Universitat Oberta de Catalunya
uoc.edu

CarOut!

Víctor Manuel Revaliente Casilla
Máster en Diseño y Desarrollo de Videojuegos
Trabajo Final de Máster

Joan Arnedo Moreno
Jordi Duch Gavaldà

06/06/21

FICHA DEL TRABAJO FINAL

Título del trabajo:	*CarOut!*
Nombre del autor:	*Víctor Manuel Revaliente Casilla*
Nombre del consultor/a:	*Jordi Duch Gavaldà*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Master de Diseño y Programación de Videojuegos*
Área del Trabajo Final:	*Trabajo Final de Máster*
Idioma del trabajo:	*Castellano*
Palabras clave	*Videojuego, arcade, 3D*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Este trabajo tiene como objetivo el desarrollo desde 0 de un videojuego. Incluye tanto la fase de diseño como la de implementación y pruebas. El resultado es CarOut!, un videojuego en el que controlamos un coche dentro de un escenario delimitado y en el que, mediante choques contra otros vehículos y el uso de trampas, ataques especiales o ítems, deberemos de expulsar del escenario al resto de contrincantes de modo que seamos los últimos en quedar sobre el ring.

El juego es de estilo arcade, busca la diversión directa y rápida, enfocada especialmente al multijugador local de hasta 4 jugadores.
Durante su desarrollo se ha intentado seguir la filosofía del “Indie solo developer”, en el que una sola persona cubre el diseño, desarrollo y creación del videojuego completo, abarcando tanto las artes como el sonido, la música, programación, QA, animaciones… Aunque por la limitación temporal se ha tenido que recurrir a assets externos, hemos trabajado material propio para todas las áreas.

El resultado final es un videojuego completo con las mecánicas definidas, apto para añadir fácilmente nuevos escenarios y modos de juegos que amplíen la experiencia, creen nuevas dinámicas y aumenten la duración del juego.

Abstract (in English, 250 words or less):

This work aims to develop a video game from scratch. It includes both the design phase and the implementation and testing phase. The result is CarOut!, a video game which we control a car into a defined scenario and with mechanics as collisions with other vehicles and the use of traps, special attacks or items, we will have to expel the rest of the opponents from the scene. So that we are the last to be in the ring.

The game is arcade style, looking for direct and fast fun, especially focused on local multiplayer with up to 4 players.

During its development it has tried to follow the philosophy of the "Indie solo developer", in which a single person covers the design, development and creation of the complete video game, covering both the arts, sound, music, programming, QA, animations ... Although due to the time limitation we have had to resort to external Assets, we have worked our own material for all areas.

The final product is a complete video game with defined mechanics, suitable for easily adding new scenarios and game modes that extend the experience, create new dynamics and extend the duration of the game.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El mundo de los videojuegos se encuentra en constante evolución. Resulta difícil concretar cual fue el primer videojuego de la historia. A nivel conceptual, se puede considerar al “Dispositivo de entretenimiento con tubo de rayos catódicos” creado por Thomas T. Goldsmith Jr y Estle Ray Mann como el primer dispositivo en ejecutar un juego electrónico interactivo **[1]**. La máquina se publicó a finales de 1948 y emulaba los lanzamientos de misiles tal como lo hacían los radares que se empleaban durante la Segunda Guerra Mundial.

No fue hasta 1972 y con el lanzamiento de la primera consola, Magnavox Odyssey, cuando apareció el primer videojuego popular y que, para muchos, sigue siendo el primer videojuego de la historia: Pong.**[2]**

Desde entonces, la evolución de los videojuegos ha ido fuertemente ligado al hardware para los que estaban diseñados. La mayor capacidad de procesamiento, el contar con más memoria y las nuevas tecnologías para la interfaz humano-maquina, han ido haciendo que aparezcan nuevos géneros de videojuegos.
Algunos de estos géneros tienen como base las mecánicas creadas por un título que en su día fue un éxito a todos los niveles, ya que, junto a su rompedor concepto, solía acompañarle un apartado técnico inaudito para su época.
También ha habido una tendencia clara en cuanto a la dificultad y el desafío que suponía completar un videojuego. Salvo excepciones, los juegos aparecidos en la última década han dotado a los jugadores de todo tipo de ayudas y facilidades para poder progresar, llegando a haber títulos que prácticamente consisten en recorrer un pasillo. Esta tendencia también ha atraído a un público más casual, popularizando aún más la industria y creando perfiles que eran impensables hace unos años.

Perfiles como el de los **streamer** de videojuegos, cuya actividad principal es jugar y comentar partidas para que otros los vean, no hace más que ratificar una tendencia en la que la gente tiene tal pereza que prefiere ver como otros juegan a los videojuegos en vez de hacerlo ellos.

Y es que la industria prefiere invertir en lo que les funciona en estos momentos: videojuegos largos, con muchísimos contenidos que, por si fuera poco, son ampliados tras sus lanzamientos. Con miles de cosas por hacer y de horas de juego que hay que justificar alargando tareas superfluas y aburridas que más que un desafío, suponen una pérdida de tiempo y no aportan nada.

Volviendo a la época de los salones recreativos, nos retrotraemos a los videojuegos **arcade.** Título en los que se pagaba por partida, diseñados para partidas cortas, pues el volumen de negocio dependía de ello. Para ello, los

desarrolladores se centraban en el punto que mejor podía enganchar a los jugadores de la época: la diversión.

Hoy en día, los juegos ya han dejado de ser un pasatiempo para convertirse en un trabajo. Hay **gamers** que han profesionalizado la actividad mediante el streaming o la competición. Dedicándose a videojuegos enfocados a los **esports** para la competición. Videojuegos de corte multijugador que enfrenta a equipos con hordas de seguidores cual hinchas de futbol y que cada día atrae a más público. El diseño de estos videojuegos suele tener unas reglas muy definidas, un tiempo limitado por partida y una alta parametrización del avatar para poder ser competitivos, lo cual, resta tiempo de juego para probar, configurar y balancear la configuración del jugador.

Por otro lado, tenemos a los streamers, que suelen jugar a títulos con partidas más largas y que conlleva mucho tiempo de juego. Estos títulos suelen tener una narrativa más elaborada que su jugabilidad, lo que invita a los espectadores a descubrirlas en manos de otra persona para ahorrarse el tiempo.

Desde hace unos años, existe un movimiento que cada vez es más popular, el de los estudios de desarrollo **indies**, los cuales, gracias a sus bajos presupuestos, buscan mecánicas innovadoras junto a una jugabilidad directa procurando no caer en el contenido superfluo de muchas superproducciones o títulos **AAA.** Anteponen la diversión ante el resto de elementos del juego. Incluso muchos de ellos, recuperan el estilo gráfico, sonoro y jugable de títulos de la época de los 16 bits, curiosamente pertenecientes a la época de esplendor de los salones recreativos.

Con CarOut! Pretendemos recuperar la filosofía de los juegos arcade de antaño, en los que se buscaba una diversión directa, sin complicaciones de miles de configuraciones. Sin tener que jugar bastantes horas para poder alcanzar el potencial entretenimiento del título, sin cargas entre fases ni largas instalaciones que rompan el ritmo. Buscamos un juego rápido, directo y al ser posible, con amigos al lado para aumentar el desafío y pasar un buen momento.

Otro punto que buscamos atacar es el de la originalidad. Buscamos un estilo de juego que se diferencie de todo lo demás. Algo complicado, pues tras cientos de miles de juegos, es muy difícil innovar.

1.2 Objetivos del Trabajo

A continuación, listaremos los objetivos marcados para la consecución de nuestro trabajo:

- **Diseño del videojuego**: La búsqueda de un estilo de juego que se diferenciase de lo habitual, conllevó un esfuerzo en el diseño de las mecánicas. Desde las ideas iniciales hasta las primeras pruebas jugables, el concepto que se fue gestando para el juego debía acercarse lo máximo posible entre la imaginación y la implementación real.
- **Planificación temporal**: El juego debía estar finalizado con sus requisitos mínimos exigibles durante el semestre que dura la asignatura, por lo que era de vital importancia hacer una planificación correcta y realista para llegar a un buen producto final.
- **Aprendizaje**: Durante el desarrollo, seguramente encontraríamos escollos que deberíamos solventar mediante la puesta en práctica de nuevas técnicas y conocimientos que deberíamos adquirir.
- **Apartado artístico**: La inclusión de **assets** gráficos y sonoros propios era otro de los objetivos marcados. Al menos los elementos más diferenciadores del videojuego deberían estar realizados por nosotros. Con ello le daríamos un toque de distinción y personalidad al producto y nos ayudaría mejorar nuestra habilidades creativas y técnicas con herramientas de diseño.
- **Contenido mínimo**: Nos marcamos una serie de mínimos auto exigibles para que el videojuego tuviera la capacidad de demostrar su concepto. Dichos mínimos eran:
    - Mínimo de 4 vehículos diferenciados.
    - Mínimo de 4 escenarios diferenciados.
    - Mínimo de 2 modos de juegos distintos.
    - Multijugador local.

1.3 Enfoque y método seguido

Para el desarrollo del trabajo, hemos querido ajustarnos a la asignatura y desarrollarlo desde 0 durante el semestre. No hemos arrancado de trabajos previos ya que el ajustarnos a los tiempos de entrega conlleva un hándicap real que existe en el mundo profesional del desarrollo de videojuegos.
Es por ello que el diseño y la planificación se comenzaron a realizar desde los primeros días del inicio del semestre.
Además, al tratarse de un videojuego no generalista, con unas mecánicas ya consolidadas y más que probadas, hemos tenido que partir desde un proyecto en blanco.

En cuanto a la metodología que hemos seguido, la que más se ajustaba a nuestras necesidades era el método ágil **SCRUM [3]**. Aunque está pensado para trabajar colaborativamente en equipo, a nosotros nos viene bien si obviamos ciertas fases como la de “sincronizaciones diarias”, pues al estar solos, no hace falta ponernos de acuerdo con otros miembros del equipo para avanzar o corregir partes del desarrollo.

Al igual que las entregas parciales que eran requeridas en la asignatura, los proyectos Scrum se ejecutan en una serie de ciclos cortos y fijos (en nuestro caso venían siendo de un mes) en los que se debe de proporcionar un incremento en el desarrollo del producto. De esta forma, nuestro director del trabajo podía validar, aconsejar o corregir conceptos del juego con la suficiente antelación como para que no tuvieran que ser corregidos en fases más avanzadas del desarrollo. Esto mejora la planificación y minimiza el tiempo perdido en desarrollos que no llegarían con los mínimos exigibles de calidad o no tuviesen lugar conceptualmente con nuestro trabajo.

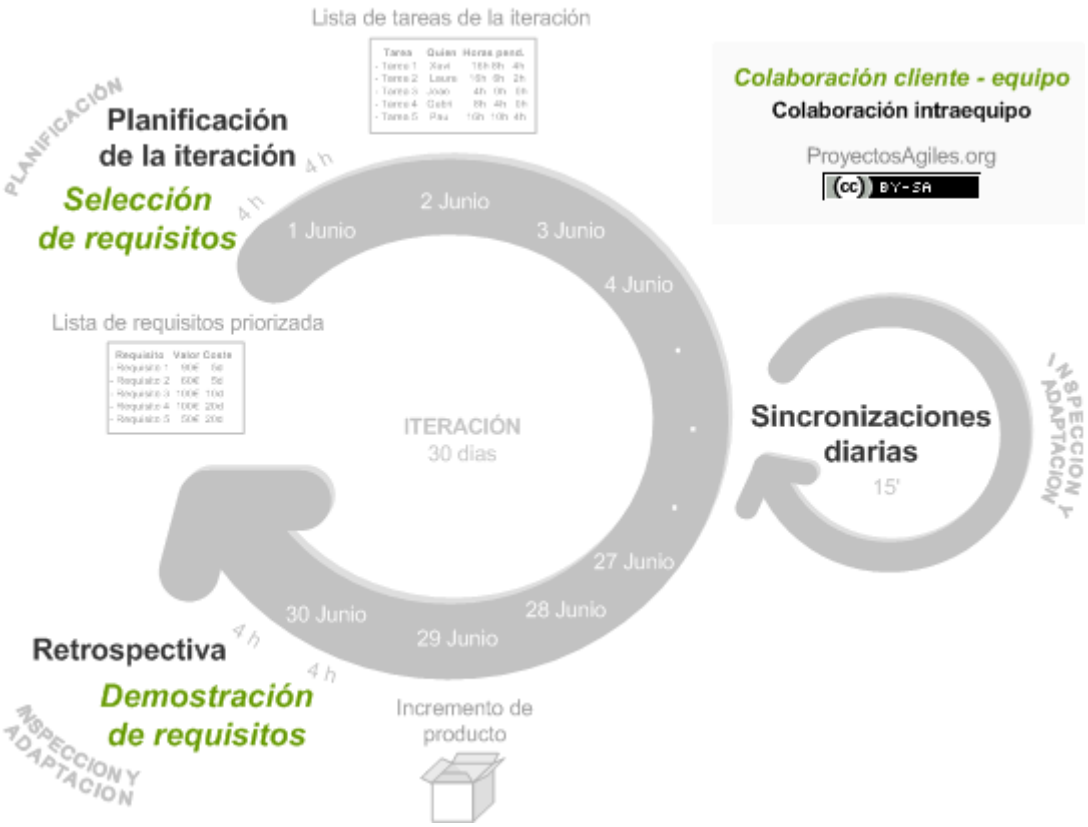


Ilustración 1 Metodología Scrum

1.4 Planificación del Trabajo

Contamos como fecha de desarrollo desde el 17 de febrero hasta el día 6 de junio de 2021, fecha tope para la entrega del trabajo fin de master. Este proyecto se estructura al igual que una asignatura del master, por lo que, tendremos que hacer entregas parciales de prácticas que pertenecerán a distintas fases del desarrollo del videojuego y que clasificaríamos del siguiente modo:

- **PEC1**: Del 17 de febrero a 7 de marzo. Corresponde a la fase de diseño. El objetivo es definir el videojuego y hacernos una idea global de sus características: género, mecánicas, personajes, modos de juego. Definimos unos objetivos mínimos y la viabilidad durante el desarrollo. Además, se valoró la posibilidad de ampliar opciones y modos de juego.

En esta primera parte trataremos de entregar lo antes posible el documento para recibir pronto feedbacks por parte del director con el objetivo de ir adelantando trabajo para la siguiente fase.

- **PEC2**: Del 8 de marzo al 4 de abril. Se realizó la entrega de una versión parcial del videojuego. Implementamos todas las mecánicas de los vehículos y la básica para los ítems sobre un solo escenario, sin atender al aspecto gráfico. Trabajamos sobre una IA de los vehículos muy básica para poder dar una idea jugable del título. Por último, se diseñó un HUD provisional y se realizaron pruebas de diseño y de colocación dentro de la interfaz.

- **PEC3**: Del 5 de abril al 9 de mayo. En esta fase hicimos una versión jugable del producto. Nuestro objetivo era que la mayor parte de los gráficos del juego estuviesen ya creadas por nosotros para darle un aspecto único, personal y que no difieran los estilos entre los distintos assets que pudiésemos encontrar. Para esta fase se crearon 3 de los 4 vehículos y escenarios planificados, todos ellos distinguibles entre sí con sus correspondientes ajustes y mecánicas propias. También se introdujo como característica no planificada una escena de presentación del escenario que se ejecuta como previo a la partida. Los distintos modos de juegos “World Championship” y “hot potato” fueron implementados.

- **PEC final:** Del 10 de mayo al 6 de junio. Es la última fase y teníamos menos de un mes para pulir los detalles del juego, así como eliminar bugs y probarlo a fondo. En esta fase de desarrollo pudimos contar con mayor tiempo de dedicación al proyecto y se implementaron tareas tan necesarias como el multijugador local para hasta 4 jugadores, la mejora significativa de la inteligencia artificial, el sistema de puntuación, los sonidos, el nuevo HUD, la introducción y el último vehículo y escenario, así como el resto de **items**. Por último, debimos de generar la memoria y el video de presentación del proyecto.

En el siguiente diagrama podemos ver una planificación global del proyecto, tanto en fecha como en número de horas estimadas. Existe una labor constante que no se ha llegado a definir, la integración del proyecto, que recoge todas las tareas necesarias de programación, gestión y control para unir los distintos elementos que lo conforman y que estaría repartida entre los periodos que comprende cada PEC y haría uso de la bolsa de horas que no llegásemos a necesitar del resto de tareas.

Nombre de la tarea	Fecha de inicio	Fecha de finalización	17.02.2021	28.02.2021	07.03.2021	14.03.2021	21.03.2021	28.03.2021	04.04.2021	11.04.2021	18.04.2021	25.04.2021	02.05.2021	09.05.2021	16.05.2021	23.05.2021	30.05.2021	06.06.2021	HORAS	
PEC1	17.02.2021	28.02.2021																		
Definición del juego	17.02.2021	24.02.2021																	18	
Documentación	25.02.2021	28.02.2021																	6	
PEC2. Version Parcial	01.03.2021	04.04.2021																		
Escenario Básico	01.03.2021	07.03.2021																	3	
Mecánicas de los vehiculos	01.03.2021	07.03.2021																	12	
IA	08.03.2021	21.03.2021																	27	
Items	15.03.2021	21.03.2021																	6	
Trampas de los escenarios	22.03.2021	28.03.2021																	12	
Documentación	29.03.2021	04.04.2021																	6	
Multijugador	29.03.2021	11.04.2021																	24	
PEC3. Versión Jugable	05.04.2021	09.05.2021																		
Graficos y Vehiculos	05.04.2021	20.04.2021																	30	
Menus e interface	12.04.2021	18.04.2021																	12	
Graficos Items	21.04.2021	25.04.2021																	6	
Graficos y Escenarios	21.04.2021	02.05.2021																	30	
Sonidos	02.05.2021	09.05.2021																	21	
Pruebas	03.05.2021	09.05.2021																	12	
Documentacion	03.05.2021	09.05.2021																	6	
PEC4. Versión Final	10.05.2021	06.06.2021																		
Correcciones	10.05.2021	16.05.2021																	24	
Incluir efectos y mejoras.	17.05.2021	30.05.2021																	30	
Documentación final	30.05.2021	06.06.2021																	15	
																			300	TOTAL

No olvidemos de que se trata de una estimación que principalmente nos va a servir a nivel organizativo, pero que, como todo proyecto ágil, suele resultar utópica y acaba modificándose.

En nuestro caso, tuvimos que dejar el terreno preparado para finalizar tareas planificadas para fases más temprana de modo que pudiésemos finalizarlas en fases finales donde dispondríamos mayor tiempo para la dedicar al trabajo.
En la fase 2 podemos apreciar como la IA y los items y el modo multijugador quedaron delegados a la fase correspondiente a la PEC4, aunque a nivel interno, las estructuras y el código ya estaban pensados y preparados para su implementación.
En la fase 3 pasó lo mismo, esta vez con tareas como el menú y la interface, algunos gráficos y la implementación del sonido.
Por suerte, los trabajos previos de planificación hicieron que realizarlos en la última fase fuera mucho más sencillo y rápido. A su vez, permitió probar con mayor agilidad la integración entre los elementos que componen el trabajo.

A continuación, pasamos a enumerar los distintos recursos que emplearemos para la realización del TFM, todos ellos implicarán un coste 0 para el autor ya que, o los poseía, o dispone de licencias temporales, de estudiante o son herramientas gratuitas.:

- **Suite Ofimática Microsoft Office**. Empleado para las tareas de documentación de las distintas PECs, las cuales serán realizadas en Word, apoyándonos en Excel y Publisher para aportar tablas o gráficos.
- **Photoshop**. Para diseño y retoque gráfico.
- **Autodesk SkeetchBook**. Para el dibujo digital de bocetos y **sprites**.

- **OBS Studio**. Para la captura de videos.
- **Adobe Premier**. Para la edición de videos.
- **Blender**. Para el modelado de los vehículos, escenarios, ítems y otros assets.
- **Trello**. Para la organización de tareas del proyecto.
- **Unity.** El motor en el que implementaremos el juego.
- **Visual Studio**. Editor de código que emplearemos para programar.
- **GarageBand.** Editor de música diseñado para Ipad.
- **Gitlab.** Utilizado para el control de versiones y backup en la nube del proyecto, a parte, será accesible para el director para que éste pueda ver en cada momento el estado actual del juego.
- **Blogspot.** Plataforma de Google para creación de blogs. Con dicha herramienta crearemos y actualizaremos el showcase del juego.

1.5 Breve sumario de productos obtenidos

El resultado final del trabajo de fin de master es una versión del juego completa. Si bien en cuanto a contenidos, hoy en día, sería insuficiente para una comercialización, si deja probar la esencia del juego. Con tan solo agregarle más contenidos que le dé variedad al juego. Con más coches, escenarios y modos de juego, obtendríamos un videojuego indie bastante completo. Ideal para consolas o incluso para plataformas móviles.

Junto a la entrega final del juego se ha elaborado la presente documentación, así como el trailer del juego y su video presentación.
Durante el desarrollo, hemos ido proporcionando distintos materiales para su evaluación y la presentación de los avances.
Entre dichos materiales, se han elaborado informe de progresos, videos demostrativos y un blog a modo de showcase donde hemos ido publicando los hitos que íbamos logrando.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos, procederemos a describir el proceso creativo y su implementación para todas las áreas que han compuesto el diseño y desarrollo del videojuego.

1. **Estado del arte.**

   Estudio del género del juego y su estado actual dentro de la industria. También realizaremos un análisis de las tecnologías que hemos empleado para su implementación.

2. **Definición del juego.**

   Veremos cuáles son sus mecánicas, en cuales juegos nos hemos influenciados, los conceptos empleados y el diseño artístico que hemos seguido.

3. **Diseño técnico.**

   Describiremos el apartado técnico del juego, así como la arquitectura de procesos que hemos diseñados y el resto de componentes que forman el juego.

4. **Diseño de niveles.**

   Presentaremos el diseño y las peculiaridades de cada nivel.

5. **Conclusiones.**

   Narraremos cuales son las conclusiones que hemos obtenido tras el desarrollo del juego y cuales han sido los problemas que nos hemos ido encontrando, así como su soluciones o alternativas.

6. **Manual del usuario.**

   A modo de anexo, definiremos cuales son los requisitos mínimos probados para la correcta ejecución del juego y la guía para los usuarios puedan jugar.

2. Estado del arte

2.1 El género

No es fácil en un primer momento clasificar el título en uno de los géneros habituales. Es un juego en el que los actores son coches, pero no es un juego de conducción al uso, pues no existe un trazado, ni un contador, ni tiempo, ni gana el primero que llega a meta.

Es un juego de combates, pero nuestros avatares no son seres humanoides que realicen distintos ataques y ni siquiera tenemos una barra de energía en la que, al agotarse tras recibir múltiples ataques, nuestro avatar desfallezca.

Lo que sí tenemos es un juego con mecánicas sencillas, fácil de controlar y de comprender, con partidas rápidas e intensas, y comportamientos y físicas que no se asemejan a la realidad, alejándose de la simulación, por lo que, si debemos de englobar el juego en un género, lo haríamos dentro de los arcades deportivos.

Hoy en día es muy difícil innovar y presentar propuestas únicas. Son cientos de miles de videojuegos desarrollados durante la historia, y a no ser que la interfaz máquina-humano sea novedosa y abra nuevos caminos, prácticamente todo está inventado hoy en día.

La idea original viene en un momento en el que el diseñador se para a observar a qué estaba jugando en un momento puntual con su sobrino y cae en que sería una buena idea plasmarlo en un videojuego. Inmediatamente empiezan a surgir ideas que vienen de otros videojuegos que le gustó de joven y que le gustaría que se vieran reflejadas.

El primer referente es ***Destruction Derby***, un videojuego de Reflections Interactive aparecido para PC y consolas de 32bits. La demo que se distribuyó, contaba con una fase con una pista redonda donde los coches debían de chocar entre si hasta destrozarse de forma que ganaba el último que se pudiese mover.

Ilustración 2 Destruction Derby

Aunque es una idea divertida, este juego tenía su grado de simulación. Nuestro concepto pasa de la destrucción a la expulsión de los escenarios. Sin tener en cuenta los daños que puedan sufrir los vehículos.

Otro juego de la época que llamó la atención por su vertiente multijugador, la diversión y hacer referencia a otros videojuegos, fue **Worms.**Especialmente recordado fue el ataque que había disponible en el que nuestro gusano podía realizar un Hadouken (poniéndose antes una cinta roja como la de Ryu de **Street Fighter 2**).

Ilustración 3 Worms

En base a esto nos pareció buena idea, ya que el videojuego es una especie de combate de coches, que el aspecto de estos hiciera referencia a luchadores clásicos de juegos de luchas como Ryu, Terry Bogard de Fatal Fury o Goku de Dragon Ball y que a su vez tuvieran la posibilidad de realizar algún tipo de ataque especial propios de estos luchadores.

Otras referencia procedente de juegos de luchas vendrían de los títulos de Capcom, Power Stone y Power Stone 2, en especial su segunda parte, por su multijugador, la cantidad de elementos interactivos de los escenario y la posibilidad de sacar del escenario a los contrincantes, objetivo principal de otro título de lucha, Super Smash Bros, en el que también se emplea luchadores de videojuegos de títulos dispares y en el que muchos de ellos, poco tienen que ver con el género de la lucha.

Ilustración 4 Power Stone 2

Ya con el concepto en la mente nos pusimos a buscar juegos que compartiesen conceptos, y cuando ya parecía que iba a ser una idea única encontramos de casualidad un juego que guarda una idea parecida: Car Fight, de IO[4]

CarFight es un juego multijugador web competitivo. Las diferencias con las mecánicas radican en el comportamiento de los vehículos, que es completamente irreal y además llevan siempre la misma velocidad, sin posibilidad de acelerar ni frenar, lo que incentiva que puedas salirte de la pista sin que te eche ningún rival. La otra diferencia notable es el control, en el que solo tendríamos que señalar con el ratón o el dedo (en una pantalla táctil) el sentido al que queremos que se dirija el vehículo. En nuestro concepto de juego, los controles son más clásicos y estarían pensado para teclados o **gamepads**.

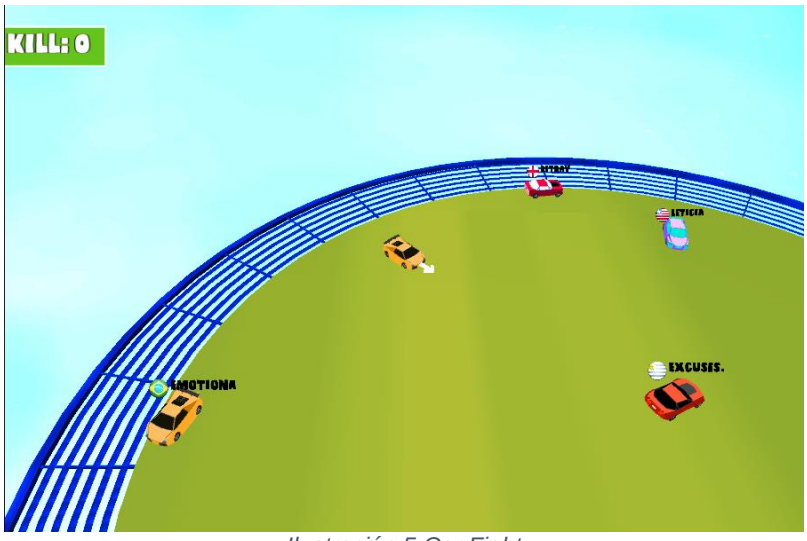


Ilustración 5 Car Fight

Con dicho título como rival más directo y pensando en un título inicial para el juego llegamos a la idea de nombrarlo: “Ring Out Car”. En alusión a la dinámica que introdujo Virtua Fighter de ganar un combate al sacar al rival del tatami y cuya voz del comentarista con su “Ring Out!”, quedó grabada en la mente de los aficionados. Fue buscando referencias a dicho termino, cuando nos encontramos por casualidad una máquina recreativa que nos era desconocida hasta aquel momento: “Ring Out 4x4” para la placa Naomi de SEGA y que no llegó a salir de Japón.

Ilustración 6 Ring Out 4x4

El juego se asemeja conceptualmente bastante a nuestra idea (incluso al título que teníamos pensado). Era un título competitivo de hasta 4 jugadores en los que solo uno podía quedar dentro del ring para ganar la partida. La recreativa

constaba de cuatro puestos con volantes y pedal y en vez de coches, enfrentaba vehículos más pesados como palas o 4x4 que disponían de unos parachoques con los que podían golpear.
Evidentemente, este hallazgo nos hizo replantearnos el título dejando uno que pensamos que gana en sencillez e impacto: "**Car Out!**"

2.2 Tecnología

A la hora de seleccionar el motor de juego que íbamos a emplear, hemos tenido en cuenta diversos factores basándonos en el concepto del juego.

Aunque teníamos experiencia con suites para el desarrollo de videojuegos, estas estaban completamente obsoletas.
Nos referimos a Div Game Studio. Fue lanzado en 1998 por la empresa Hammer Technologies **[5]**. Contaba con un lenguaje de programación propio con estructuras muy parecidas a las usadas por Pascal o C**.** Fue la primera suite que se popularizó en España debido a la campaña de marketing y la venta directa en quioscos. Además del lenguaje, el paquete contaba con herramientas como el compilador, un diseñador de sprites y un programa de edición de sonido. Aunque se llegaron a sacar algunos pocos juegos profesionales, la herramienta no llegó a contar con el apoyo de los desarrolladores profesionales.
Existen proyectos que intentaron resucitar el espirito de DIV, como el proyecto Fénix o Bennu., pero a día de hoy están olvidados.

El hecho de que queríamos que el juego se reprodujese en un entorno tridimensional hizo descartar todos aquellos motores especializados en videojuegos 2D como Game Maker.

La decisión de que el juego fuese tridimensional radicaba en, por una parte, facilitarnos las tareas a la hora de programar las físicas, y por otra, el facilitarnos las tareas gráficas.
El desarrollo de un juego tridimensional nos requeriría tan solo el modelado completo del vehículo. Esta tarea es a priori más rápida y sencilla que tener que pintar todos los ángulos de los distintos vehículos.
Podíamos haber utilizado una vista aérea en el juego para así podernos haber ahorrado el dibujado de los vehículos desde los distintos ángulos, pero en nuestro concepto, visualizábamos el juego desde una vista isométrica para darle una mejor apariencia visual, más espectacularidad y que el jugador pudiese moverse por mayor espacio sin necesidad de que los sprites se viesen minúsculos.

Quedaban por tanto los 2 motores de videojuegos más populares para el diseño de videojuegos en tres dimensiones: Unity y Unreal Engine**.**

Las ventajas con la que ambos contaban eran la gratuidad para proyectos pequeños y no comerciales, su facilidad de uso y la gran cantidad de documentación, artículos y assets que podemos encontrar para ellos.

Si bien Unreal es más potente que Unity en cuanto a resultados gráficos y está respaldado por compañías profesionales que lo eligen para el desarrollo de sus juegos AAA**.** Contábamos con el hándicap de que no teníamos experiencia previa desarrollando con él.
El aprendizaje de dicha herramienta iba a ser muy costosa en cuanto a tiempo de aprendizaje. Debido al limitado tiempo para realizar el trabajo, fue una opción que tuvimos que descartar.

Queda por tanto Unity como nuestro motor ideal para el desarrollo del videojuego. A sus ventajas ya descritas anteriormente, hay que unirle la experiencia adquirida durante el master. Los conocimientos adquiridos durante la formación en varias de sus asignaturas agilizarán los tiempos de desarrollo del proyecto.

Por último, otra de las ventajas de Unity es su capacidad y facilidad para desarrollos multiplataformas. Aunque por practicidad, el videojuego original iba a ser desarrollado para PC con sistema operativo Windows, el género del juego es muy apropiado para consolas. Plataformas como la Nintendo Switch con las que el modo multijugador local es muy accesible, serían ideales para acoger el título. Con poco trabajo y el coste derivado de las licencias, sería muy rápido y sencillo portar el juego.

Aunque empezamos a utilizar la versión 2019.4.10f1 en el primer semestre, los requisitos de otras asignaturas nos permitieron actualizar a 2020.2.2f1 a mitad de curso para finalmente utilizar la versión 2020.2.4f1 para las asignaturas restantes.

La elección de la versión 2020.2.4f1 como base para nuestro desarrollo fue realizada en base a la experiencia adquirida y en tratarse de una versión estable con la que ya habíamos trabajado sin problemas en prácticas anteriores. Lo que teníamos claro es que no debíamos de cambiar de versión durante el desarrollo del proyecto, ya que los desarrolladores no lo recomiendan por los posibles cambios internos que puedan afectar al proyecto. No obstante, a la hora de empezar a prototipar el juego, elegimos por error la versión 2020.2.2f1. Cuando nos dimos cuenta, procedimos a actualizar lo que llevábamos hecho a la versión 2020.2.4f1. Fue en ese momento en el que pudimos corroborar el consejo de no cambiar de versión una vez iniciado un proyecto, ya que, incluso para un salto de versión tan pequeño, el proyecto empezó a presentar diferencias en el tratamiento de la iluminación, dejándonos unos tonos pasteles en los colores que no eran de nuestro agrado. Una vez corregidos, continuamos el trabajo con dicha versión.

3. Definición del juego

3.1 Descripción del juego.

CarOut! es un videojuego en el que controlamos un coche dentro de un escenario delimitado al que llamaremos ring y en el que, mediante choques contra otros vehículos y el uso de trampas, ítems y ataques especiales, deberemos de expulsar del escenario al resto de contrincantes de modo que seamos los últimos en quedar sobre el ring.

Cada partida presenta enfrentamientos de 4 vehículos y está ideado para partidas rápidas y directas donde se pretende que, en su modo multijugador, el jugador consiga la experiencia definitiva para pasar momentos divertidos con los amigos.

La posibilidad de añadir nuevos modos con reglas nuevas da variedad a un título con unas mecánicas básicas muy definidas.

3.2 Historia y ambientación.

Como la mayoría de los arcades, la historia del juego es secundaria o su narrativa suele ser demasiado ligera con el objetivo de entrar en acción lo más rápido posible. Recordemos que lo que prima es la interactividad y el tiempo que estemos contando una historia sin medio de la interacción, es tiempo de juego que estamos perdiendo.

Si bien desde un primer momento nos planteamos darle algo de **lore** al juego, con el origen de esta nueva modalidad deportiva, tras analizarlo, nos pareció algo irrelevante y forzado. Ya hemos visto ejemplos anteriores en los videojuegos y sobre todo, en el salto de algunas licencias al cine donde el resultado no ha sido nada positivo, convirtiendo productos en fiascos comerciales. Un ejemplo de ello es la conversión del popular juego de mesa Hundir la flota (Battleship) donde tratar de meter con calzador una historia sobre las mecánicas de un juego de mesa no gusto ni al público ni a la crítica.
Y es que hay géneros como los juegos tipos puzle (Tetris, Columns, Puzzle Bubble…) que no necesitan de una narrativa elaborada para hacerlos atractivos. Con los videojuegos arcade deportivos, viene ocurriendo lo mismo. No necesitan más historia que la de la celebración de un campeonato y en el que obviamente, el jugador quiere ganar.

En cuanto a la ambientación, al estar el juego centrado en los combates entre coches, quisimos coger referencia de juegos de luchas **VS** clásicos. De este modo hacíamos un guiño a uno de nuestros géneros favoritos. Además, esta decisión ayudaría a familiar al jugador rápidamente con el juego y despertaría la curiosidad de los entusiastas del género, los cuales, pueden sentir la curiosidad de ver como se ha trasladado esta idea al juego.

Entre los detalles, que veremos más en profundidad en las próximas secciones, podemos destacar:

- Una **intro** en la que hacemos un homenaje a la primera introducción de Street Fighter 2
- Coches caracterizados como personajes icónicos de juegos de luchas.
- Ataques especiales archiconocidos.
- Escenarios que recordarán ciertos momentos de la historia de algunos juegos o series.
- HUD que recuerdan más a un juego de combate que a uno de coches.

Aunque en una primera fase del desarrollo se optó por modificar los nombres de los vehículos y de los ataques especiales ligeramente para que no fueran los originales, a posteriori se optó por no cambiarlos por los originales debido a tratarse de un trabajo no comercial y a que los nombres son todos comunes. Ninguno ha sido registrado pues no procede. Incluso los ataques especiales son frases cortas en otros idiomas como el japones o el inglés y hay hasta una de ellas que corresponde al nombre del que fuera un rey hawaiano.

3.3 Modos de juego

CarOut! posee 2 modos de juegos:

- **World Championship**: Sería el modo clásico del juego. Es un campeonato en el que se van recorriendo los distintos escenarios del juego. En cada nuevo campeonato, estos se secuencian de forma aleatoria. Cada escenario es superado cuando uno de los 4 vehículos gane 3 rondas. Recordemos que se gana una ronda cuando eres el último vehículo en el escenario (o también al ser el último en caer). Por cada ronda que se gane, el jugador obtiene 1 punto. Si gana en ese escenario se le premia con otros 3 puntos. Al finalizar todos los escenarios, ganará el jugador que más puntos haya obtenido.

- **Hot Potato**: También denominado patata caliente. Hemos incluido este modo para demostrar las posibilidades lúdicas que puede presentar el título con pequeños cambios en las reglas. En este caso, nos encontramos en un ring cerrado del que no podemos salirnos. La mecánica de la partida consiste en 2 fases:
    1. Llegar lo antes posible a la zona marcada como “safe zone”. El jugador que no haya llegado a esa posición antes de los demás jugadores será el portador de la bomba.
    2. La bomba explotará en un tiempo delimitado pero aleatorio en cada ronda. Por lo que no sabemos en qué momento exacto puede explotar. Nos hemos asegurado de que el tiempo límite no sea lo suficientemente largo para agilizar las partidas y hacerlas rápidas dándole además mayor tensión. El portador de la bomba puede pasarle esta a los otros jugadores simplemente chocando directamente con ellos. Para hacer el sistema de juego más robusto hemos dado un pequeño margen de tiempo y un sistema

de colisión para que si se quedan los coches enganchados o los impactos están siendo instantáneos, no se estén pasando la bomba de forma continua.

Para agregar más tensión y que el jugador pueda hacerse una idea del tiempo restante, la velocidad de reproducción de la música se ira acelerando dinámicamente en función del tiempo restante.
Cuando la bomba le explote al jugador portador, este se saldrá de la partida y la ronda se reiniciará con los jugadores restantes hasta que quede solo uno. La aparición del “safe zone” es totalmente aleatoria en cada ronda y beneficiará a unos jugadores a veces y otras no.

3.4 Vehículos

El **rooster** original está compuesto por 4 vehículos. Todos ellos caracterizados, balanceados y con distintas parametrizaciones para distinguirse entre sí.
Existen 4 parámetros que diferencian la manejabilidad de cada vehículo. Estos son:

- **Peso**: Determinará acciones como los impactos y la aceleración. Tener mayor peso implica que al vehículo le costará acelerar más. Sin embargo, su mayor peso es una ventaja ante los impactos de los rivales, pues a mayor peso, serán despedidos menos distancia.

- **Freno**: A mayor potencia de freno, antes podemos parar el vehículo.

- **Velocidad máxima**: Tener mayor velocidad máxima también implica el poder asestar choques con mayor fuerza, a su vez, será más costoso frenar si no hemos acertado en el objetivo.

- **Ataque especial:** Cada vehículo posee uno propio. Están basado en famosos ataques de los personajes en los que están inspirados los vehículos. El comportamiento de cada ataque es completamente distinto para que el jugador pueda emplear las dinámicas con las que se encuentre más cómodo o les sean más útiles.

RYU

Basado en el protagonista de la saga Street Fighter de Capcom. El diseño del vehículo está basado en el Lexus LS 400 que, a su vez, aparecía en la fase de bonus de Street Fighter 2. De esta edición de la saga, también recuperamos los colores clásicos de Ryu para el texturizado del coche: color blanco predominante junto a una línea roja en el frontal del capot que emula la cinta que llevaba el personaje en la cabeza. Por último, el parachoques negro representa su color de cabello.

Ilustración 7 Ryu

El ataque especial es el Hadouken, una bola de energía que es lanzada desde el frontal del vehículo.

Las características del vehículo son las siguientes:

PESO	FRENO (torque)	VELOCIDAD MAXIMA
1000 KG	70000	140 km/h

Es un vehículo ligero, el más rápido de los 4, pero también es el que menos potencia de frenada posee, lo cual es su mayor hándicap.

TERRY

Camioneta que comparte los colores de Terry Bogard, el luchador protagonista de la saga Fatal Fury.
Hemos escogido este tipo de vehículo debido a que muchos medios han solido relacionar al personaje con la profesión de camionero, debido quizás, al atuendo que ha lucido en casi todas sus apariciones.
Al vehículo le hemos incluido una especie de visera en la parte trasera de la cabina del conductor para hacerle un guiño a la gorra que suele llevar el personaje.
El rojo es color predominante en el vehículo, junto a detalles amarillos, azules y blancos que hacen un guiño al cabello, los pantalones y la camiseta del luchador.

Ilustración 8 Terry

Su ataque especial es el Power Geiser**,** como su propio nombre indica, es un geiser de fuego que si lanzado a la distancia correcta respecto al rival, puede mandarlo muy lejos. Este ataque también es uno de los ataques mas poderosos de Terry Bogard. Corto alcance, pero poderoso y otros usos estratégicos que el jugador podrá descubrir durante la partida.

Estas son las características del vehículo:

PESO	FRENO (torque)	VELOCIDAD MAXIMA
1300 KG	90000	120 km/h

Es el coche más pesado del rooster, sin embargo, tiene buen balance entre su potencia de freno y la velocidad máxima que puede alcanzar, lo que le proporciona una buena respuesta antes los embistes que podamos realizar cerca de los límites de los rings**.**

GOKU

Son Goku o Kakarot, como es nombrado en su planeta natal, es el protagonista de la serie Dragon Ball, escrita y dibujada por Akira Toriyama
Para su representación en CarOut! Hemos optado por diseñar un vehículo familiar ya que de todos los personajes en los que nos hemos inspirado, es el único que tiene una familia y queríamos diferenciar lo máximo posible todos los vehículos y que a su vez tuviera algo de sentido la elección de un modelo u otro.
Los colores del coche están basados en el atuendo más común del personaje en su época adulta. El naranja es el color predominante, optando como color secundario el azul. Este color también era empleado por el luchador en prendas secundarias como muñequeras, botas, cinturón o camiseta interior.
En el techo hemos colocado filas de pirámides negras que simulan el peculiar cabello del guerrero.

Ilustración 9 Goku

El kamehameha es el ataque especial de este tercer vehículo. En concepto muy parecido al Hadouken, sin embargo, este es más alargado y para diferenciarlo del vehículo Ryu, en vez de lanzarlo desde el frontal del coche, lo hace desde la parte trasera, emulando aquella ocasión en la que usó dicha técnica contra Picolo durante el torneo de artes marciales que cerraba el arco de Dragon Ball y cuya historia sería reanudada en Dragon Ball Z.

Estas son las características del vehículo:

PESO	FRENO (torque)	VELOCIDAD MAXIMA
1200 KG	100000	110 km/h

De todos los vehículos es el que mayor potencia de frenado posee, pero también es el que menos velocidad máxima puede alcanzar. Su alto peso unido a la potencia de frenado lo hacen muy útil para jugar en zonas arriesgadas.

KYO

El coche negro corresponde al protagonista de la saga The King of Fighters. Kyo es un joven estudiante del clan Kusanagi que poseen la habilidad de controlar el fuego.
Hemos considerado que un vehículo estilo Cadillac clásico es el que más encaja con el look y el carácter del personaje en el que está basado.

El coche es completamente negro, como la indumentaria del personaje, con algunos detalles en rojo (faros trasero y parachoques) para hacer referencia a la relación que tiene con el fuego.

Ilustración 10 Kyo

“Kurai Yagare” es el ataque especial de Kyo. Es una frase en japones que viene a significar “¡comete esto!”. Cuando se activa, el vehículo es rápidamente propulsado mientras es envuelto en llamas para posteriormente frenar casi en seco. Esta habilidad puede ser usada tanto ofensivamente, para chocar contra los rivales a una velocidad superior a la del límite del vehículo, como defensivamente si vemos que vamos a ser golpeados y necesitamos una aceleración extra para esquivar el envite del enemigo.

Las características del vehículo son:

PESO	FRENO (torque)	VELOCIDAD MAXIMA
1100 KG	80000	130 km/h

Es probablemente el vehículo más balanceado del juego, no obstante, su mayor debilidad puede venir por su ataque especial, ya que al no ser un ataque que se lance a distancia lo expone ante errores del jugador y a los rivales.

3.5 Items

Durante la partida, irá apareciendo con aleatoriedad tanto en la localización como en el tipo, un item que se podrá usar cuando queramos tan solo pulsando una tecla o botón específico para tal fin.
Los item dan ciertas ventajas a sus poseedores y las dinámicas producidas por su uso puede alterar el transcurso de una partida.
Están representados por cajas de distintos colores con un efecto de partículas ascendentes del mismo color que ayudan a distinguirlos mejor dentro el escenario. En cada lado de la caja hay dibujado un icono que representa la habilidad que nos proporciona.

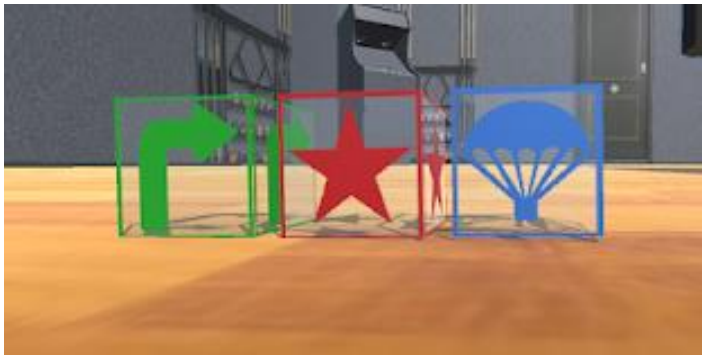


Ilustración 11 Items

Hemos decido emplear solo 3 items para que los jugadores no se saturen y tengan bien definida la habilidad que representan y el uso que se les pueda dar. Estos serían:

FINTA

Es de color verde y su icono es una flecha que cambia de dirección. Cuando lo activamos, el vehículo hace un giro instantáneo de 90 grados hacia la derecha. Este item es muy útil en ocasiones que estamos a punto de salirnos del escenario o que vamos a ser golpeados por un rival.

Ilustración 12 Icono del item finta

ATAQUE ESPECIAL INSTANTANEO

Cuando poseamos este item, tendremos a nuestra disposición la posibilidad de lanzar un ataque especial sin necesidad de haber rellenado por completo la barra destinada a ello. Su uso debe de ser estratégico, pues si tenemos la barra completa y lo activamos habremos perdido tanto el item como la acumulación de la barra en un solo ataque.
Es representado con una estrella y su color es rojo. El motivo de haber seleccionado la estrella es debido a las representaciones que se hacen en los comics con este tipo de figuras cuando se asestan golpes. El color rojo también representa agresividad y daño, es por ello por lo que optamos por él para el diseño.

Ilustración 13 Icono del Item Ataque Especial

SALVAVIDAS

El salvavidas nos permite usar una habilidad muy útil y que puede ser clave para ganar o perder una partida.
Cuando se usa, el jugador desaparece y aparece instantáneamente en el centro del mapa, de modo que, si nos estamos saliendo del escenario y somos lo bastante rápidos en reflejo, podemos ponernos a salvo. También puede ser usado defensivamente en ocasiones en la que un impacto pueda sacarnos del ring, de modo que, podemos dejar al rival completamente vendido e incluso pudiendo salir del escenario por sí mismo.
El efecto que se consigue con esta habilidad es muy parecido al que empleaba Goku de Dragon Ball con la técnica de traslación instantánea.

El color escogido para su implementación es el azul. Un color que da seguridad y calma. El icono que emplea es el de un paracaídas. Creemos que este símbolo era más representable y fácil de entender que dibujar un salvavidas de tipo flotador o cualquier otro elemento.

Ilustración 14 Icono del Item Salvavidas

3.6 Arte del juego

El diseño artístico del juego a seguido los estándares de los juegos arcade, estos son modelados bien definidos, con una cantidad de detalles media y una iluminación que permitiese vislumbrar todos los detalles en pantalla sin generar fatiga visual.

Era importante el uso de los colores para diferenciar los vehículos entre ellos, del mismo modo, debíamos resaltar los elementos del escenario mediante el uso de tonos de alto contraste respecto a los elementos cercanos, para que su visualización y diferenciación fuera rápida y efectiva.

No se ha prestado atención al detalle en los vehículos. Con esto, logramos reducir la carga poligonal y por tanto la necesidad de procesado. Esta decisión también es acertada pues los vehículos son visualizados a bastante distancia de la cámara, por lo que muchos de los pequeños detalles se perderían y serían superfluos.

El modelado de los escenarios es de baja poligonización también. Aunque son elementos de mayor tamaño que los vehículos, durante las partidas apenas se muestran los elementos externos al ring, pues la cámara se focaliza en éste para que el jugador pueda ver en todo momento que está ocurriendo.

4. Diseño técnico

4.1 Entornos utilizados

Para el desarrollo del título hemos empleado las siguientes plataformas:

UNITY

Como ya hemos comentado anteriormente, nos decidimos por dicha herramienta por motivos como su facilidad de uso, accesibilidad y la experiencia previa.
Su capacidad para desarrollar proyectos en entornos 3D lo hacía ideal para el desarrollo de nuestra idea. Además, está más que probado y es muy utilizado profesionalmente para el desarrollo de títulos indies como este que nos ocupa.

VISUAL BASIC STUDIO 2019

Este **IDE** nos provee de un entorno para codificar completamente integrado con Unity, lo cual, facilita y mejora mucho el desarrollo, ahorrándonos pues, bastante tiempo en esta parte del trabajo.
Gracias a esta herramienta podemos ver en tiempo real donde tenemos errores en el código y sus posibles soluciones.
El lenguaje que se ha empleado es C#

PHOTOSHOP CC

Entorno de retoque fotográfico que en nuestro caso hemos empleado para el diseño gráfico del juego. Con esta herramienta hemos dibujado elementos como los iconos de los ítems, retocado las texturas de los vehículos y demás elementos de los escenario y componentes del juego como el título o los avatares que aparecen en durante la pantalla de selección de vehículos.

BLENDER

Aunque en un primer momento se optó por el uso de Maya, ya que teníamos disponible una licencia de estudiante y cierta experiencia, decidimos dar el salto a Blender con vistas al futuro ya que esta herramienta es gratuita. La adaptación fue rápida y encontramos cierta ventaja sobre todo a la hora de texturizar, con tareas que pudimos hacer más rápida y fácilmente como, por ejemplo, el coloreado del modelo y su mapa **UV** directamente pasando un pincel sobre la figura. Este acto es mucho más natural y permite ver resultados al instante, aunque tiene como desventaja la precisión a la hora de afinar en pequeños detalles si el mapa de UV lo ha dejado figuras muy pegadas entre sí.

GARAGE BAND

Aplicación nativa para entornos IOS y más concretamente tablets IPAD. Con esta herramienta podemos crear, de forma muy intuitiva música y efectos de sonidos sin necesidad de conocimientos previos de solfeo o metodologías musicales tal como ha sido nuestro caso.

OBS STUDIO

Con este programa hemos podido capturar en video las imágenes del juego para posteriormente emplearlas en tareas y productos requeridos para el trabajo final del master tales como los videos de presentación de las prácticas intermedias, el trailer y el video de presentación del proyecto.

ADOBE PREMIER

Editor de video empleado para el montaje de los videos que se nos ha ido requiriendo para la asignatura. Contamos con una licencia temporal y ya estábamos familiarizados con su entorno.

MICROSOFT OFFICE 2019

Suite de ofimática que emplearemos sobre todo para la documentación. Haremos uso de Word para redactarla y Publisher para la generación de los diagramas.

4.2 Arquitectura

En este apartado vamos a presentar la arquitectura que conforma el sistema completo del videojuego.
El proyecto en Unity está dividido en un total de 15 escena que van dándose paso unas a otras según el ciclo en el que se encuentre el videojuego. En las siguientes secciones iremos abstrayendo cada proceso interno para mostrar de la forma más detallada posible el funcionamiento interno de este y el porqué de ciertas decisiones de diseño.

4.2.1 Arquitectura general

Conforma el núcleo general del producto y que posteriormente iremos desgranando por áreas.

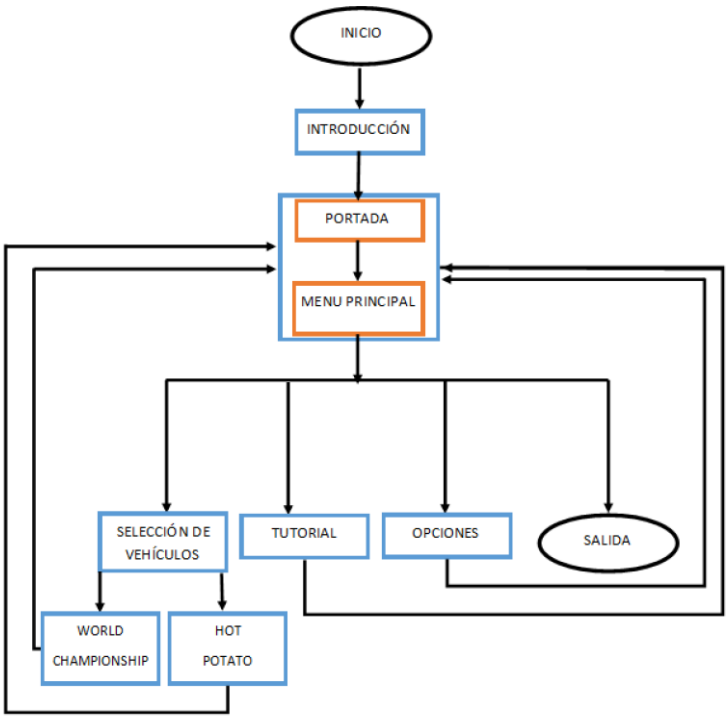


Ilustración 15 Diagrama general del juego

Es la base desde donde se desplegarán el resto de escenas que componen las distintas opciones y modos del juego.

La primera escena del juego corresponde a la introducción. Es una escena no interactiva en la que se reproduce una cinemática inspirada en la introducción del clásico Street Fighter 2 pero adaptada a nuestro título.

En apenas 15 segundos intentamos transmitir al jugador el objetivo del juego.

Acto seguido pasaremos a la siguiente escena del juego. Corresponde a la portada o título del juego y al menú principal del juego. Desde esta escena derivan otras 3 que completan la estructura y que corresponden a la escena de selección de vehículos, la del tutorial y la de las opciones, todas ellas serán explicadas en las siguientes secciones.

4.2.2 Arquitectura del menú de selección de vehículos

Esta escena está diseñada para que los jugadores puedan seleccionar un vehículo y empezar la partida.

Visualmente recuerda a la pantalla de selección de personaje típica de los juegos de lucha 1 vs 1. Lo que no era muy normal es el hecho de que en este tipo de juegos se pudiera jugar con más de 2 jugadores.

Ilustración 16 Pantalla de selección de vehiculos

El juego ha sido pensado para que el mismo vehículo no lo pudiera elegir varios jugadores. Para ello, optamos por bloquear la posibilidad de que el jugador lo elija mediante código. Los jugadores pueden estar posados encima del mismo vehículo, pero el primero que le dé al botón de confirmación será el que vaya a jugar con él. En la interfaz, el recuadro de selección quedará teñido del color que pertenece al jugador que lo ha seleccionado.

Como varios jugadores se podían posicionar sobre el mismo vehículo, optamos por diseñar un selector que se colocara únicamente en uno de las esquinas del cuadro. De este modo no habría conflictos si tuvieran el mismo tamaño y posición, ni tendríamos que tener en cuenta profundidades para que se visualizasen correctamente. En la siguiente imagen podemos ver como quedan los selectores cuando todos los jugadores están posicionados en el mismo coche.

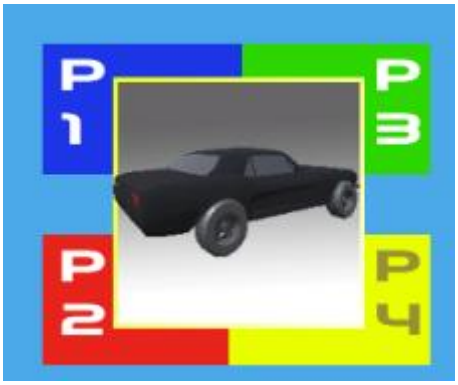


Ilustración 17 Marcadores de selección de vehiculos

Arquitectónicamente, toda la lógica de la escena se centraliza en un **game object** llamado “SelectPlayerController” el cual contiene el script homónimo.

Este script conecta los distintos elementos de la interfaz los cuales se encuentran concentrados dentro del game object “Canvas” e irán modificando en función de las acciones del usuario.
Las acciones se activan con ejes axis de los mandos. En esta ocasión con los horizontales. Que sea realizada de esta forma implica que estamos empleando un método analógico para decisiones que son puramente binarias (el pasar de un avatar a otro). Esto implicaba que cuando posicionáramos el stick a izquierda o derecha, fuera muy rápido pasar del jugador actual al primero o el último de la fila. Era complicado seleccionar algún vehículo intermedio debido a que Unity no tiene de serie un método por defecto para detectar como una única acción el desplazamiento de un eje. Es por ello que hemos tenido que implementar mediante el uso y control de variables booleanas dicha acción para que funcionara tal y como debería hacerlo.

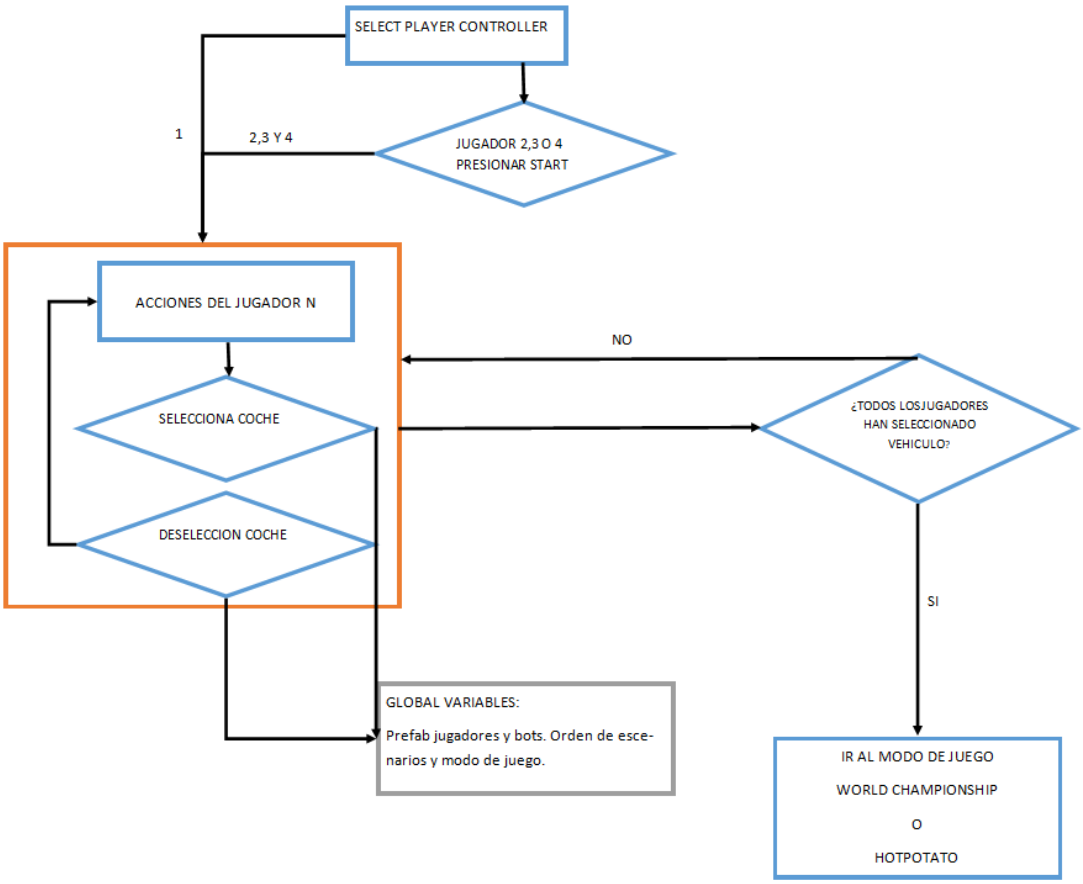


Ilustración 18 Diagrama de selección de vehiculos

Este script también almacena en un vector los **prefab** de los vehiculos, tanto de los jugables como los modelos bots que serán controlados por la **IA**. Cuando se confirma la selección de un vehículo por parte de un jugador, el sistema identifica el número de la posición de dicho vehículo y carga el prefab en el vector que contendrá los vehículos de los jugadores dentro del game object **Global Variables**.

Este game objects no se destruirá durante el cambio de escena y a parte de los, anteriormente mencionados, game object con los prefab de los vehículos de los jugadores, también almacenará la puntuación de estos para el modo campeonato, así como el orden en el que los escenarios que serán jugados.
Este orden es aleatorio y se decide cada vez que entramos en esta escena.
Para tal fin almacenamos en un array de cadenas de texto el nombre de las escenas de presentación que preceden a las partidas y que veremos más adelante.
Por último, también registraría el número de fase en el que se encuentra la partida, para llevar un control, y el modo de juego. Este valor viene asignado desde la escena de la portada y determinara con un 1 si la partida es en modo "*world championship*" o con un 2 si es en el modo "*Hot Potato*". Esta decisión de diseño deja la puerta abierta a agregar fácilmente nuevos modos de juego al título sin necesidad de modificar el sistema de selección de vehículos.

4.2.3 Arquitectura del modo “World Championship”

Este modo consiste en la sucesión de partidas en distintos escenarios que han sido seleccionados previamente, y de forma aleatoria, en el menú de selección que vimos en la sección anterior.
Se cambia de escenario cuando un jugador ha ganado el número de rondas prefijadas en el juego (3) o definidas por el usuario desde el menú de opciones.
Los datos de puntuación del campeonato se irán almacenando en el objeto no destructible, tras los cambios de escena, llamado Global Variables.
La estructura de una partida en este modo seguiría el siguiente patrón:

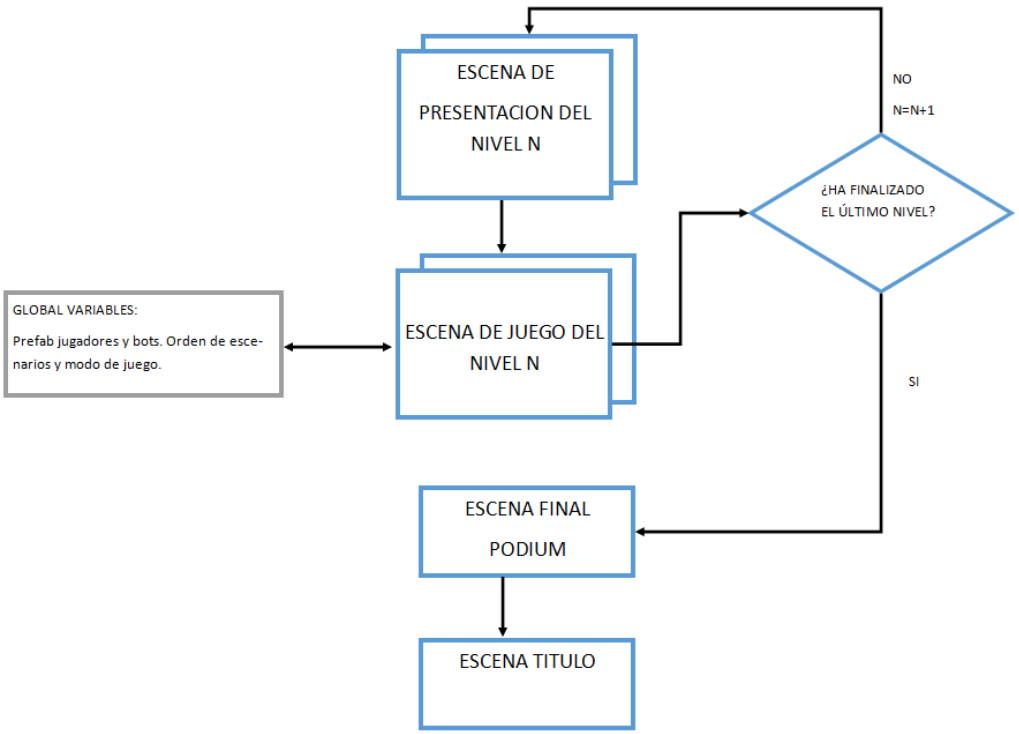


Ilustración 19 Diagrama del modo World Championship

Cada nivel está compuesto por 2 escenas: una dedicada a la presentación del escenario y que ejecuta una cinemática mostrándolo mientras aparece el ranking actual y el nombre de este.
La otra contiene el juego en sí y su lógica que veremos en la siguiente sección.
Cuando un jugador gana en ese nivel, se le asigna una puntuación extra de 3 puntos y desde el controlador de juego “Game Manager” (que explicaremos en la siguiente sección) se comprueba si se han superado todos los niveles.

Si no se han superado todos los niveles accederemos al siguiente siempre desde su escena de presentación. En caso contrario, se cargará la escena final del juego en la que se muestra el ranking final y los coches en el lugar del podio que les correspondan. Acto seguido volveríamos a la pantalla del título, completando de esta forma uno de los ciclos del juego.

4.2.4 Arquitectura de una ronda en el modo “World Championship”

Las partidas en el modo “*World Championship*” están compuesto por un número X de rondas y es superada cuando un jugador gana N rondas, siendo N un valor que puede definir el usuario desde el menú de opciones y que si no modifica está prefijado en 3.

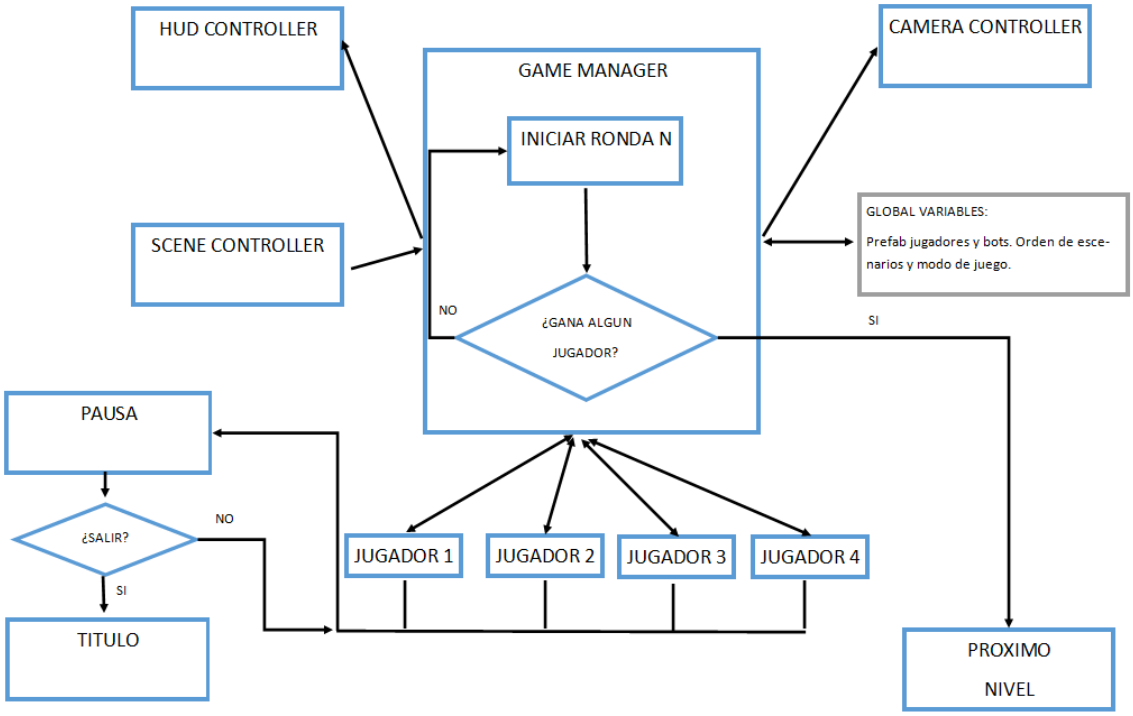


Ilustración 20 Diagrama del controlador de juego para el modo World Championship

La clase *Game Manager* es la encargada de gestionar y controlar el sistema de juego general. Entre sus funciones, podemos destacar las siguientes:

Al iniciar una ronda, el sistema ordena los jugadores del 1 al 4, siendo los primeros puestos otorgados a los jugadores humano y el resto a los manejados por la máquina. Posteriormente los instancia en los puntos de **spawn** que tienen prefijados. Estos están colocados estratégicamente para que alguno de los jugadores no tenga ventaja sobre los demás.

Cuando uno de los jugadores cae del ring o es lanzado a una distancia determinada fuera de éste, se produce un “car out!” su game object se desactiva y el sistema hace un recuento para ver cuántos jugadores quedan en la partida. Si solo queda uno, se cierra la ronda otorgándole un punto para la clasificación general (a la variable que tiene definido para ello en el game object Global Variables) y un punto en su contador de rondas ganadas. Si ha ganado el número N de ronda, ganaría la partida y pasaría a la siguiente fase. En caso contrario se iniciaría una nueva ronda mostrando mediante llamadas al HUD del juego quien ha ganado la ronda.

Por otra parte, tenemos la clase Scene Controller. Está disponible desde el game object padre del ring y es la encargada de definir los límites del escenario para la gestión de las salidas del ring que se hace desde el Game Manager. Estos límites están prefijados y emplea distintas variables en función de si el escenario tiene forma rectangular (tiene en cuenta el ancho y el largo del escenario) o circular (tiene en cuenta el radio del escenario.

Scene Controller también maneja el sistema de generación de ítems. Estos aparecerán en el escenario en zonas aleatorias. También, de forma aleatoria, se seleccionará uno de los 3 tipos disponibles. El tiempo de generación de los ítems es aleatorio dentro de unos límites y nunca se generarán más de uno al mismo tiempo. De esta forma buscamos el conflicto entre los jugadores para hacerse con él, ya que, si hubiera varios a la vez, irían a por el que mejor les vendría pudiendo evitar el enfrentamiento.

Las cámaras son gestionadas mediante la clase Virtual Camera Player Controller que podemos encontrar en el game object CameraGame. Hace uso de las cámaras virtuales de Cinemachine. Emplea durante la partida una cámara con perspectiva isométrica aérea de tipo ortográfica que sigue a los jugadores que queden en pantalla y hace zoom en función de la distancia en la que se encuentren entre ellos. Cuando se finaliza una ronda, se cambia la cámara principal por una que lleva cada coche. De este modo queremos darle más dinámica al acontecimiento y mostrar quien ha sido el ganador durante unos segundos hasta que comienza la nueva ronda.

Por último, la case HUD Controller es la encargada de comunicarse con los game object de los jugadores y el Game Manager para mostrar todos los datos que el jugador necesita, siendo estos:

- El avatar del jugador.
- Número de jugador.
- Barra de carga del ataque especial.
- La posesión o no de un item y el tipo.
- El número de ronda ganadas y las necesarias para ganar la partida.

Por último, cada escenario puede contener una sería de prefab que pertenecen a trampas que tienen varios cometidos y acciones diferentes.

4.2.5 Arquitectura de una ronda en el modo “Hot Potato”

En el modo Hot Potato, el desarrollo de la partida tiene algunas reglas que difiere del modo anterior.

Existe una clase llamada *Game Manager Hot Potato* que se complementa con la clase Game Manager y gestiona el transcurso de una partida en este modo.

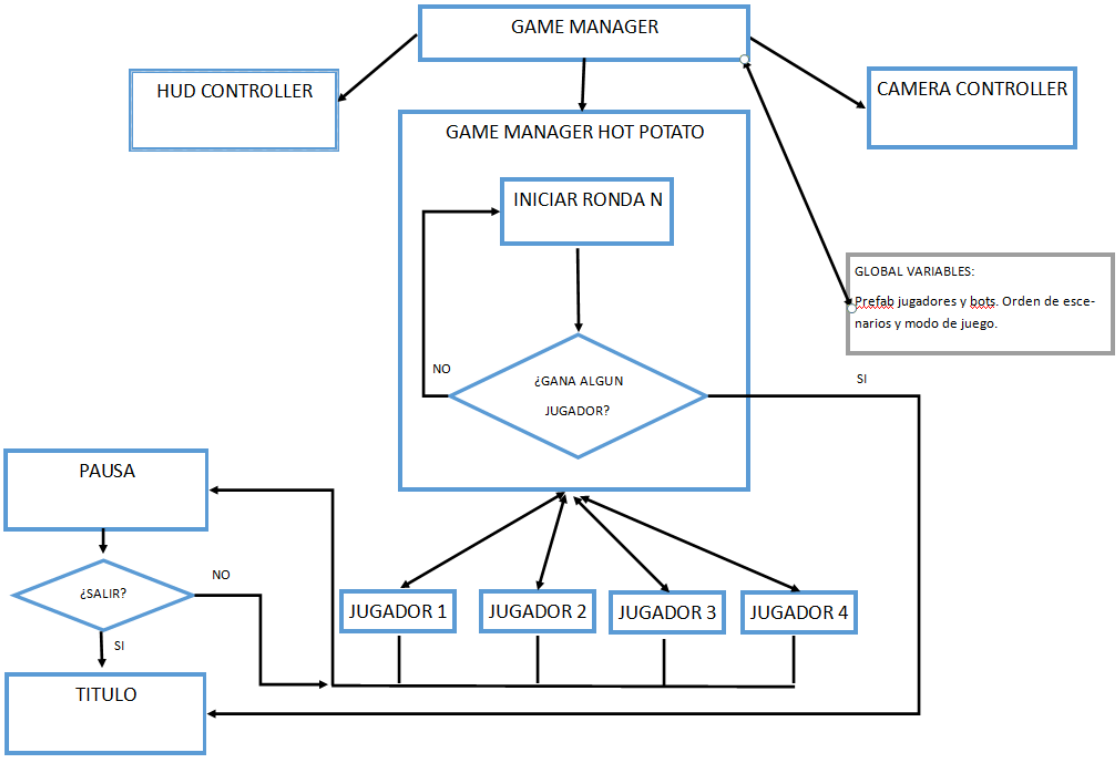


Ilustración 21 Diagrama del modo Hot Potato

Hasta la instanciación de los jugadores, todo se mantiene. Sin embargo, al comenzar una ronda, se instancia un prefab llamado SafeZone, el cual coloca en pantalla en una zona aleatoria del escenario un área que denominamos Safe Zone. El sistema controla quien es el último jugador que ha tocado esa zona e instancia un prefab llamado Bomba en el jugador que no lo ha hecho.

Este prefab define un tiempo en el que explotará entre un rango de tiempo que hemos prefijado. El jugador no tiene más pista del momento en el que se va a producir la explosión que la sonora, pues la música que acompaña la partida y que se reproduce desde el origen de sonido que le hemos añadido a la cámara, irá aumentando su velocidad dinámicamente en función de este.

Cuando la bomba explota, el jugador no se volverá a instanciar en la siguiente ronda, la cual continuará con la misma mecánica hasta que solo quede un jugador.

4.2.5 Arquitectura de los vehículos

Internamente, todos los vehículos están compuestos por los siguientes tipos de game object hijos:

- **Colliders**: Compuesto en forma rectangular en vez de ser tipo de **mesh**, de este modo ahorramos en procesamiento y conseguiremos una respuesta menos realista y arcade ante las colisiones.

- **Wheel Collider**: Tipo de **collider** interno que emplea Unity para simular el comportamiento de las ruedas. Dispone de mucho tipo de parametrización para conseguir resultados más o menos realistas.

- **Partículas**: Que simulan el humo del coche.

- **Modelado**: Compuesto por el modelo 3D del vehículo y de las ruedas por separado.

- **Cámara**: Es la cámara que mostrará al vehículo cuando gane una ronda.

Desde el GameObject padre podemos acceder al script que gestionan y controlan los vehículos.

Tanto para los modelos controlables por la máquina como para los controlables por los jugadores, tenemos una clase en común que se llama Player Controller. Sirve para unificar parámetros de los vehículos ya que los scripts para el manejo de ambos tipos de vehículos son distintos.

En esta clase almacenaremos valores como el número de jugador, el nombre del modelo del coche, número de victorias, item disponible, si ha pasado por la zona segura en el modo Hot Potato o si es el portador de la bomba. También es el encargado de detectar si un vehículo ha volcado para voltearlo automáticamente unos segundos después.

En los vehículos **bots** hemos utilizado para su control las clases que vienen en la colección de assets de ejemplo de Unity, estas son:

- **Car Controller**: Implementa un sistema de control realista para vehículos.

- **Car AI Control**: Se complementa con Car Controller y permite el movimiento de vehículos hacia la posición Target que le indiquemos.

- **Car Audio**: Gestiona el sistema de sonido de los vehículos en función de la velocidad y los cambios de marcha que sean marcados desde Car Controller.

Por último, hemos implementado nuestro propio sistema de **inteligencia artificial** en un script llamado Tree Behavior al que veremos en su sección

correspondiente y que irá marcando a la clase Car AI Control hacia donde debe de dirigirse el vehículo entre otras funciones.

Los vehículos controlables por los jugadores humanos disponen de las siguientes clases para su control:

- **Wheel Vehicle**: Hereda de Car Controller, siendo modificado en el asset de saarg, Arcade car physics (https://assetstore.unity.com/packages/tools/physics/arcade-car-physics-119484) y que posteriormente hemos modificado nosotros para adecuarlo a nuestro proyecto, parametrizándolo e integrando el resto de funciones que hemos ido necesitando.

- **Engine Sound Manager**: Para la gestión del sonido de los vehículos.

4.2.6 Arquitectura del tutorial

El procedimiento de esta escena es sencillo, tan solo cuenta con una imagen que describe los controles predefinidos del juego tanto para gamepads como para teclado. También da la opción para reproducir un videotutorial en el que se explican los controles y las mecánicas básicas del juego.

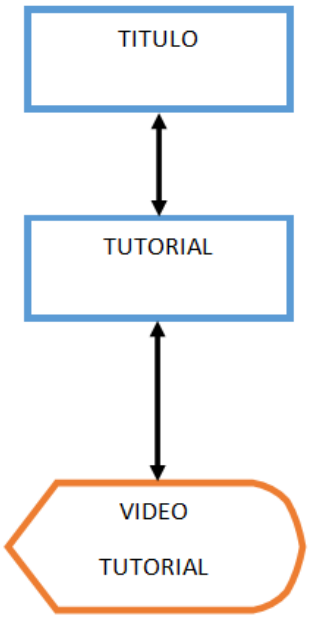


Ilustración 22 Diagrama del tutorial

4.2.7 Arquitectura de las opciones

Al igual que la arquitectura anterior, es de las más sencillas del programa. Consta de un menú en el que se le da al jugador la opción de elegir el número de rondas por nivel que quiera jugar (entre 1 y 10) y la dificultad del juego, el cual dispone de 3 opciones: fácil, normal y difícil.

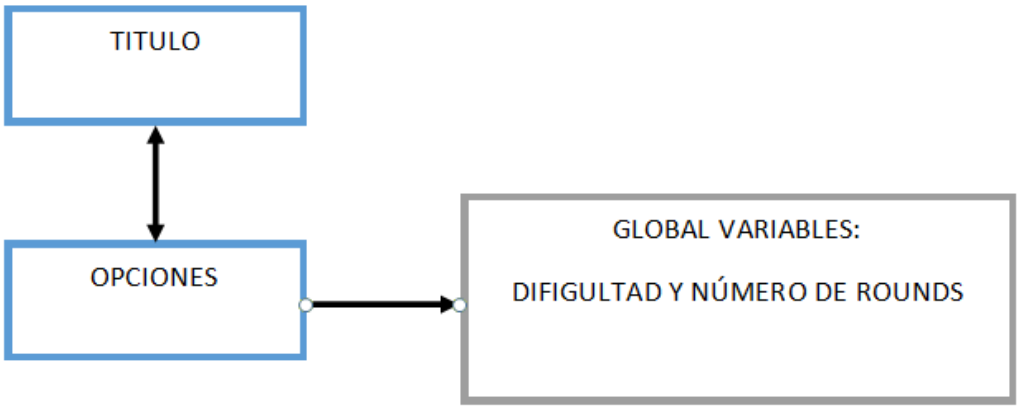


Ilustración 23 Diagrama de las opciones

Los cambios son almacenados en las variables de la clase no destruible entre escenas “Global Variables” para posteriormente ser procesados por el Game Manager (número de rondas para ganar en una fase) y por la IA del juego (nivel de dificultad)

4.3 Gráficos

En esta sección repasaremos el trabajo, la metodología, los diseños y las decisiones que hemos ido tomando sobre el apartado gráfico del juego.

4.3.1 Diseño conceptual

Uno de los objetivos de este trabajo era el diseñar la mayor cantidad de assets gráficos disponibles.

Desde un primer momento empezamos a plasmar conceptos e ideas con papel y lápiz para posteriormente, hacer uso de una tablet con pen para el finalizar los diseños y darles color.

Ilustración 24 Bocetos conceptuales

El logotipo del juego también pasó por varias fases de prototipado y pruebas con varios diseños entre los que quedaron 2 finalistas de los cuales, el último de ellos quedó como diseño final.

Ilustración 25 Diseño inicial del logotipo

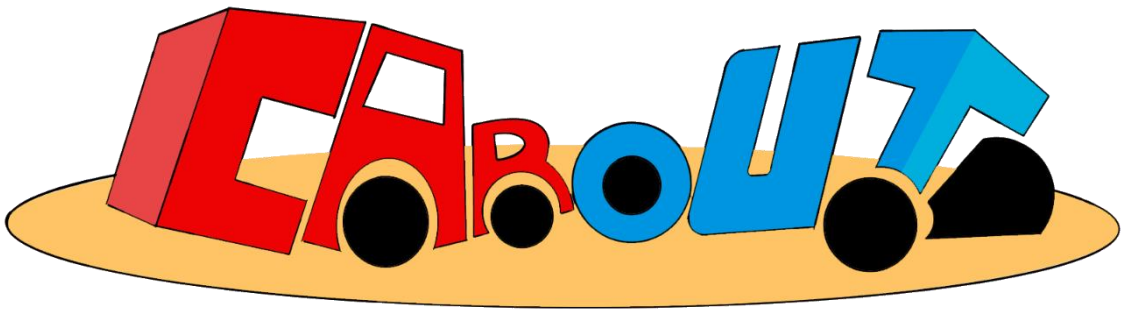


Ilustración 26 Diseño final del logotipo

En ambos se buscó plasmar la esencia del juego intentando ilustrar dos vehículos que chocaban entre sí para tratar de sacarse de una zona. La decisión principal de elegir la segunda ilustración es debido a que la primera nos parecía demasiado infantil y no iba a encajar con la estética del resto del juego.

4.3.2 Modelado 3D

Para los gráficos 3D teníamos claro en todo momento que queríamos modelados realizados con baja poligonización. Mas aún que cabe en los escenarios, pues los elementos aledaños al ring iban a ser poco vistos durante el transcurso de una partida. Con los vehículos trabajaríamos un poco más los detalles por ser los protagonistas del juego y por las cinemáticas que se reproducen cuando un jugador gana una ronda y se acerca la cámara a su vehículo tomándole un primer plano.

En una primera fase, empezamos a trabajar el modelo de los assets 3d con Maya, ya que disponíamos de la licencia de estudiante y quisimos aprovecharla.

Con Maya diseñamos algunos modelos que emplearíamos para la decoración de los escenarios. Sin embargo, el hecho de que en el futuro no íbamos a poder utilizarlo hizo plantearnos la posibilidad de utilizar una herramienta gratuita para tal fin. Después de estudiar las posibles alternativas, llegamos a la conclusión de que Blender iba a ser la más adecuada a nuestras necesidades.

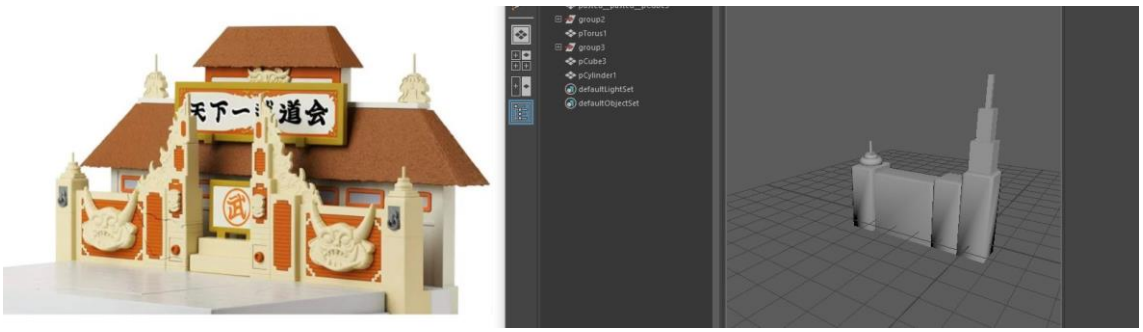


Ilustración 27 Ejemplo de modelado basado en un escenario

Ilustración 28 Modelado final del templo

En este trabajo no solo queríamos demostrar lo que habíamos aprendido con la asignatura, sino que también queríamos adquirir nuevas habilidades y aprender a usar Blender era una forma de hacerlo.

Con Blender modelamos principalmente los vehículos. Partimos de figuras básicas como base e inspirados en modelos reales, fuimos añadiendo formas, modificando y añadiendo nuevos vértices a estas para ir dándole las figuras redondeadas que el vehículo que teníamos en mente requería. Con esta metodología creamos los 3 primeros vehículos: Ryu, Terry y Goku.

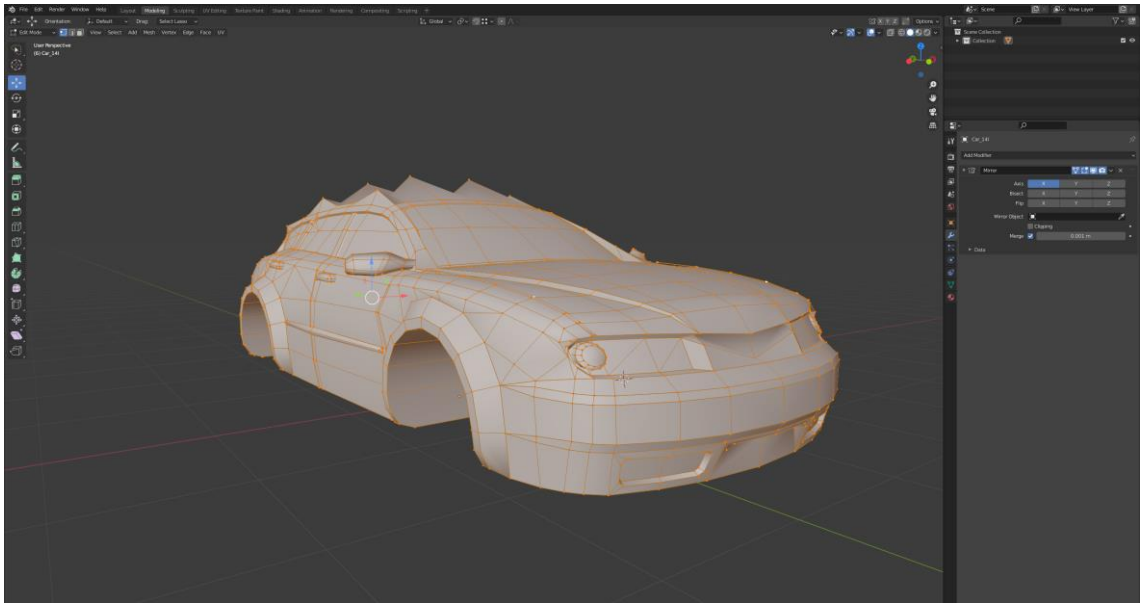


Ilustración 29 Edición del diseño de Goku en Blender

Este proceso nos estaba consumiendo mucho tiempo, sobre todo en el primer vehículo en el que no controlábamos bien la herramienta. Como el objetivo era tener 4 vehículos disponibles, optamos por utilizar un modelo gratuito que encontramos en turbosquid y aplicarle modificaciones para que no se pareciera tanto al modelo original. Así fue como se creó Kyo. Su asset original fue descargado de:

https://www.turbosquid.com/es/3d-models/3d-muscle-cougar-xr1970-blender-car-model-1573283

Una de las ventajas que nos encontramos al utilizar Blender fue su facilidad para crear los mapas UV los cual es la forma de añadir textura al modelo 3D a partir de un mapa. Además de dicha facilidad, el programa también te permite colorear las figuras directamente aplicandole los colores con un pincel al modelo 3D. Mediante esta técnica y el uso de Photoshop para ciertos detalles, aplicamos las texturas a los vehículos.

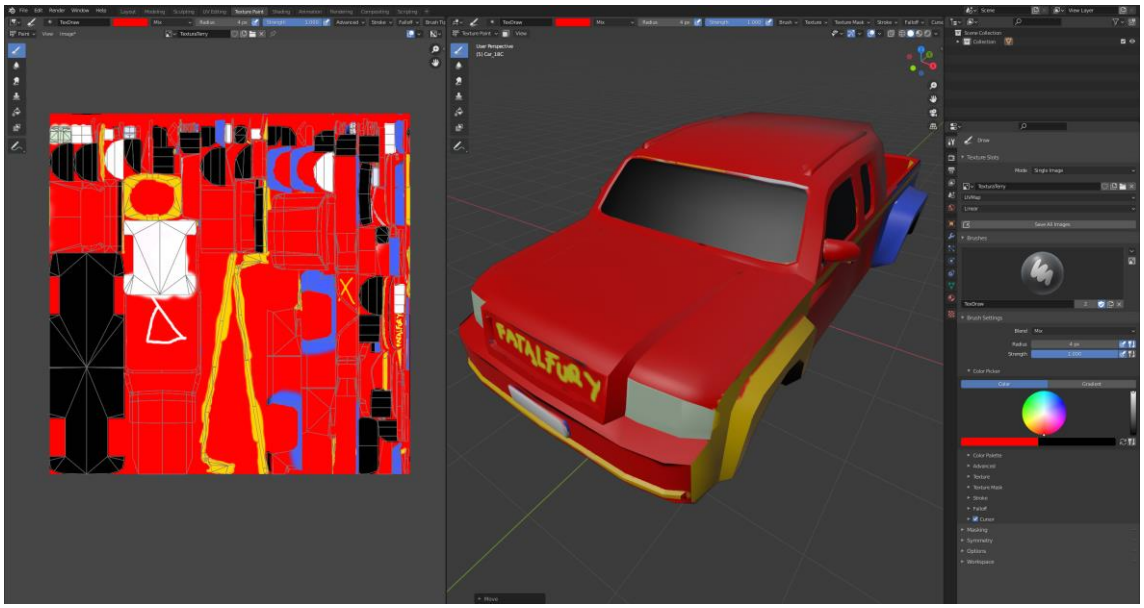


Ilustración 30 Mapa de UV y resultado del texturizado de Terry en Blender

El resto de coches 3D que empleamos vienen de assets como el “Arcade car physics” https://assetstore.unity.com/packages/tools/physics/arcade-car-physics-119484 , en el que usamos uno de sus vehículos como público, junto al que viene en las Standard Assets de Unity.

Otros assets tridimensionales externos que empleamos para los escenarios y que modificamos según nuestras necesidades fueron:

- Castillo asiático : https://www.turbosquid.com/es/3d-models/asia-asian-building-3d-model-1679121
- Palmera: https://www.turbosquid.com/es/3d-models/blender-carrot-crystal-oak-tree-3d-model-1189852
- Mesa de billar: https://free3d.com/es/modelo-3d/pool-table-v1--600461.html

El resto de figuras tridimensionales que se pueden encontrar en el juego corresponden a modelados que hemos realizado a partir de figuras básicas que ofrece Unity.

4.3.3 Grafismo 2D

Para los gráficos bidimensionales hemos utilizado varias técnicas:

Fotográficas

Hemos extraído texturas de fotografías para posteriormente aplicarlas a elementos como suelos o paredes.

Dibujado

Mediante el programa Photoshop hemos creado assets gráficos propios para la interfaz, los menús e ítems.

Retoque digital

Para la integración de elementos dentro del juego. Por ejemplo, la captura de imágenes de los vehículos para posteriormente recortarlo y guardar imágenes que emplearíamos para la pantalla de selección la interfaz.

Uso de imágenes con licencias gratuitas

Para agilizar la creación de escenarios, hemos empleado las siguientes imágenes:

- Textura de los edificios de la intro: https://besthqwallpapers.com/es/texturas/fondo-con-ventanas-pared-de-edificio-con-ventanas-ventanas-azules-textura-de-la-casa-fondo-de-la-fachada-de-la-casa-144147
- Imagen del skyline de New York para la intro: https://pixabay.com/es/photos/edificios-r%C3%ADo-ciudad-paisaje-urbano-668616/
- Textura del suelo del escenario Industrial: https://i.pinimg.com/originals/af/ab/23/afab231244a141a911fa45a24803ccd4.jpg
- Fondo del escenario de Industrial: https://images.pexels.com/photos/5206979/pexels-photo-5206979.jpeg?cs=srgb&dl=pexels-julia-volk-5206979.jpg&fm=jpg
- Textura de tejas para escenario Budokai: https://besthqwallpapers.com/Uploads/7-12-2019/114723/thumb2-orange-roof-tile-texture-4k-macro-old-roof-orange-wavy-background.jpg
- Textura de cemento: https://thumbs.dreamstime.com/b/textura-de-la-superficie-del-cemento-papel-pintado-concreto-gris-contexto-muro-viejo-fondo-blanco-vintage-natural-o-material-152392444.jpg
- Cuadro de perros jugando al billar: https://www.pinterest.es/pin/491877590528181706/
- Mesa de billar en forma de coche: https://thehappening.com/un-auto-como-mesa-de-billar/
- Fondo de montañas chinas: https://images.chinahighlights.com/allpicture/2017/09/838faa599d3a41269c046a26_cut_750x400_349.jpg

4.4 Sonidos

Al igual que con el apartado gráfico, con el audio del juego quisimos crear también nuestros propios assets, todo ello para experimentar los procesos creativos y adquirir nuevas habilidades, aun no teniendo conocimientos ni experiencia en la materia.
Para lograr nuestro objetivo, utilizamos la aplicación Garage Band con un IPad, la cual tuvimos que aprender a utilizar desde 0. Por suerte, resultó ser una herramienta muy intuitiva y con muchas posibilidades. Lo que nos permitió realizar trabajos bastantes aceptables teniendo en cuenta nuestra muy limitada capacidad como músicos.

Por motivos técnicos y económicos también tuvimos que hacer uso de recursos gratuitos para complementar este apartado.

4.4.1 Temas musicales

Podemos distinguir 2 tipos de temas musicales, las melodías inspiradas en otros temas y las originales.

Las melodías inspiradas hacen guiño a otras sin copiarlas, buscábamos solo que se parecieran a los originales del mismo modo que los vehículos hacen referencia a famosos luchadores de videojuego. Estas melodías son:

- **Intro**: Inspirada en la intro y la pantalla de título de Street Fighter 2.

- **Budokai**: Basada en la melodía que utilizaban en los resúmenes de los capítulos de Dragon Ball.

- **Victoria**: Fuertemente inspirada en la melodía que sonaba cuando se conseguía un nuevo récord mundial en el video juego Athlete Kings de Sega.

El resto de melodías que se pueden escuchar en el juego son de creación propias. Algunas fueron buscadas, para otras, simplemente nos dejamos llevar por la herramienta.

Podemos destacar los siguientes temas musicales en esta categoría:

- **Tema principal**: Se realizaron varias versiones con la melodía. Tras muchos probar con los instrumentos, decidimos usar 3 versiones.

    - **Título**: Mas roquera, dándole bastante presencia a la guitarra.

    - **Selección**: Experimentamos el uso de un **sample** que pronunciaba el título del juego para usarlo con las notas de la melodía.

    - **Opciones, tutorial y modo Hot Potato**: La melodía la realizamos mediante silbidos en un solo corte. Es más relajante que las 2 anteriores e ideal para el video tutorial y el menú de opciones del juego. Es también la única melodía que suena in-game y lo hace solo en el modo Hot Potato cumpliendo la función de temporizador de la bomba. El tema va a aumentando su velocidad de reproducción dinámicamente en función del tiempo que le quede a la bomba por explotar.

- **Presentaciones:** A excepción de Budokai, que nombramos anteriormente, el resto de melodías son propias y hemos pretendido que guarden relación con la ambientación del escenario.

4.4.2 Speaker

Nosotros mismos hemos grabado las locuciones del speaker del juego. Hemos empleado también el IPad y Garage Band, por la calidad de su micrófono y por el efecto de eco que les da a las grabaciones, lo cual se asemeja al efecto que se solía escuchar en las voces de los arcades.

El speaker narra lo siguiente dentro del juego:

- Inicio a las rondas: “Ready… Go!”
- Cuando un coche ha salido del escenario: “Car Out!”, para este evento, hemos realizado varias grabaciones con distintas entonaciones que se eligen aleatoriamente para darle mas variedad y dinámica a la situación.
- Cuando ganan una ronda: “Winner!”
- Ataques especiales de los coches:
    - Ryu: “Hadouken!”.
    - Terry: ”Power Geiser!”.
    - Goku: “Kamehame ha!”.
    - Kyo: “Kurai Yagare!”

4.4.3 FX

En este apartado entran los efectos especiales como el de las explosiones, los choques entre los coches y el sonido de la “traslación instantánea” que se reproducía en la serie de Dragon Ball y que el jugador puede lograr con el item salvavidas.

Todas las melodías son de uso libre y las hemos podido descargar de www.sonidosmp3gratis.com

Para las colisiones entre los vehículos, hemos utilizado varias grabaciones del mismo modo que hicimos con la narrativa de las salidas del ring.

Los efectos de sonidos correspondiente a los motores de los vehículos son los que venían con los assets a los que pertenecían. No se han modificado.

4.5 HUD

El **HUD** del juego tras estudiarlo detenidamente, debía presentarnos la siguiente información:

1. Número de jugador.
2. Imagen del vehículo que controla.
3. Barra de carga para obtener un ataque especial.
4. Indicación de que posee un ataque especial.
5. Indicación de que posee un item.
6. Número de rondas ganadas y las necesarias para ganar

Como podemos observar, son bastantes datos que además se multiplican por 4 al tener que mostrar cada uno de los jugadores.

En una primera aproximación, este fue el diseño inicial:

Ilustración 31 Diseño inicial del HUD

No nos resultó atractivo y ocupaba demasiada área dando mayor área de visionado a elementos que poco aportaban como la imagen del vehículo y poco espacio para elementos tan importantes como la barra de carga o el número de victoria.

Es por ello que decidimos rediseñarlo para obtener el siguiente diseño:

Ilustración 32 Diseño final del HUD

Ahora, aparte de ser más estético, ocupaba menos área, dejando más visión al mundo del juego mientras que mostraba la información importante a mayor tamaño. Además, cada jugador tendría un color predominante para ayudar a localizar su HUD dentro de la pantalla de juego. Por último, resolveríamos el hecho de que el jugador dispusiera de un ataque especial haciendo que la barra de recarga parpadeara cuando se llenase. Esta acción llamaría más la atención, a nivel visual, al jugador que el hecho de que se añadiese algún elemento gráfico estático al HUD.

En cuanto a la disposición, aunque queríamos seguir homenajeando a los juegos de lucha, los cuales, disponían de los HUD en una fila en la zona superior de la pantalla, decidimos rechazar esta primera idea a favor de colocar cada uno de los 4 HUD de los jugadores en las 4 esquinas de la pantalla.
Al tener el juego una cámara con perspectiva isométrica, conseguiríamos despejar gran parte de ring, tapando los HUD mayormente las zonas exteriores a estos. De este modo el jugador no vería tapada su área de acción por culpa de la interfaz del juego.

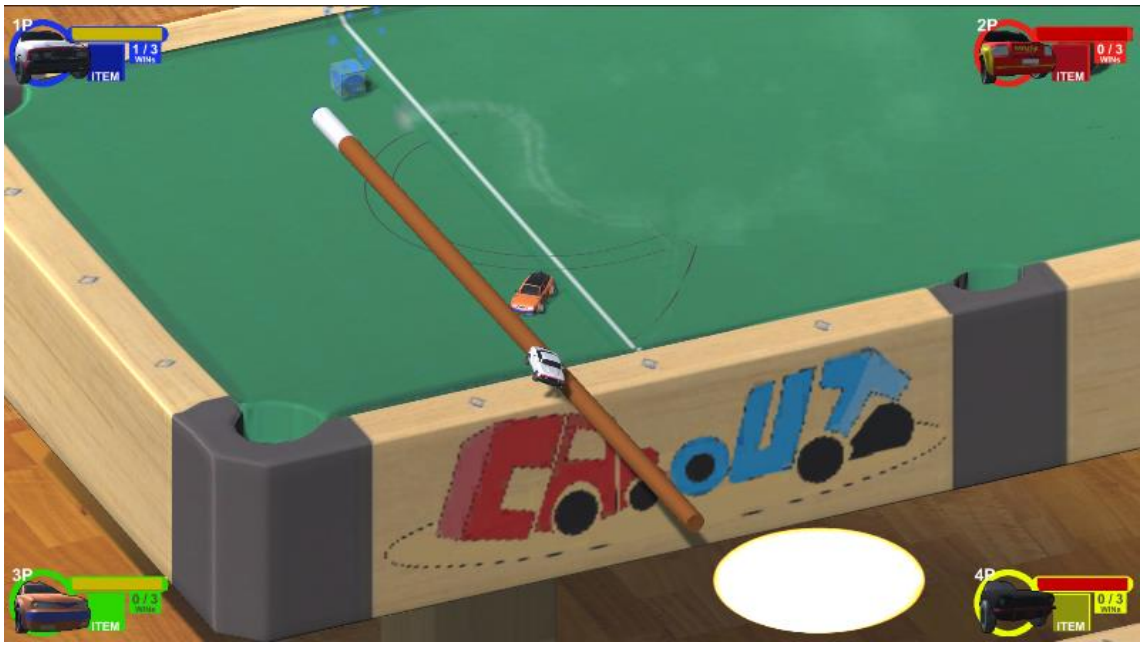


Ilustración 33 Captura del juego mostrando los HUD

4.6 INTELIGENCIA ARTIFICIAL

Aunque el juego está pensado para ser disfrutado de forma local en compañía, somos consciente de que hoy en día no resulta fácil juntar a 4 personas ni disponer de 4 mandos de control.
Es por ello que no queríamos sacrificar el juego en solitario e implementamos una IA para los jugadores restantes que no fueran humanos.

Hemos desarrollado dos modelos de prefabs por vehículo, siendo uno de ellos utilizado para los jugadores humanos y el otro para la máquina.

El sistema de IA está sustentado por la clase CarAIControl de los Standard Assets de Unity. Mediante esta clase se implementa un sistema de conducción empleando las funciones de la clase Car Controller, también pertenecientes al mismo conjunto de assets, para que el movimiento de los vehículos quede realista y parezca que un humano lo está haciendo. De este modo, nosotros tan solo tendríamos que proporcionarle a Car Ai Control un punto de destino donde queremos que vaya el vehículo, el cual denominamos Target.

En una primera versión nos dedicamos simplemente a colocar un gameobject como target y este iba apareciendo en zonas aleatorias del escenario. Esto sirvió para empezar a valorar el uso de este sistema. Aun con la sencillez con la que estaba planteado, muchas veces daba la sensación de que había un jugador humano a los mando. Sin embargo, era clara la tendencia a salirse del escenario los vehículos ya que, aunque tuvimos en cuenta las medidas del ring, si el target se generaba cerca de los límites, al vehículo no le daba tiempo a girar.
En una segunda versión de la IA básica, tuvimos este detalle en cuenta no permitiendo que se generara el target en un porcentaje cercano al límite.

Finalmente trabajamos en la versión definitiva de la IA utilizando para ello un simple pero efectivo árbol de decisiones:

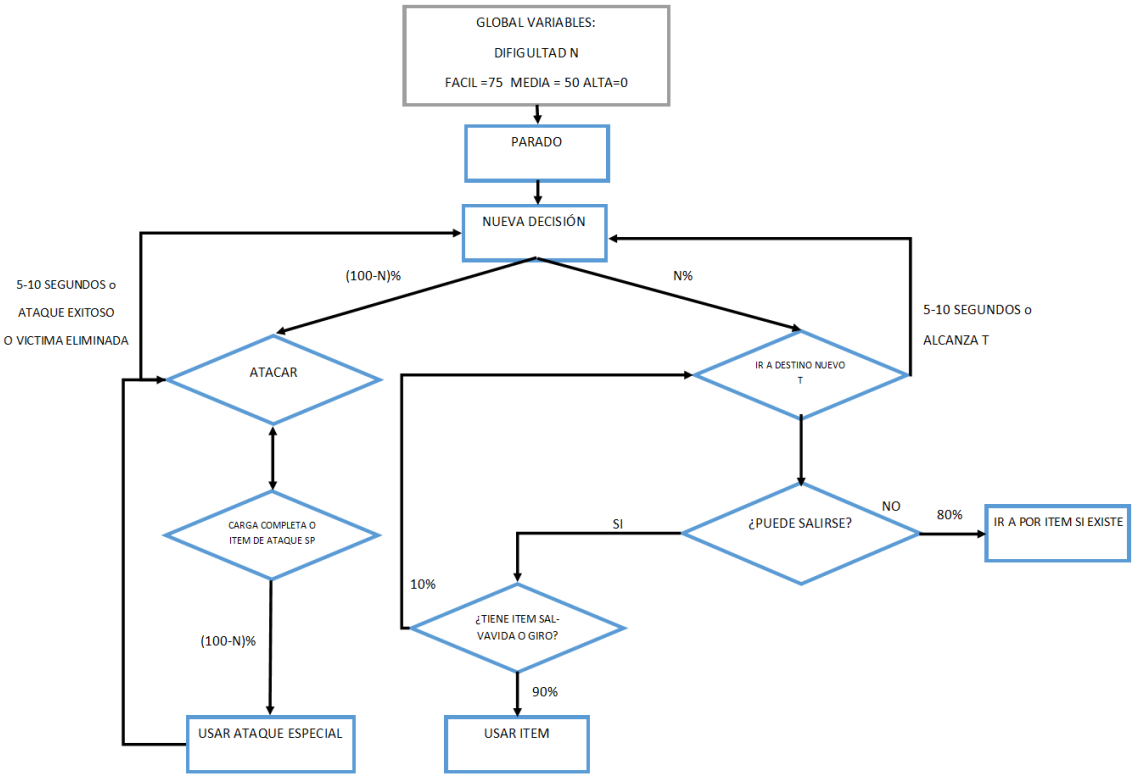


Ilustración 34 Diagrama de la Inteligencia Artificial

Las decisiones más importantes tienden a la probabilidad, según el nivel de dificulta escogido. A mayor dificultad, mayores serán las probabilidades de que los coches ataquen y usen sus ataques especiales.

Básicamente, la IA cambia de decisión entre circular o atacar cada X tiempo, siendo X un número de segundos al azar comprendido entre 5 y 10 segundos.

Si el coche está circulando, elige un punto del ring al azar, mientras se dirige a él y aparece un item en el escenario, tendrá un 80% de posibilidades de ir a por él.
Durante este estado, el coche comprueba si está a punto de salirse del ring, en caso afirmativo puede usar un item que le ponga a salvo en un 90% de las ocasiones, si no, tratará de poner rumbo al centro a la zona central del ring para salvaguardarse.
Cuando han pasado el tiempo prefijado o ha alcanzado su destino, toma una nueva decisión.

Cuando ataca, se fija en uno de los rivales que hay en el ring de forma aleatoria y procede. Si tiene la barra de recarga completa o el item de ataque especial, tendrá posibilidades variables de atacarlo con su especialidad en función del nivel de dificultad de la partida.
Si el jugador a impacto con la víctima, está ha desaparecido del ring o ha pasado el tiempo prefijado, vuelve a tomar una nueva decisión.

Para el modo Hot Potato, la IA sigue la misma lógica, tan solo que al inicio de la ronda, cuando hay que dirigirse hacia la “safe zone”, hay una probabilidad del 99% de que la máquina se dirija hacia este.

5. Diseño de niveles

Los niveles en CarOut! tienen como protagonista el ring. Es la zona donde se desarrollará gameplay del juego y toda gira a su alrededor. A esto hay que sumarle que el objetivo del juego es ser el último jugador en salir de él.

Para diferenciarlos aún más, los niveles cuentan con una serie de trampas que les darán más variedad a las partidas. Existe una en común para todos los niveles y es la bomba.
La bomba cae cuando un jugador lleva unos segundos sin moverse. No hace daño y su potencia es escasa, pero ayuda para desatascar situaciones en la que los jugadores se queden bloqueados tras una colisión o con algún elemento del circuito. Incluso puede decidir partidas si los jugadores se han quedado colgados cerca del abismo.

En nuestro proyecto hemos diseñado un total de 4 niveles para el modo World Championship y un nivel extra para el modo Hot Potato. Hemos procurado diferenciar lo máximo posible cada nivel para hacerlos más atractivos y divertido posible. Buscando la variedad para que el jugador tenga un Flow que no sea plano. Es obvio que a medida que los jugadores vayan conociendo el juego, habrá niveles que les gusten más que otro. A nuestro juicio es necesario para

mantener atento al jugador, ya que, si todos los niveles fueran iguales y no hubiese reto diferenciador, las partidas pueden llegar a aburrir.

5.1 Budokai

Es el escenario más sencillo. Está ambientado en el torneo de artes marciales que se celebraba en Dragon Ball. Su forma es la de un cuadrado.
Al ser también de los más pequeño, es fácil encontrarse con las hostilidades de los rivales

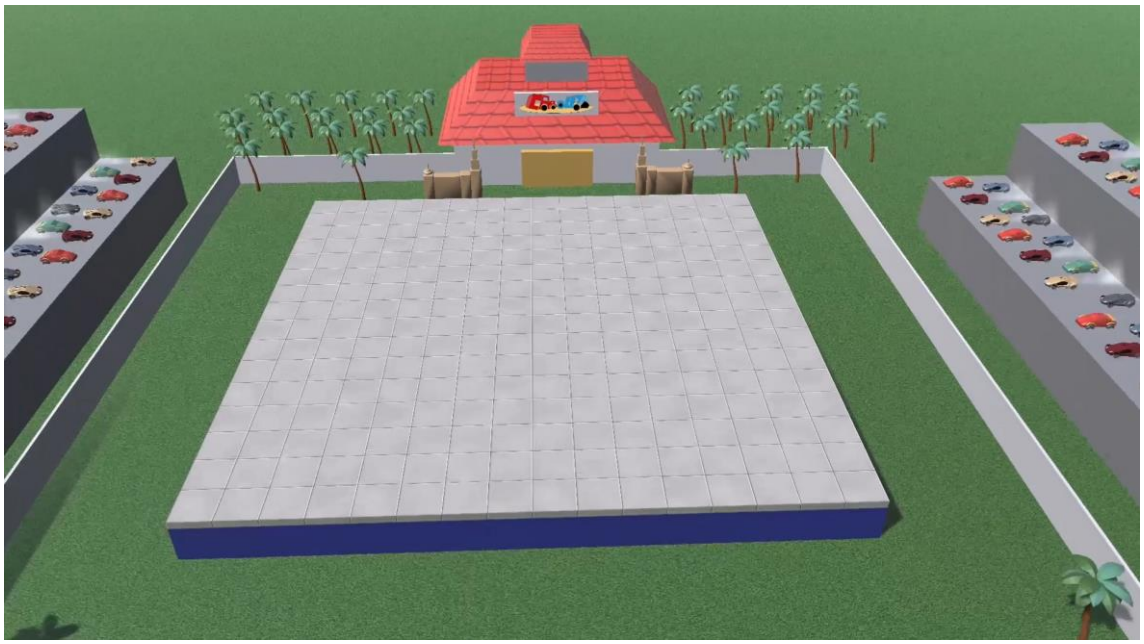


Ilustración 35 Vision general del escenario Budokai

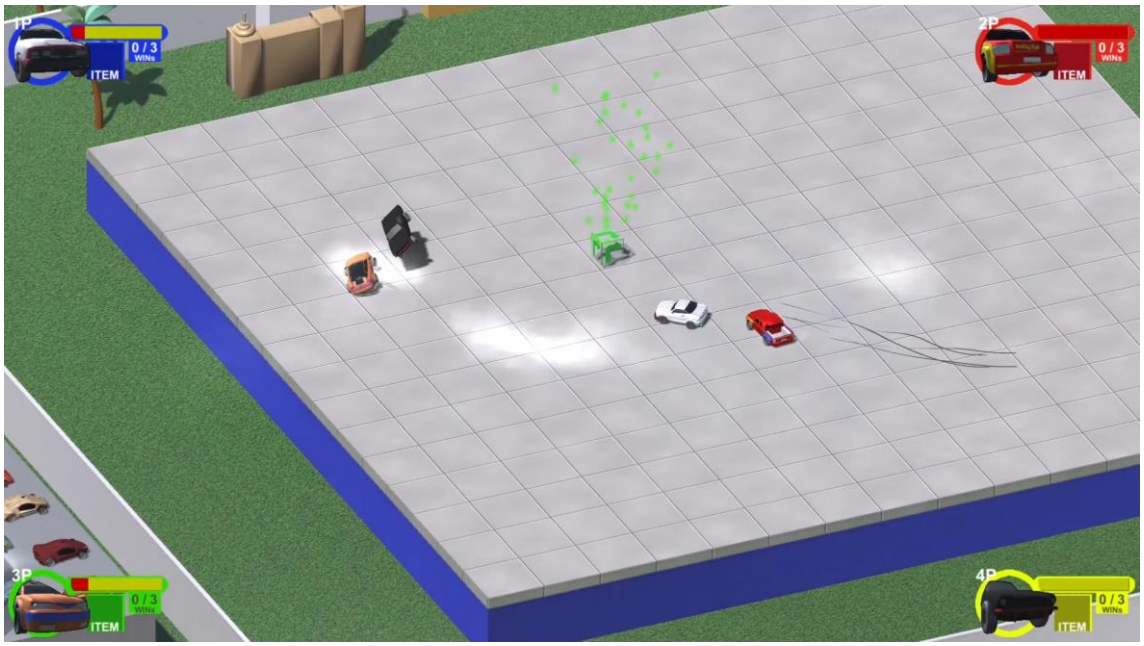


Ilustración 36 Captura del nivel Budokai

5.2 Billar

Como su propio nombre indica, se juega sobre la mesa de un billar, por tanto, su forma será rectangular. Se ha adaptado la estructura original de una mesa de billar para que su zona de juego se encuentre a la misma altura que los bordes, de este modo los jugadores pueden ser expulsados más fácilmente.
También tenemos que contar con los agujeros de la mesa, donde si un jugador cae, será automáticamente eliminado.
Pero si esto aun parecía poco, en Billar nos aparecerá aleatoriamente durante la partida, y a ambos lados de la mesa, un palo de billar que buscará hacer carambola con nosotros.

Ilustración 37 Visión general del nivel Billar

Ilustración 38 Captura del nivel Billar

5.3 Industrial

Dos son los aspectos diferenciales de este ring. El primero es que está perimetrado por una valla, por lo que, si el coche no es lanzado por encima de está, no saldría por el método convencional. Además, esta valla puede hacer rebotar a los coches si estos chocan contra ella con cierta velocidad.

El segundo, es que la valla tenía trampa, porque el suelo del ring se irá cayendo durante la ronda. Primero empezará a parpadear en rojo, irá aumentando la frecuencia del parpadeo y cuando se quede fijada en dicho color, caerá al vacío, dejando un hueco por el que los vehículos pueden salirse del escenario.

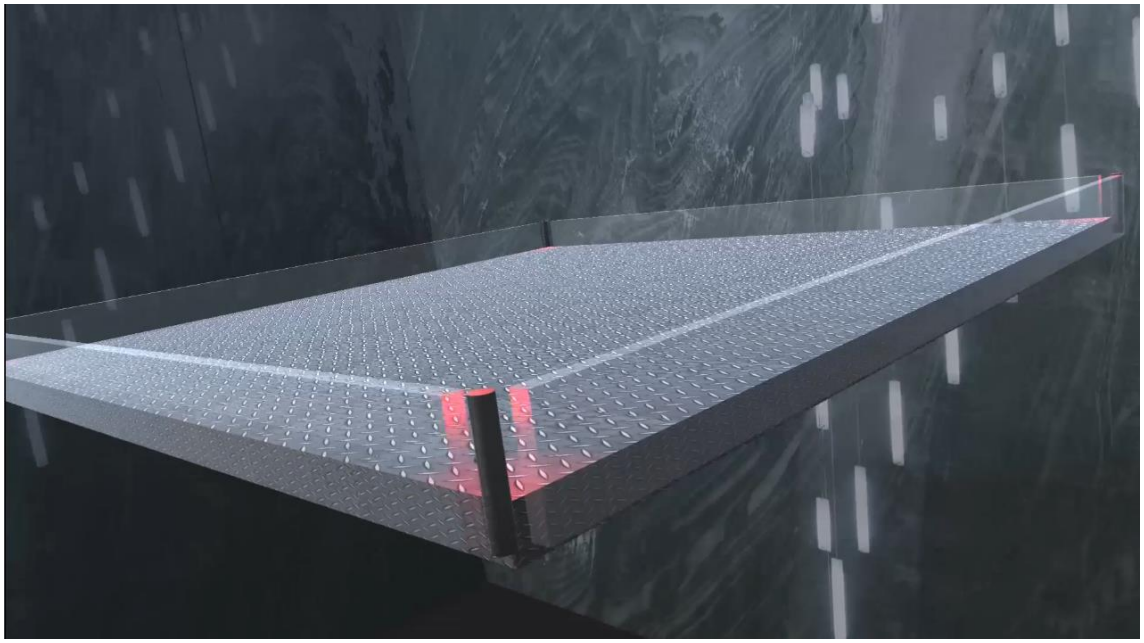


Ilustración 39 Vision general del nivel Industrial

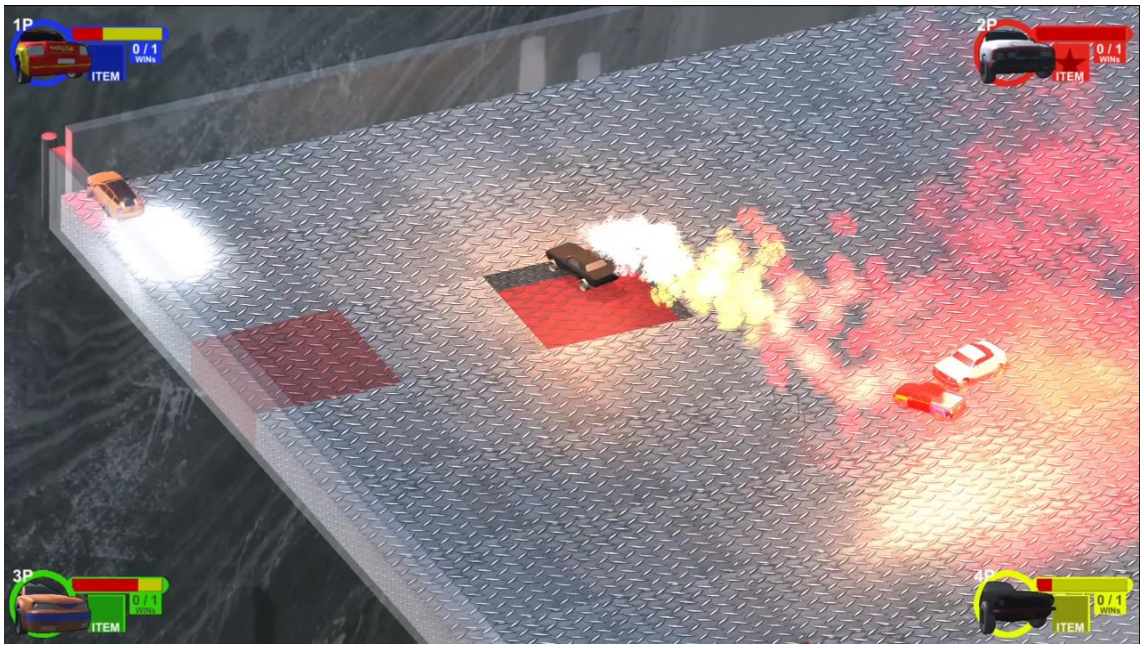


Ilustración 40 Captura del nivel Industrial

5.4 Yin Yang

El último escenario está ambientado en China y se juega sobre un símbolo de Yin Yang. Es el único ring del juego circular.
Tiene como trampas las circunferencias del símbolo que, a modo de ascensor, irán subiendo y bajando alternativamente. A veces será la salvación y otras la perdición.

Ilustración 41Visión general del nivel Yin Yang

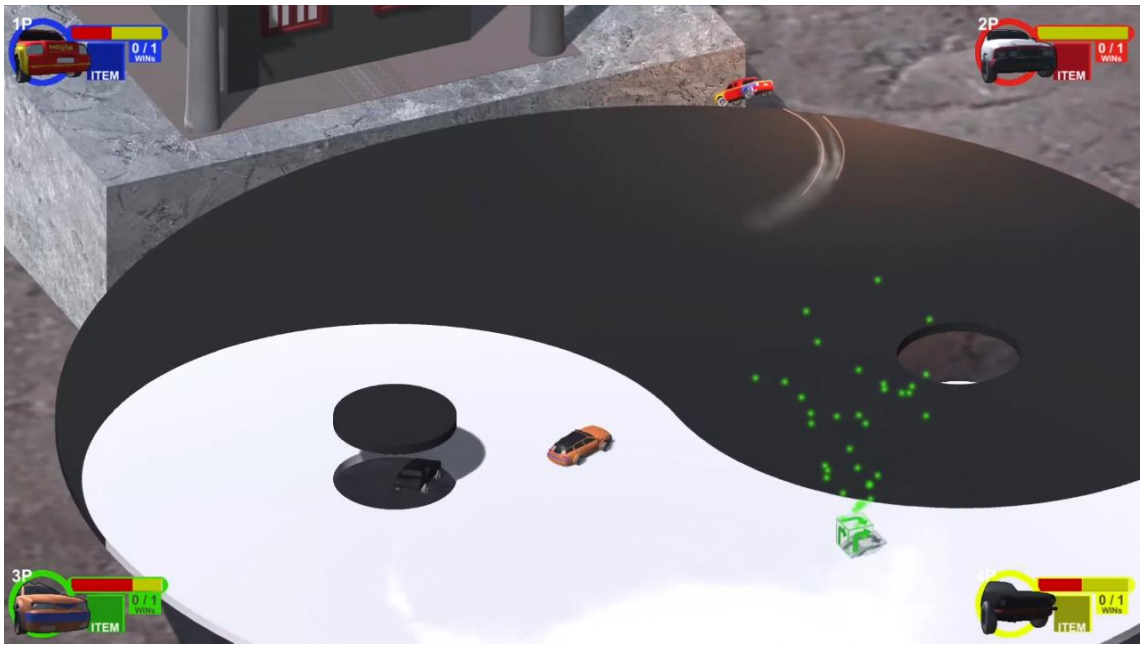


Ilustración 42 Captura del nivel Yin Yang

5.5 Hot Potato

Como ya comentamos anteriormente, es el único nivel del modo Hot Potato, debido a su peculiaridad (los coches solo pueden ser eliminados si les explota la bomba cuando la portan) cuenta con vayas perimetrales, como la que vimos en Industrial, pero en esta ocasión con mayor altura. Aun así, hemos controlado que si por algún motivo, un vehículo se saliese del ring, este volvería instantáneamente al centro de este. Su forma es rectangular.

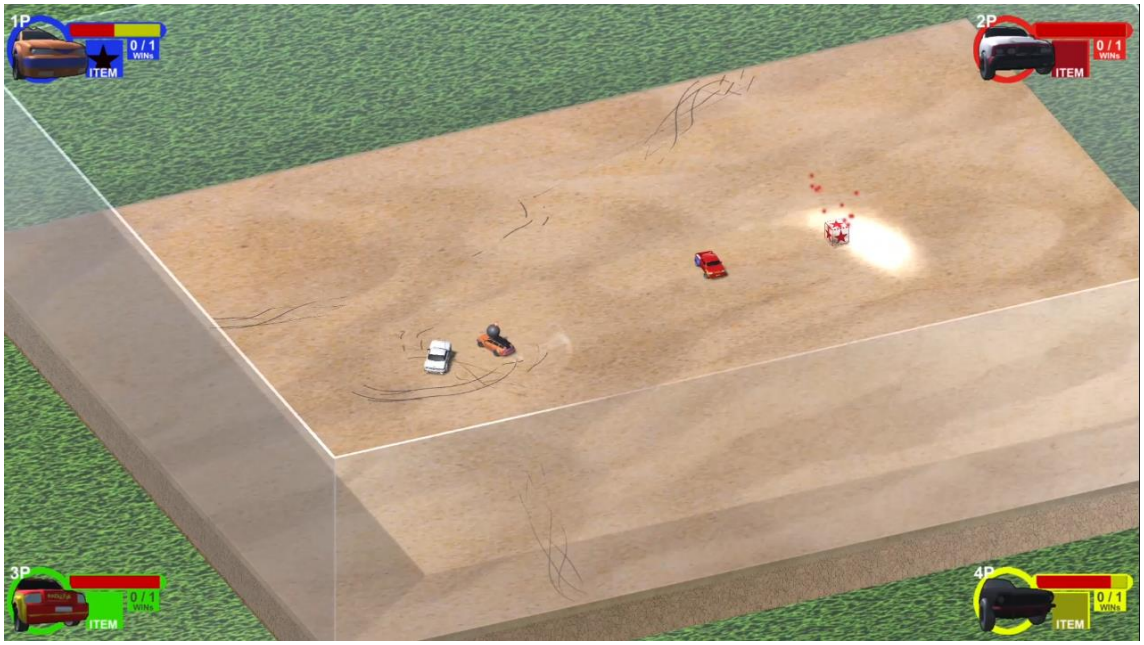


Ilustración 43 Captura del modo Hot Potato

6. Pruebas

Durante el desarrollo del juego, hemos tenido que realizar múltiples y elaboradas pruebas para comprobar que toda nueva implementación funcionaba bien y no entraba en conflicto con otras.

Nosotros hemos sido los principales testadores del juego. Somos quien más conocemos el producto y sabemos dónde puede fallar.

En la fase final y con el videojuego prácticamente finalizado, dimos a conocer el producto a familiares y amigos para que lo testeasen con nosotros con el fin de escuchar sus opiniones y consejos. De esta forma tomábamos nota de consejos e ideas y estudiábamos su viabilidad para realizar correcciones, añadidos o cambios.

El rango de edad de los jugadores de prueba ha ido desde los 6 años hasta los 39. Con perfiles que iban desde el jugador casual hasta personas no aficionadas.

En el siguiente enlace se puede acceder a un video integro de una de estas sesiones:

https://youtu.be/kpLOEe1ecYo

Estas pruebas sirvieron para realizar algunos ajustes al juego como:

- Cambios de algunas melodías
- Cambios en el menú de pausa para que fuera más difícil salirse del juego por error.
- Cambios en la navegación de los menús.
- Ajustes en el control.
- Ajustes de impacto de los ataques especiales para balancear el juego.

Las reacciones en cuanto al game feel del juego fueron bastante positivas, valorando positivamente el videojuego y concluyendo que era divertido y original.

7. Conclusiones

Tras el trabajo realizado, podemos llegar a una serie de conclusiones que expondremos en los siguientes apartados.

7.1 Lecciones aprendidas

- La principal lección que hemos aprendido es que el tiempo es uno de los recursos más valiosos, tanto para las actividades del día a día como para el trabajo y especialmente, para realizar tareas de consideración que tienen fecha límite de entrega.
- Hemos aprendido a valorar la validez del software gratuito. Estos pueden tener tanta o más funcionalidades que sus alternativas comerciales que encima, suelen ser las más conocidas.
- La planificación con vistas globales son claves para el desarrollo de trabajos de cierta envergadura. En varias ocasiones nos ha facilitado mucho la tarea el haber encontrado estructura de datos preparadas para funciones que nos iban a hacer falta en el futuro. Esto y mantener un control del código ha sido clave para poder continuar rápidamente con el proyecto después de haber estado bastantes días sin poder trabajar con él.
- El hecho de haber realizado tareas que van más allá del diseño y de la programación te restan tiempo que se podía haber dedicado a pulir detalles del gameplay o añadir nuevas **features**. Sin embargo, el adquirir este tipo de habilidades te da independencia a la hora de afrontar desarrollos en solitario y lo que es más importante, te da una visión global de los procesos de desarrollo y las necesidades de cada campo, lo cual, te habilita profesionalmente para puestos de gestión.
- A la hora de planificar hay que hacerlo con los pies en la tierra, ya que podemos convertir planificaciones en auténticas utopías. Es por ello que siempre hay que dejar márgenes en la planificación para corregir posibles desvíos en el planning.
- Hoy en día, está todo inventado o casi. Aun creyendo que teníamos una idea original, ya existían videojuegos que explotaban de cierta manera el concepto. Es por ello que creemos que nunca se debe de intentar clonar otro producto, mas aun cuando no se poseen los recursos suficientes y no se es capaz de mejorar considerablemente al original. Creemos que es mejor buscar la originalidad y la diferenciación en los productos de bajo presupuesto.

7.2 Objetivos y reflexión

El objetivo principal del trabajo era entregar una versión completa de un juego desarrollado por nosotros que demostrara todo lo aprendido durante el master en un solo semestre. Consideramos que el objetivo principal, por tanto, está cumplido.
Hemos podido desarrollar un videojuego de principio a fin, además, hemos cumplido también los objetivos automarcados por nosotros mismos. Estos eran el crear assets tanto gráficos como sonoros por nuestra cuenta.
Si bien nos hubiera gustado que el 100% de todos ellos fueran nuestros, tuvimos que recurrir a algunos recursos externos para complementarlos debido más a cuestiones temporales que a limitaciones técnicas.

Durante la primera fase del diseño del juego nos fijamos una serie de objetivos bastantes definidos. Tal como fuimos advertidos, solemos ser más optimistas de la cuenta y la mayoría de las veces auguramos una serie de features que generalmente suelen ser utópicas, más que en calidad, en cantidad.

Sabiendo esto, establecimos una tabla con los objetivos mínimos que debíamos alcanzar y los máximos:

	MINIMOS	MEDIOS	OPTIMOS
Nº de vehículos	4	6	8
Nº de circuitos	4	6	8
Nº Jugadores	Multijugador Local	<-+ BOTS (IA)	<- Multijugador online
Modos de Juegos	Campeonato	Arcade	Multiplayer y otros

Tal como podemos ver en la tabla, rescatada de la primera PEC, los objetivos mínimos del juego consistían en tener para la fecha de entrega del TFM 4 vehículos, 4 escenarios, y el modo de juego World Championship al que solo podían jugar jugadores humanos mediante multijugador local.

Podemos decir que hemos conseguido alcanzar un nivel de objetivos cercanos a los medios pues además de lo anterior, logramos implementar la IA para los jugadores y un modo extra de juego, Hot Potato.

En el tintero quedaron pues aspectos como el multijugador online, más modos de juego y el objetivo de 8 coches, 8 escenarios.

Hubo 2 motivos principales que excusan este logro:

1. Quisimos dar prioridad a la calidad antes que a la calidad y presentar un producto bien acabado. Por ello incluimos features que no entraron en la planificación inicial y que consideramos que le venían bien al juego. La escena de introducción, las presentaciones de los escenarios, el trabajo en la interfaz, tutorial, opciones de dificultad, la mecánica de la bomba, los continuos ajustes y testeos del gameplay son solo algunos ejemplos de

ellos. Gracias a esto hemos logrado transmitir el game feel que teníamos en nuestra mente cuando empezamos a idear el videojuego. Y es que el resultado final ha quedado casi exacto respecto al que imaginábamos al inicio del semestre.
2. El tiempo. Era nuestro bien más preciado. El resto de responsabilidades del día a día nos limitaban mucho y teníamos que hacer un buen uso de las horas de las que disponíamos.

7.3 Análisis de la planificación

Aunque nos intentamos ceñir a la planificación inicial, en un momento de la segunda fase tuvimos que trasladar tareas a la fase 3. Todo ello debido a la falta de tiempo.
Esto trajo una serie de desajustes que no afectaron a las entregas, pues gracias al empleo de una metodología ágil, pudimos adelantar contenidos básicos de la fase 3 a la 2.

Finalmente, debido al traslado de tareas de una fase a otra nos vimos en la fase 4 teniendo que implementar la mayoría de las tareas de la 3. Esta fase que estaba dedicada para la corrección y la optimización del juego resultó ser más laboriosa de lo esperado. También, fue esta última, la fase donde tuvimos mucho más tiempo disponible y, por suerte, pudimos finalizar las tareas que teníamos pendiente.

7.4 Futuras líneas de trabajo

Debido a la falta de tiempo, fueron algunas las características que no pudimos incorporar al videojuego. Esto abre una línea de trabajo a futuro para complementarlo y llegar a pensar en una posible comercialización

- **Llegar a los 8 vehículos**: En el planteamiento inicial, buscamos 8 posibles modelos de vehículos, cada uno caracterizado con un color y ataque especial único que lo distinguiese. En una fase intermedia del desarrollo pensamos en dejar como tarea final la posibilidad de que los jugadores pudiesen elegir clones de un mismo modelo. Se podría haber hecho relativamente fácil, bastando con cambiar la textura de los modelos o cambiarle el color a la materia que emplea el modelo tridimensional. Finalmente, no llegamos a tiempo de implementarlo.
- **Mas rings**: En un primer momento queríamos implementar 8, sin embargo, en juegos donde las partidas son cortas y por tanto, necesitan variedad para no caer en el aburrimiento, pensamos que lo ideal sería alcanzar una cifra cercana a la veintena.
- **Agregar más ítems** con funcionalidades diversas. Pero tampoco un mucho mayor número de ellos para no llevar a confusión al jugador.
- **Nuevos modos de juego**: Las mecánicas del juego permiten desarrollar nuevos modos con nuevas reglas de forma muy sencilla. Esto agregaría mucha variedad al juego. Algunos modos extras que teníamos pensados eran, por ejemplo, el rey de la colina, donde aparecerían zonas circulares

concéntricas en el escenario que darían puntos al jugador cuando estuvieran sobre ella (teniendo los otros vehículos que echarlos para ocupar su lugar y obtener ellos los puntos). Se puede incluso añadir modalidades de juegos muy clásicas como el “atrapa la bandera”, team match o battle royale.

- **Multijugador Online**: Durante el periodo de desarrollo del trabajo final del master estuvimos cursando también la asignatura de juegos multijugador. El momento en el que vimos como implementar juegos **WAN**, ya llevábamos el desarrollo demasiado avanzado y crear este modo hubiera consumido demasiados recursos.
- **Aumentar el número de opciones**: Para incluir ajustes gráficos y la posibilidad de mapear los controles
- **Optimización de recursos**: Para que el juego pudiese correr en el mayor número posible de equipos.
- **Mejoras en la IA**: Para que la máquina pudiese tomar decisiones mas avanzadas, como dar prioridad a atacar al jugador que esté en mejor situación en el ranking o detectar las trampas.

8. Glosario

Arcade: Genero de videojuegos donde las partidas son rápidas y las físicas poco realistas.

Streamer: Persona que retransmite contenidos.

Gamer: Aficionado a los videojuegos.

Esports: Videojuegos competitivos multijugador.

Indies: Videojuegos realizados con menos recursos económicos que la media de las producciones y con un pequeño grupo de trabajo.

AAA: Videojuego de gran presupuesto.

Assets: Recursos que empleamos en un videojuego, pueden ser gráficos, sonoros, código…

Scrum: Metodología de desarrollo ágil.

Item: Recurso aprovechable en un videojuego que nos puede aportar ventajas.
Sprite: Recurso gráfico utilizado en un videojuego.

Gamepad: Periférico enfocado al control de videojuegos.

Lore: trasfondo de un juego es el conjunto de historias, personajes etc que conforman el universo representado en el mismo y le dan coherencia

VS: versus, genero de videojuegos centrado en la lucha uno contra uno

Intro: Escena inicial que presenta un videojuego.

Rooster: Reparto de entidades jugables.

IDE: Entorno de desarrollo integrado.

UV: Asignación en un mapa de 2D a los vértices de una figura tridimensional.

Game Object: Objeto en Unity que puede contener a otros objetos con uno varios componentes.

Prefab: Conjunto de game object que representan una identidad y que pueden ser instanciados en el juego.

IA: Inteligencia Artificial.

Spawn. Punto de reaparición de un jugador. Tanto al inicio de la partida, como después de morir.

Collider: Estructura necesaria en Unity para detectar colisiones.

Mesh: Rejilla que compone una estructura tridimensional

Sample: Sonido pregrabado para realizar melodías.

Feature: Característica diferenciada de un videojuego.

9. Bibliografía

- [1] Primer videojuego - Wikipedia, La enciclopedia libre, 24 de junio de 2012 https://es.wikipedia.org/wiki/Primer_videojuego (23/5/21)

- [2] Welcome to PONG_Story – The site of the first video game – David Winter, 1996 http://www.pong-story.com/intro.htm (23/5/21)

- [3] Scrum Methodology definition – Diego Calvo, 7 de abril de 2018 https://www.daiegocalvo.es/en/methodology-scrum-methodology-agil/scrum-methodology/ (25/5/20)

- [4] Car Fight –IO https://www.fandejuegos.com/juego/car-fight (20/2/21)

- [5] Div Games Studio – Wikipedia, La enciclopedia libre, 3 de octubre de 2020 https://es.wikipedia.org/wiki/DIV_Games_Studio (31/5/21)

10. Anexos

Manual de usuario

Requerimientos técnicos del hardware

Los requisitos mínimos testeados son:

- Procesador i5 4ta generación.
- Gráfica integrada.
- 4 GB de RAM.
- 400 MB de espacio en disco.
- Direct X 10

Los requisitos máximos testeados son:

- Procesador Ryzen 1800x
- Nvidia 980ti 6GB RAM
- 16 GB RAM DDR4
- 400 MB de espacio en SSD
- Direct X 12

Si bien, el juego puede ser disfrutado por hasta 2 jugadores con un solo teclado. Se recomienda el uso de controles estandarizados con XBOX. Estos son los mandos que toma como base la plataforma Windows. El mapeado de los controles está realizado pensando en dichos periféricos.
Con 4 controles de este tipo pueden jugar otros tantos jugadores.

Instrucciones del juego

Para inicializar el juego hay que hacer doble click en su correspondiente icono:

Ilustración 44 Icono del juego

Una vez inicializado podemos ver la introducción del juego, la cual nos podemos saltar pulsando cualquier tecla.

Lo siguiente en aparecernos es el título del juego junto a un mensaje parpadeante que nos indica que pulsemos una tecla si queremos comenzar.

Ilustración 45 Titulo con opciones

Podemos movernos por el menú con las flechas verticales del teclado y acceder a sus acciones con la tecla intro. Tambien podemos usar los mandos de control tal como aparecen por pantalla.

El juego dispone de 2 modos de juegos, un tutorial y las opciones.

Opciones

Podemos variar 2 parámetros del juego, el número de rondas necesarias para ganar (1 a 10) y el nivel de dificultad (fácil, medio y difícil).

Para modificar dichos parámetros nos colocaremos sobre ellos desplazándonos verticalmente y mediante las flechas o joystick horizontales, modificaremos sus valores.

Una vez configurado al gusto, confirmaremos con “Save and Return”

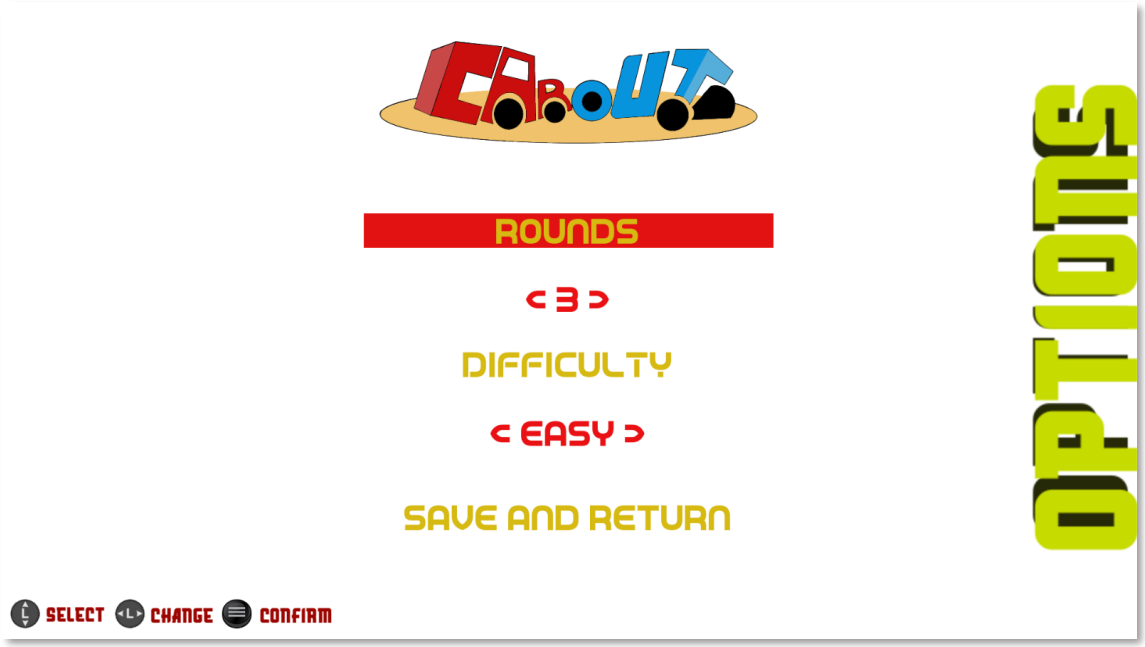
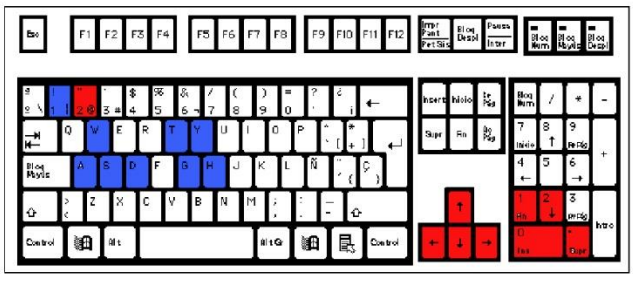


Ilustración 46 Opciones del juego.

Tutorial y controles

En la pantalla de tutorial se muestran los controles del juego y la posibilidad de ver un video de menos de 2 minutos explicándolos junto a las mecánicas del juego.
Los controles de CarOut! son los siguientes:



ACCION	JUGADOR 1	JUGADOR 2
Girar a la izquierda	A	←
Girar a la derecha	D	→
Acelerar	W	↑
Frenar	S	↓
Marcha delantera	G	0 Bloq Num
Marcha trasera	H	Supr. Bloq Num
Freno de Mano	G	0 Bloq Num
Ataque especial	T	1 Bloq Num
Usar Item	Y	2 Bloq Num
Seleccionar / pausa	1	2

Ilustración 47 Controles del juego.

Durante el juego, si pulsamos los botones de Start (1 o 2 en teclado y el icono de “play” o las 3 líneas paralelas en el mando (depende si el control corresponde a Xbox 360 o One)). El juego se pausará mostrando las instrucciones para reanudar la partida o salir de esta.

Selección de vehículos

Para seleccionar un vehículo, colocaremos el recuadro con nuestro número de jugador y color encima del coche deseado mediante los controles horizontales.

Presionaremos el botón A (Drift – Freno de mano) en los gamepad o el equivalente si estamos jugando con teclado (G para jugador 1 y el 0 del bloque numérico para el jugador 2).

Solo se puede coger un modelo de coche por jugador. En caso de que hayamos elegido uno no deseado, podemos revertir nuestra acción siempre y cuando los demás jugadores no hayan confirmado aún sus vehículos. Esta acción la haríamos con el botón de start.

Ilustración 48 Pantalla de selección de vehiculos

Modo World Championship

Consta de 4 fases en la que debemos de ser los últimos jugadores en quedar sobre el ring. Cada fase cuenta con n rondas que de forma estándar son 3. Cuando un jugador gana una ronda se le suma un punto. Si gana las rondas necesarias de la fase, se le darán 3 puntos extras.

Entre fases se muestra un ranking con la puntuación actual.

Cuando se finalizan las 4 fases veremos la escena final donde se muestra quien ha sido el ganador de la partida. En caso de empate, se vuelve a jugar la última fase para desempatar.

Modo Hot Potato

La dinámica de la partida sucede de la siguiente forma:

1. El jugador debe de dirigirse y tocar la “Safe Zone”. El que no lo haga antes de los demás, será el portador de la bomba.

2. Hay un tiempo límite en el que explotará la bomba, la única pista que hay es mediante el aumento de la velocidad de reproducción de la melodía.

3. El jugador portador de la bomba debe de pasársela a otro jugador. Para ello deberá chocar con él para pasársela.

4. El resto de jugadores deben de huir del portador.

5. Cuando la bomba explote el jugador será eliminado volviendo al punto 1 hasta que solo quede un vehículo en el ring.

TRAILER

En el siguiente enlace se puede acceder al trailer de presentación que hemos realizado para el videojuego:

https://www.youtube.com/watch?v=CEONEBxWn5o