera, a los sistemas ECS con un diseño orientado a datos, en cuanto a que reconoce la evidente repetición de tareas y, por lo tanto, realiza la ejecución por condiciones y acciones, y no landmarks. Es decir, como los diferentes landmarks harán uso de condiciones y acciones que tendrán uso compartido, el sistema ejecuta los diferentes métodos agrupados por condiciones y acciones. Esto es, en cuanto a optimización de memoria, para quizás 10 landmark, que pueden ejecutarse en un orden aleatorio, el código para la evaluación de la condición que comparten sólo es cargado en memoria una vez, ejecutado las 10 veces, una por landmark y, finalmente, extraido de memoria. Si esta optimización no se diese, entonces ocurriría que por cada landmark, se tendría que cargar en memoria dicha condición, ejecutarla y extraerla de memoria, ocupando más ciclos de reloj en inserción y extracción que en propia ejecución.

4.3.3 Módulo de la GUI

Como último módulo genérico, el módulo de la GUI establece las clases que son reutilizables en cuanto a la interfaz gráfica. Este módulo es dependiente del paquete GUI de Babylon.js, que establece un sistema para el renderizado de elementos gráficos mediante el uso de un canvas HTML de dos dimensiones. Del mismo modo que con otras dependencias con Babylon.js, lo ideal sería desacoplar los componentes mediante interfaces, para poder usar este módulo con otros motores de renderización. No obstante, en este caso, no es tan necesario el desacoplamiento, dada la muy probable dificultad de encontrar APIs tan similares.

En este módulo se establecen dos componentes:

- Transition: es una clase que permite la representación de una transición de fundido a negro entre escenas.
- Window: es una parte esencial del videojuego desarrollado, pues es el componente que habilita la interacción del usuario con el mundo de juego. El componente define una ventana y su comportamiento, a la que se le puede incorporar contenido.

Estas ventanas son posteriormente administradas por la escena correspondiente, permitiendo la apertura y eliminación de ventanas, ordenándolas gráficamente e incluso permitiendo el movimiento alrededor de la escena.

En cuanto al movimiento de la ventana por la escena, el recurso utiliza es interesante. Se instancian modelos 3D, esferas en el caso de este videojuego, que no son visibles por el jugador, pero que se utilizan como referencia para la posición de la ventana y que son las que realmente se arrastran por pantalla. En este sentido, las esferas sólo se mueven por un plano, de modo que no haya translaciones extrañas de las ventanas. Esta lógica se debe a que no hay otro método óptimo para mover las ventanas pues, imagino, tampoco era la intención de los diseñadores de la herramienta de tener interfaces que se pudieran arrastrar.

4.4 Implementación del juego

El juego se implementa mediante los componentes desarrollados en el módulo Game, y que responden al contexto específico del videojuego, y con las escenas en el módulo Scenes, que contiene los elementos gráficos y el comportamiento de los mismos. La idea detrás de esta implementación es la de establecer un diseño con componentes de modelo y vista claramente diferenciados, puesto que el videojuego dependerá mayormente de los componentes de la interfaz y es viable desarrollar un motor con un diseño de este tipo. De este modo, el modelo se actualiza independientemente de la vista y la vista, que son los componentes de la interfaz, es la encargada de recoger el valor y actualizarse ella misma.

El contexto del juego está representado por diferentes clases que son instanciadas por el GameState. El GameState es el núcleo que centraliza el estado del mundo de juego y que se encarga de actualziarlo. Así, el GameState isntancia los modelos:

- University: el modelo que define la universidad en su conjunto. La universidad, a su vez, tiene instancias de los diferentes profesores, bajo el modelo Teacher, y además tiene instanciados los valores de la cantidad de fondos que tiene la universidad, el coste de la matrícula y el número de estudiantes.
- Time: este modelo es diferente al del core del motor y define el tiempo en el mundo de juego. Este tiempo se caracteriza por representar los diferentes meses, que transcurren cada ciertos segundos. El componente define Enero del 2009 como año de inicio, pues es el año en que empezarían los recortes de presupuesto en educación y es una parte fundamental de la narrativa y dinamizadora de las mecánicas de *feedback negativo*.

Este GameState es inicializado al inicio del juego e inyectado a las diferentes escenas de juego para que puedan acceder al contexto y actualizarse correspondientemente.

Las vistas se construyen en las diferentes escenas. Una de las características que se han implementado en el motor de creación propia es, precisamente, un gestor de escenas (SceneManager) con propiedades multiescena y esto permitirá que el juego se componga de varias escenas que se ejecutan simultaneamente. Veanse las diferentes escenas que compone el videojuego:

- SplashScreenScene: esta escena se ejecuta al comienzo del videojuego y tiene la función de mostrar las entidades u organizaciones colaboradoras en el desarrollo, como bien puede ser el logotipo de la empresa desarrolladora o, en el caso de este Trabajo, el logotipo de la Universitat Oberta de Catalunya.
- GameplayScene: esta escena es la que contiene la lógica principal del videojuego, así como la representación más diegética del mundo de juego. También es la encarga de mandar a actualizar el modelo del juego, así como la de gestionar la descarga de assets que requerirán esta escena y las que se ejecuten simultáneamente. Este último aspecto es debido a la falta de una lógica que gestione de forma apropiada la

descarga de assets de diferentes escenas y, técnicamente, la descarga de assets debería ser deber exclusivo de cada escena, en función de los assets que requiriese la misma.

- HUDScene: esta escena contiene los elementos que conforman el HUD del juego, que son el controlador de tiempo, el scoreboard y el menú del sistema o de pausa.
- GUIScene: esta escena contiene, de momento, el sistema de ventanas del juego, esenciales para la interacción del jugador. Es decir, la escena es capaz de crear y eliminar ventanas, ordenarlas (z index) y moverlas por pantalla. Este sistema podría muy posiblemente separarse en una clase independiente.

Las escenas se comunican entre sí con los diferentes eventos que se definen en el módulo Events, que contiene los eventos GameplayEvent y GUIEvent. De este modo, si se abre el menú del sistema en el HUDScene, el GameplayScene recibe un evento para pausar el juego.

5. Conclusiones

Las conclusiones que se pueden extraer de este trabajo son varias y dispares, y las describo a continuación.

Por un lado, el Programa de estudio ofrecía un tiempo ajustado para realizar el Trabajo, de un semestre. En esta situación, la planificación es de un aspecto crítico para el correcto desarrollo del proyecto. Este tiempo ajustado apremia a aquellos que son capaces de virar rápidamente en cuanto encuentran un inconveniente y, en este sentido, el proyecto ha podido llegar a su fin, con un resultado modesto, lo cual, a mi juicio, ya es *de facto* una demostración de las capacidades que se exigen al finalizar el Programa.

Por otro lado, con respecto al producto del Trabajo, no esperaba trabajar tanto en la parte más técnica del proyecto, dada que mi especialidad en el Programa está más dirigida al diseño de videojuegos y menos a motores. La elección de motor fue un error: el objetivo del trabajo era el prototipado de un videojuego con unas mecánicas específicas pero, no obstante, puse en relieve aspectos más técnicos, de distribución y acceso al videojuego. Por suerte o por desgracia, el Trabajo me ha permitido desarrollar esa parte que no pude desarrollar en su momento, luego adquirir conocimientos adicionales que fortalecerán mi carrera profesional.

Si bien, aún dadas las difíciles circunstancias, se ha podido practicar, de una forma muy simple, con mecánicas basadas en el *feedback negativo*. No obstante, me hubiese gustado poder desarrollar más características del videojuego, que permitiesen una mayor interacción, una mayor aparición de dinámicas y, en definitiva, una experiencia de juego mucho más completa. Luego, este objetivo no se ha podido cumplir completamente y quedará como una tarea pendiente en el futuro.

Y es que, observando el panorama sobre el que he tenido que desarrollar el Trabajo, me sorprende gratamente saber que ha podido llegar a buen puerto,

con un trabajo a jornada completa, una pandemia global acechando y una repatriación forzosa de 3000 km por carretera. ¡Este Trabajo ha recorrido Europa!

6. Bibliografía

1. Información sobre serious games en Wikipedia (16 de Junio, 2020)
   https://en.wikipedia.org/wiki/Serious_game

2. Web de The McDonald's Videogame (16 de Junio, 2020)
   https://www.molleindustria.org/mcdonalds/