Auto Tele – videojuego de conducción controlado mediante seguimiento ocular

Juan José Caballero Novella
Grado en Ingeniería Informática
75.640 – Videojuegos

Gustau Marcos Ballester
Joan Arnedo Moreno

Diciembre 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Auto Tele – videojuego de conducción controlado mediante seguimiento ocular.*
Nombre del autor:	*Juan José Caballero Novella*
Nombre del consultor/a:	*Gustau Marcos Ballester*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	12/2021
Titulación:	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*Trabajo Fin de Grado*
Idioma del trabajo:	*Castellano*
Palabras clave	*Unity, simulación, control visual*

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

Los videojuegos no son únicamente una forma de entretenimiento. También pueden suponer una forma de autorrealización, de evasión momentánea y de superación de los propios límites, aspectos que pueden ayudar en la calidad de vida de muchas personas. En este sentido, las experiencias jugables comerciales no siempre están adaptadas a situaciones de movilidad reducida o diversidad funcional y existen muy pocas soluciones creadas específicamente con esta realidad en mente.

El presente proyecto tiene por objetivo crear un videojuego de conducción que pueda controlarse única y exclusivamente mediante uno de los periféricos más accesibles que existen: los dispositivos de seguimiento ocular, que no necesitan nada más que el movimiento de los ojos para interactuar con los componentes jugables que lo compongan.

La experiencia propuesta, denominada Auto Tele, se restringe a los dispositivos de la marca Tobii por motivos de eficiencia de recursos y de tiempo, pero demuestra que es posible adaptar géneros tradicionales de videojuegos en principio complejos a sistemas que puedan ser disfrutados por sectores de población a menudo excluidos de un modo de ocio cada vez más relevante en la sociedad.

Abstract (in English, 250 words or less):

Video games are not just a mere form of leisure. They can also be a form of self-realization, escapism and overcoming your own limits, aspects that can help in the quality of life of many people. In this sense, commercial playable experiences are not always adapted to situations of reduced mobility or functional diversity and there are very few solutions created specifically with this reality in mind.

This project aims to create a racing video game that can be controlled solely and exclusively by one of the most accessible peripherals that exist: eye-tracking devices, which do not need anything more than the movement of the eyes to interact with the playable components in it.

The proposed experience, called Auto Tele, is restricted to Tobii brand devices for reasons of resource and time efficiency, but shows that it is possible to adapt traditional, supposedly complex, genres of video games into systems that can be enjoyed by sectors of the population often excluded from a way of entertainment which is increasingly relevant in society.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Los videojuegos son un mercado en continuo crecimiento; solamente en España supusieron durante 2020 una facturación de aproximadamente 1.747 millones de euros [1] y cada vez son más las personas que quieren sumarse a disfrutar de ellos. Tradicionalmente, ese tipo de entretenimiento digital se ha llevado a cabo de un modo normativo, siguiendo una filosofía de diseño y unos esquemas de control que no tienen en cuenta las necesidades especiales de personas con limitaciones físicas debido a accidentes y a condicionantes como diversidad funcional o parálisis cerebral.

Para personas que no tienen la capacidad de desenvolverse libremente en su día a día, ponerse a los mandos de un videojuego puede suponer una forma de expansión y de autorrealización que les ayude a afrontar otro tipo de terapias o contribuya a metas de superación personal. Aunque cada vez son más los títulos que incorporan opciones para mejorar la accesibilidad y se construyen periféricos adaptados, no existe un mercado dedicado a desarrollar desde cero juegos específicos que implementen géneros tradicionales (plataformas, conducción, aventura…) a estos colectivos.

El proyecto entre manos busca demostrar que es posible implementar un juego de un género clásico, como es la conducción, que se controle únicamente mediante dispositivos de seguimiento ocular.

1.2 Objetivos del Trabajo

Las metas generales a alcanzar son:

1) Desarrollar un videojuego con manejo exclusivamente mediante reconocimiento ocular, sin intervención de otro tipo de controles.
2) Disponer de varios modos de juego que ofrezcan una experiencia variada.
3) Utilizar dispositivos de seguimiento ocular del tipo más sencillo posible que únicamente sigan el seguimiento de los ojos, sin otro tipo de funcionalidades avanzadas como funciones de clic o simulación de ratón, para apostar por una mayor inclusividad.
4) Demostrar mediante los tres puntos anteriores que pueden crearse todo tipo de videojuegos con la accesibilidad como principio.

Los objetivos se abordarán en más detalle en el capítulo 3.3, dedicado al desarrollo y al *Roadmap*.

1.3 Enfoque y método seguido

La estrategia inicial del proyecto planteaba el juego como una carrera en la que el vehículo controlado por el jugador se desplazaba automáticamente a lo largo de un circuito, limitándose la jugabilidad a la elección entre una serie de carriles o pistas con distintas características: velocidades variables, grados de riesgo / recompensa, ubicación de obstáculos, presencia de objetos potenciadores, etc. Aunque este enfoque permitía tener bajo control las variables relativas a los inputs del jugador y simplificaba el diseño general, pronto quedó patente que conllevaba un sobreesfuerzo adicional al tener que configurar manualmente las guías de movimiento del vehículo, reajustar la posición de otros elementos cada vez que se modificaban éstas y suponía una experiencia de juego muy limitada.

Tras realizar pruebas y ajustes en los esquemas de control, especialmente en los componentes relacionados con las ruedas, se adoptó una solución intermedia entre un movimiento completamente libre y el prefijado inicialmente propuesto. Este nuevo sistema deja en manos de la lógica del juego la velocidad y aceleración del vehículo, pero proporciona al jugador un control total sobre el eje delantero que determina el encaramiento del vehículo; aunque obliga a introducir métodos de contención adicional para acotar los límite del escenario a los que puede accederse (como muros de delimitación del circuito transitable) supone una experiencia más próxima al de los títulos tradicionales del género y amplía el potencial de los modos de juego que pueden crearse.

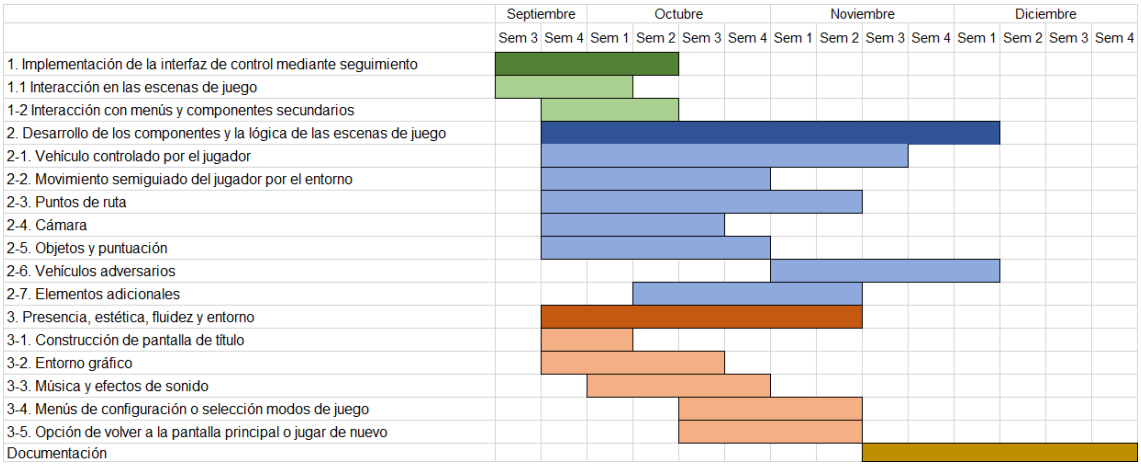
1.4 Planificación del Trabajo

Los requisitos necesarios e imprescindibles para el desarrollo del proyecto son:

- Motor de desarrollo de videojuegos Unity.
- Dispositivo de seguimiento ocular Tobii.
- IDE para programación en C# (Visual Studio o equivalente).
- Acceso a internet para descarga de recursos desde Unity Assets Store

En el capítulo 4.1, dedicado al entorno de desarrollo, puede encontrarse el listado completo de útiles necesarios y las especificaciones técnicas que aplican.

En lo que respecta a las tareas, se replica el diagrama de Gantt con la planificación de objetivos que aparece en el capítulo 3.3.1:



1.5 Breve sumario de productos obtenidos

El resultado del presente trabajo ha servido para obtener un videojuego de conducción que se controla única y exclusivamente mediante tecnología de seguimiento ocular. Una vez iniciada la aplicación mediante el archivo ejecutable proporcionado, se interactúa con todos los controles tanto de selección de opciones como relativos a la jugabilidad mediante un elemento hardware de la gama Tobii.

1.6 Breve descripción de los otros capítulos de la memoria

1) Introducción: Resumen de los conceptos generales, que se tratarán individualmente con más detalle en el resto del documento.
2) Estado del arte: Historia de las bases teóricas que fundamentan el proyecto y estudio del contexto tecnológico precedente.
3) Definición del juego: Aspectos del desarrollo y la planificación no pertenecientes a la fase de construcción y programación.
4) Diseño técnico: Descripción de los componentes que conforman la solución y los mecanismos empleados para su cohesión.
5) Diseño de niveles: Recorrido por los escenarios de interacción ofrecidos al jugador.
6) Análisis de costes: Estimación de la inversión de tiempo y recursos, tanto económicos como energéticos, que han sido necesarios para materializar la solución.
7) Conclusiones: Discusión final sobre la solución y perspectivas sobre su posible futuro.
8) Glosario: Relación de términos específicos relacionados con el proyecto.
9) Bibliografía: Recopilación de documentación consultada durante el proyecto.
10) Anexos: Contiene la acreditación de recursos externos incorporados a la solución y el manual de usuario.

2. Estado del arte

2.1 Historia de la tecnología de seguimiento ocular

El seguimiento ocular o *eye-tracking* consiste en el análisis y evaluación tanto del movimiento de los ojos como del punto de fijación de la mirada con respecto a una superficie fija, registrando los momentos en que la mirada permanece más tiempo en una posición determinada o la rapidez en que los ojos efectúan un desplazamiento.

Los primeros estudios sobre el comportamiento de los movimientos oculares y visuales se atribuyen al oftalmólogo Louis Émile Javal a finales de 1870. Como consecuencia de una investigación sobre la fisiología de la lectura, Javal constató que durante la misma los ojos no se movían continuamente a lo largo de una línea de texto (como se suponía anteriormente) sino que se trataba de un proceso no lineal, lo que significa que los ojos no se mueven continua e ininterrumpidamente durante el paso por cada línea, realizando en su lugar movimientos cortos y rápidos entremezclados con pausas o paradas breves sobre elementos concretos [2]. A partir de entonces, surgieron varios dispositivos relacionados con esta área de conocimiento, como la cámara de grabación de movimiento ocular ideada por Charles H. Judd en la primera década de 1900, el oftalmógrafo y el metronoscopio creados por Earl, James y Carl Taylor en 1933 [3] o el Reading Eye II desarrollado por Stanford Taylor en 1971, que registraba los movimientos oculares a través de sensores infrarrojos traspasándolos a papel sensible al calor [4].

En relación con el mundo de la informática y la computación, el primer hito destacable aparece en 1985 con el Visagraph I de Stanford Taylor; tomando las lecturas de emisores infrarrojos que eran transferidas a un Macintosh, este sistema evaluaba tanto la eficiencia de lectura como la competencia visual / funcional a través de un análisis automatizado por ordenador. Aun cuando el avance de la tecnología ha ido permitiendo una mayor reducción, sofisticación y perfeccionamiento del hardware empleado, la metodología de los dispositivos de seguimiento ocular disponibles actualmente sigue basándose en el mismo principio: un rastreador ocular envía luz infrarroja cercana que se refleja en los ojos, siendo estos reflejos captados de vuelta por cámaras en el rastreador. Mediante filtrado y cálculos realizados por software, se determina dónde está situada la mirada de la persona frente al dispositivo [5].

Figura 1 - Visagraph I

Las aplicaciones de este tipo de tecnología abarcan múltiples campos, como puede ser la detección de puntos de interés máximo y mapas de calor en publicidad y márketing [6], apoyo a diagnósticos médicos [7] o terapias ocupacionales y rehabilitación [8].

2.2 Videojuegos y accesibilidad

Los videojuegos, ya en su planteamiento, presentan una combinación de características que no siempre están al alcance de todo el mundo; podríamos definirlos como un modo de entretenimiento audiovisual expuesto a través de una pantalla en la que la persona o personas que juegan tienen cierto control sobre los eventos que suceden mediante un periférico o accesorio. Debido a ciertos condicionantes físicos, discapacidades o diversidades funcionales, la mencionada combinación deja fuera a potenciales jugadores que no pueden disfrutar la experiencia lúdica completa o incluso quedan totalmente excluidos.

En sus orígenes los videojuegos eran un mercado muy reducido y esta problemática no formaba parte de las prioridades del sector, pero con más de 2.600 millones de personas participando en alguna forma de ocio digital en la actualidad [9] cada vez se hace más evidente la necesidad de no dejar a nadie fuera, máxime cuando son conocidos los beneficios de la estimulación audiovisual en casos de daño cerebral [10].

La accesibilidad en el campo de los videojuegos puede abordarse desde dos perspectivas diferentes:

1) Ayudas de configuración y opciones de aumento en las opciones individuales de cada videojuego, que ofrecen rangos parametrizables o herramientas de asistencia en aspectos relacionados con el color, para realzar determinados elementos de los entornos interactivos o que insertan guías adicionales de tipo auditivo y / o sonoro [11].

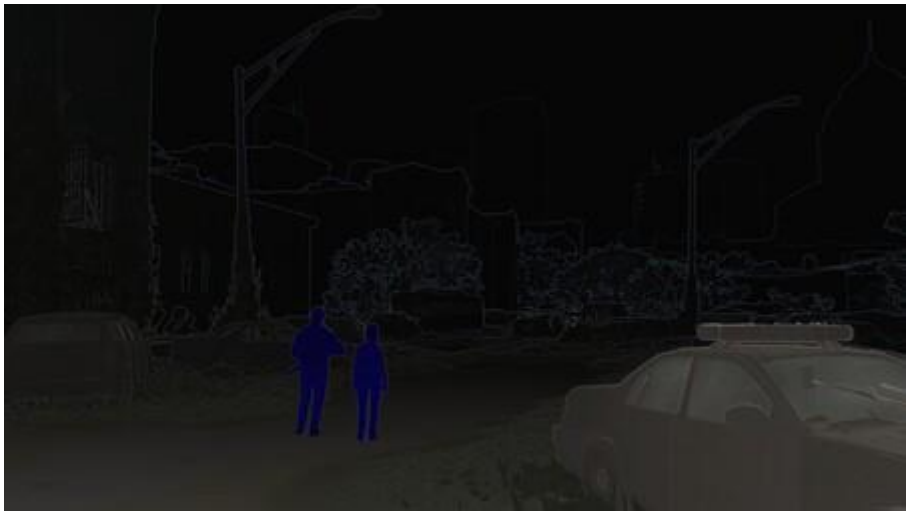


Figura 2 - Ayuda visual accesible para imagen de alto contraste (The Last of Us Part II) [12]

Este enfoque contribuye a solventar las barreras sensoriales que impidan hacer uso de las mecánicas diseñadas, pero dependen de la voluntad de cada equipo de desarrollo y no sirven frente a los impedimentos de tipo físico.

2) Dispositivos de hardware adaptado, que reimaginan la disposición y tamaño de los controles o incluso llegan a delegar algunas funcionalidades tradicionalmente manejadas con las manos en dispositivos accionados mediante las piernas o la boca. Suelen ser soluciones *ad hoc* creadas por los propios jugadores perjudicados por la problemática [13] pero en los últimos años han comenzado a surgir iniciativas profesionales, como **Ablegamers** [14] que diseñan controladores centrados en la accesibilidad, o **SpecialEffect** [15] quienes no solo adaptan periféricos sino que también crean soluciones software basadas en seguimiento de la mirada (*Gaze Tracking*). La propia Microsoft, desde 2018, comercializa el periférico de centro unificado de control denominado **Xbox Adaptive Controller** [16] que fue desarrollado en colaboración con las dos entidades anteriormente mencionadas.

Figura 3 - Mapeo de controles para Xbox Adaptive Controller

Dentro de esta categoría también podemos incluir a ciertos dispositivos de seguimiento ocular como los pertenecientes a la familia **Tobii**, que dispone de asistencia para más de 120 juegos [17].

3. Definición del juego

3.1 Punto de partida

3.1.1 Origen y breve descripción

El concepto del que parte el videojuego a desarrollar proviene de las necesidades especiales de ciertos colectivos, como las personas con movilidad reducida (debido a discapacidades, parálisis cerebral, etc.) y dentro de éstas, especialmente los niños. Las opciones de ocio y expansión personal disponibles para estas personas son muy escasas si no directamente inexistentes, siendo el germen de este proyecto el contacto personal de su autor con un chico que sólo podía comunicarse mediante un *text-to-speech* controlado por un dispositivo de seguimiento ocular Tobii, quien expresó su deseo de poder jugar a un título de conducción / “de coches” (sic) / ”de carreras” (sic) sin tener que depender de terceros.

Por todo lo anterior, el objetivo del presente TFG es el desarrollo de un juego de conducción que pueda ser controlado exclusivamente mediante dispositivos de seguimiento ocular.

3.1.2 Subgénero e inspiración en videojuegos existentes

Dentro del género de la conducción, el videojuego a se acercará más a títulos de corte arcade o desenfadado que a una simulación arcade, tanto por el público al que está dirigido (usuarios jóvenes o niños) como por las propias limitaciones del control de seguimiento ocular con respecto a periféricos más tradicionales como un gamepad o volante.

En lo que respecta a similitudes con otros juegos, originalmente se pretendía proporcionar una experiencia cercana a lo que ofrecen las entregas 3D más recientes de la serie Sonic the Hedgehog, concretamente Unleashed en sus fases diurnas, Generations y Colors, juegos en los que el jugador se desplaza hacia adelante de modo constante pudiendo cambiar entre varios “carriles” para esquivar los obstáculos y recoger objetos o potenciadores. No obstante, durante el desarrollo se comprobó que este tipo de jugabilidad discretizada resultaba limitada y coartaba aspectos como el seguimiento de la cámara, la inserción de nuevos modos de juego o las posibilidades de emplazamiento de objetos, siendo finalmente sustituido por un modo semiguiado en que la aceleración del vehículo a controlar es automática pero el jugador puede orientar libremente el eje de giro para variar el encaramiento.

3.1.3 Tipo de interacción juego-jugador

El manejo de los menús así como del propio control durante los segmentos jugables se realizarán exclusivamente mediante un dispositivo de seguimiento ocular, partiendo de la base de que los usuarios potenciales no tienen ninguna otra posibilidad de interacción debido a limitaciones físicas.

Para los menús, el sistema a implementar consistirá en un puntero guiado por el seguimiento ocular que seleccionará la opción correspondiente al posar la mirada sobre él. Con el fin de no dar lugar a selecciones accidentales y evitar una experiencia artificial de uso que implique no mirar aquellas opciones que no quieran ser seleccionadas en el momento, será necesario mantener el puntero durante un tiempo para validar una selección, ofreciendo *feedback* sobre el proceso mediante una barra de progreso. En el caso de las escenas jugables, se dispondrán paneles alrededor del vehículo a controlar con acciones, como por ejemplo elegir la dirección de giro, que se desencadenarán el pasar el puntero del seguimiento ocular sobre ellos.

3.1.4 Plataformas de destino y de desarrollo

La instalación, configuración y requisitos de los dispositivos de seguimiento ocular conducen a un desarrollo para ordenadores personales, ya sea en forma de equipo desktop o portátil.

3.2 Conceptualización

3.2.1 Historia, ambientación y / o trama

Dado el género al que pertenece el proyecto (conducción deportiva de vehículos) en principio no requiere de un argumento o motivación específica que lo justifique, si bien eso no exime de mantener una homogeneidad en todos sus componentes y de ofrecer un desarrollo de eventos coherente que contribuya a una experiencia inmersiva.

La ambientación más habitual para este tipo de títulos, a grandes rasgos, se engloba en dos categorías:

3) Realista, con una selección y representación tanto de los vehículos o de los circuitos que se amoldan a las leyes del mundo real. No quiere decir que el control, modelos o localizaciones sean reproducciones de entidades existentes o que hayan existido, sino que todo lo representado en el videojuego respeta las leyes de la física y el comportamiento del mundo real tal y como lo conocemos.

   Ejemplos de esta categoría podrían ser Virtua Racing, Forza o Asetto Corsa.

Figura 4 - Forza Horizon 5

4) Fantástico, incorporando componentes o ambientaciones imposibles de encontrar en el mundo real o bien representando situaciones demasiado avanzadas tecnológicamente como para existir en el momento del desarrollo del videojuego. A menudo incorpora un elemento de irrealidad adicional y exageración en sus mecánicas con el uso de potenciadores o incluso armas durante las carreras.

   A esta descripción se amoldarían títulos como Mario Kart, Twisted Metal o Wipeout.

Figura 5 - Wipeout HD [18]

Para el videojuego a desarrollar, la ambientación escogida será fundamentalmente realista, aunque se dará prioridad a un aspecto general desenfadado para hacerlo amigable al público objetivo, incorporando la presencia de objetos o potenciadores en algunas modalidades de juego.

3.2.2 Definición de los personajes / elementos

Los componentes principales que se pretenden incorporar al videojuego son:

5) Vehículo principal: representa a la persona que juega, quien lo controla y utiliza para interactuar con el resto de los componentes.

6) Puntos de ruta o *waypoints*: guían movimiento automatizados necesario para comportamiento guiado mediante Inteligencia Artificial (IA) y hacen de puntos de posicionamiento para otros elementos.
7) Objetos positivos: otorgan puntos cuando son recogidos.
8) Objetos negativos: restan puntos si son recogidos.
9) Vehículos rivales: vehículos controlados por IA que actúan como antagonistas a los objetivos del jugador.
10) Circuito: terreno delimitado dentro del que se sitúan el resto de los objetos que participan durante una carrera. Ninguna acción puede transcurrir fuera del circuito.
11) Cronómetro: registra el tiempo que se tarda en completar el circuito.
12) Paneles de control: pulsadores que permiten al jugador mover al vehículo mediante el dispositivo de seguimiento ocular.
13) Cámara: recoge lo que sucede durante la partida.
14) Botones de selección: permiten interactuar con los menús e interfaces jugador-juego para navegador por las diferentes opciones o modos de juegos.
15) Cursor: indica donde se encuentra el punto de foco del seguimiento ocular que posibilita la interacción.
16) Paneles informativos: textos y rótulos que muestran el control de puntos o vueltas dadas al circuito, informan de acciones realizadas, los nombres de las opciones disponible o el estado del juego en general.

3.2.3 Interacción entre los actores del juego

Las principales interacciones previstas entre los elementos anteriormente mencionados son:

1) Cursor → Paneles de control: habilitan el control sobre el vehículo principal.
2) Paneles de control → Vehículo principal: como consecuencia del punto anterior, la lógica vinculada a los paneles determinará el comportamiento del vehículo principal.
3) Cursor → Botones de selección: determinan el flujo del juego en lo que respecta a tránsito entre menús y escenas, o selección de opciones.
4) Vehículo principal → Objetos positivos / Vehículo principal → objetos negativos: incrementan o decrementan el marcador de puntos.
5) Cámara → Vehículo principal: la cámara seguirá en todo momento al vehículo principal, para permitir un correcto seguimiento de los eventos que transcurren durante el juego.

6) Puntos de ruta → Objetos positivos / Puntos de ruta → Objetos negativos: los puntos de ruta servirán como un punto de aparición y referencia física para ubicar los objetos.

7) Puntos de ruta → Vehículos rivales: el desplazamiento de los vehículos controlados por la IA estará determinado por la posición que ocupen una serie de puntos de ruta situados a modo de balizas de seguimiento.

8) Circuito → Vehículo principal / Circuito → Vehículos rivales / Circuito → Objetos positivos / Circuito → Objetos negativos / Circuito → Puntos de ruta: el trazado del circuito hará las funciones de espacio delimitador para albergar en su interior a aquellos objetos que tienen una influencia directa sobre el transcurso de la carrera.

Adicionalmente, las interacciones mencionadas y la propia lógica del juego generarán una serie de eventos que se reflejarán en el Cronómetro y los Paneles informativos.

3.2.4 Objetivos planteados al jugador

Se plantearán dos modos distintos de enfrentarse a los retos del juego, que variarán los elementos que participan y los objetivos a completar:

1) Carrera en modalidad de puntuación: deberán recogerse objetos positivos que acumulan puntos en el marcador y evitar otros negativos que los restarán. El objetivo final es llegar a la línea de meta al final del circuito con la máxima puntuación posible. A modo de subobjetivo, se contabilizarán el total de objetos de cada tipo recogidos durante la carrera.

2) Carrera en modalidad contra rivales: en lugar de interactuar con objetos, se competirá de forma directa contra vehículos rivales controlados por la IA. El jugador deberá completarse un número de vueltas al circuito antes de que lo haga alguno de estos oponentes.

3.3 Desarrollo y Roadmap

3.3.1 Planificación de objetivos

La hoja de ruta para la finalización exitosa del proyecto pasa por tres objetivos principales, cada uno con sus correspondientes subobjetivos:

1) Implementación de la interfaz de control mediante seguimiento ocular, que a su vez se divide en:

1-1) Interacción en las escenas de juego: si no se logra que una persona juegue a la experiencia propuesta empleando únicamente

controles de seguimiento ocular, el resto de los logros que puedan alcanzarse carecen de relevancia.

1-2) Interacción con menús y componentes secundarios: una vez se haya conseguido una jugabilidad viable, se puede complementar la experiencia con pantallas de título o menús de selección de opciones que también tendrán que ser controlables exclusivamente con el seguimiento ocular.

2) Desarrollo de los componentes y la lógica de las escenas de juego, atendiendo a:

2-1) Vehículo controlado por el jugador: en relación con el subobjetivo 1-1, se atenderá al control, la interacción con el resto de los componentes de la escena de juego como objetos, puntos de ruta, vehículos adversarios, etc.

2-2) Movimiento semiguiado del jugador por el entorno: los juegos de conducción están asociados a controles analógicos que requieren precisión, algo que solo podría ser reproducible por un controlador de naturaleza digital como es el seguimiento ocular mediante la inversión de una cantidad de tiempo y esfuerzo que quedan fuera del alcance de un TFG. Por ello el avance por el circuito transcurrirá de forma semiautomática delegando el avance y aceleración a parámetros automatizados, con el jugador en control de la dirección de giro del eje que determina el encaramiento del vehículo principal.

2-3) Puntos de ruta: el movimiento de vehículos rivales mediante IA y el emplazamiento automático de objetos implican una lógica sin intervención de elementos humanos durante la ejecución del juego que debe construirse alrededor de posiciones preestablecidas a lo largo del circuito.

2-4) Cámara: el seguimiento de la acción debe estar configurado de forma que el jugador no pierda la perspectiva de lo que está presente en la escena de juego, en relación con los factores que puedan afectar de un modo más inmediato al vehículo que controla.

2-5) Objetos y puntuación: ya se trate de objetos positivos o negativos, su distribución espacial y comportamiento cuando el vehículo principal interactúe con ellos debe ser equitativo y justo de forma que cualquier fallo a la hora de recolectarlos o esquivarlos no se sienta como una imprecisión del sistema de colisiones, un error de diseño, o cualquier causa ajena a la propia habilidad o conocimiento del juego, motivando a mejorar con cada partida. Para evitar la repetición y el tedio, la distribución debería ser aleatoria o pseudoaleatoria en cada carrera.

2-6) Vehículos adversarios: los vehículos controlados por la IA deben ofrecer un estímulo competitivo en medida proporcionada. Deben suponer un reto más directo que la recolección de objetos, pero no

oponerse al jugador con excesiva ferocidad para impedir sensaciones de frustración o derrota.

2-7) Elementos adicionales: el jugador debe sentir que le llega información apropiada desde el juego en forma de carteles o letreros que varíen sus mensajes en función del flujo del juego.

3) Presencia, estética, fluidez y entorno *user-friendly*, que complementen la experiencia jugable con una presentación completa y agradable, comprendiendo:

3-1) Construcción de pantalla de título y / o presentación para no comenzar la experiencia jugable abruptamente.

3-2) Entorno gráfico: a la hora de diseñar los componentes del juego, se deben considerar potenciales problemas de visión que puedan verse perjudicados por las combinaciones de colores seleccionadas. Tanto los elementos de interfaz como los vehículos u objetos se configurarán en gamas de colores asociados al par naranja y azul, por ser estos dos diferenciables entre ellos en todas las versiones de condiciones visuales asociadas a deficiencia cromática. En aquellos casos en que se necesite una gama cromática más amplia, se incorporarán los colores neutros blanco y negro.

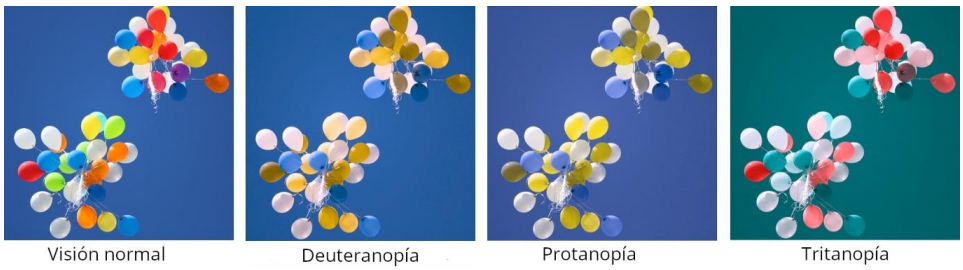


Figura 6 - Tipos de deficiencia cromática (“daltonismo”) [19]

En este subobjetivo hay que atender a la posibilidad de emplear *assets* de origen externo. En este supuesto, debe prestarse atención a que las licencias de uso sean las correctas.

3-3) Música y efectos de sonido: incorporación de pistas musicales en bucle que amenicen el juego, además de efectos de sonido que refuercen la sensación de respuesta inmediata ante ciertos eventos como la interacción con objetos.

Al igual que con los elementos gráficos, debe ponerse cuidado en las licencias de uso si se incorpora contenido que no sea de creación propia.

3-4) Menús de configuración o selección modos de juego: la posibilidad de elegir el modo de juego deseado debe ser intuitiva y de acceso inmediato, necesitando el mínimo número de interacciones posibles para comenzar a jugar.

3-5) Opción de volver a la pantalla principal o jugar de nuevo: una vez finaliza una carrera, el jugador debe tener la opción de volver a jugar la modalidad actual sin necesidad de pasar por el menú principal. También debe dársele la posibilidad de salir en mitad de la carrera o de reiniciar la misma sin obligársele a completar un trazado en el que no desee continuar más.

3.3.2 Cuantificación de tiempo y recursos por objetivo

Se estima que será necesario acometer los objetivos y subobjetivos descritos atendiendo a tres bloques de trabajo principales, que requerirán la siguiente inversión en términos de tiempo y esfuerzo:

1) Interfaz de control mediante seguimiento ocular: 20%

La opción con respecto al seguimiento ocular que se pretende adoptar en un principio corresponde a dispositivos de la gama Tobii, los cuales disponen de un SDK con librerías para la lógica concerniente a los mecanismos de interacción y control, pero será necesario tener en cuenta las limitaciones de esta imposición frente a las necesidades para ajustar la dinámica general de los elementos presentes en las escenas y que la jugabilidad sea adecuadamente fluida. Se requerirán soluciones alternativas y creativas si deben implementarse funcionalidades que no estén contempladas en las herramientas disponibles originalmente, o incluso si en el escenario más pesimista debe descartarse este hardware en favor de otras opciones.

2) Componentes y la lógica de las escenas de juego: 50%

El grueso del desarrollo vendrá de la programación y depuración de los elementos que componen la experiencia jugable, estableciendo sus interacciones y solventando conflictos que devengan de ello. Como ya se ha mencionado anteriormente, a nivel de programación Unity libera de tareas relacionadas con organización física de componentes, asignación de clases o disposición de interfaces, con el código dividido en múltiples scripts individuales que son coordinados por el equipo de desarrollo y asignados o parametrizados según se requiere en cada escena con apoyo de herramientas visuales. La contrapartida a este paradigma más individualizado es que se requiere un control y seguimiento pormenorizado de múltiples elementos independientes que puede escalar en la aparición de errores tediosos de localizar.

3) Presencia, estética, fluidez y entorno: 30%

La mayor dificultad y consumo de tiempo en este respecto vendrá de la búsqueda de recursos artísticos, como gráficos, modelos 3D, música o fuentes de texto. A diferencia de los equipos de desarrollo profesionales, que sí disponen de personal especializado en estas ramas creativas y de tiempo para planificar un ambiente audiovisual exclusivamente diseñado para cada juego, para el presente proyecto con toda seguridad será necesario reutilizar paquetes de *assets* externos legalmente disponibles y que serán debidamente acreditados. En caso de incorporar recursos

provenientes de múltiples fuentes, deberá cuidarse que la apariencia global del proyecto sea cohesionada.

Proyectando la distribución prevista de objetivos mediante un diagrama de Gantt según las semanas disponibles:

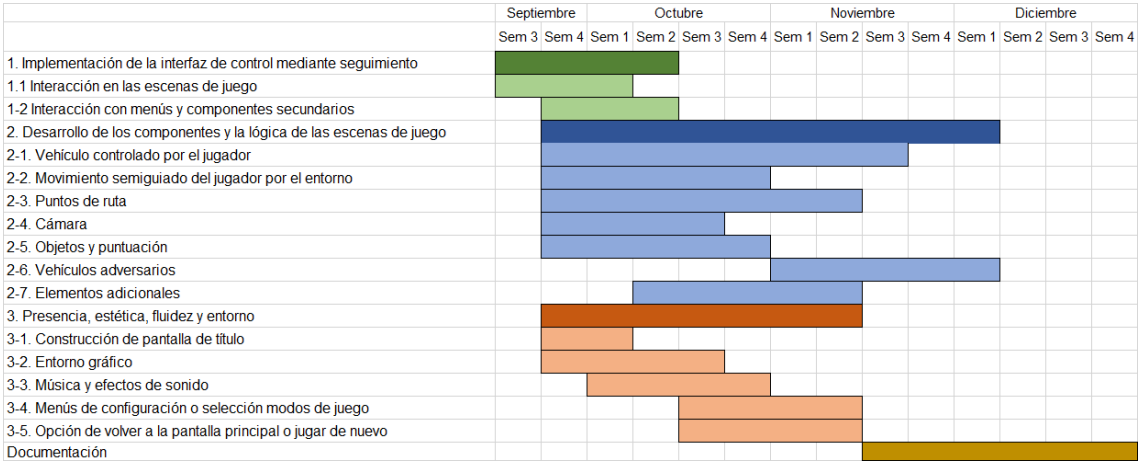


Figura 7 - Diagrama de Gantt para progreso de objetivos

4. Diseño técnico

4.1 Entorno de desarrollo

4.1.1 Elección y justificación

Para el desarrollo del proyecto se ha elegido el motor de videojuegos multiplataforma **Unity** en su versión LTS o *Long Term Support* más reciente al momento del inicio del proyecto, correspondiente a la 2020.3.18f1.

El principal motivo de esta decisión ha sido la experiencia previa del equipo de desarrollo con el motor y todo el ecosistema de herramientas que lo rodean; considerando los limitados y estrictos plazos de tiempo disponibles para ir liberando las distintas versiones del producto, incluyendo la *build* final, supone la opción menos arriesgada y que permite un inicio del trabajo más optimizado frente al desafío de tener que aprender otro tipo de utilidades completamente desconocidas.

Atendiendo únicamente a criterios objetivos, las características de Unity que contribuyen a una exitosa finalización del proyecto, y a su potencial expansión en el futuro, son:

- Plataforma de descarga y uso gratuito, con licencia sin coste para equipos de desarrollo pequeños.
- Más de 15 años de evolución y actualizaciones desde su salida en 2005.
- Utiliza el código abierto. NET para garantizar que las aplicaciones creadas puedan ejecutarse en una amplia variedad de configuraciones de hardware diferentes.
- Posibilidad de programar en C# o Javascript.
- Compatibilidad con múltiples sistemas operativos, desarrollo web y consolas de videojuegos.
- Compatibilidad entre elementos 2D y 3D en un mismo proyecto.
- Asset Store integrado para descarga de recursos, con una oferta variada que incluye un amplio contenido gratuito.
- Metodología *drag-and-drop* para la incorporación de componentes externos al proyecto.
- Comunidad oficial de ayuda, foros y tutoriales.

Unity ofrece la ventaja de que permite realizar de modo visual e interactivo gran parte de la distribución espacial y configuración de las propiedades de los objetos que participarán en las escenas que componen un videojuego, facilitando el focalizar los esfuerzos relativos a

diseño y programación de código en implementar la lógica. Con respecto a ésta, el código necesario se fragmenta en scripts de C# asignados directamente a los objetos que recibirán las propiedades de las clases implementadas.

Unity permite construir videojuegos tanto en 3D como en 2D, mezclando ambas filosofías en distintas escenas o incluso pudiendo convertir una escena tridimensional directamente en bidimensional con una selección en el editor. El presente videojuego se realizará en tres dimensiones para las escenas de juego directo, mientras que los menús tan sólo se percibirán en dos dimensiones por motivos prácticos.

El motor de videojuegos también cuenta con sus propias funcionalidades de depuración y pruebas dentro del entorno de desarrollo por medio de herramientas de modificación temporal de elementos en simulación de ejecución, con todos los cambios revirtiendo a sus parámetros originales al finalizar dicha ejecución de prueba, para analizar comportamientos y cambios.

El desarrollo se realizará bajo una licencia de tipo Personal, que permite el uso del motor de modo gratuito para proyectos personales sin ánimo de lucro o para profesionales que hayan generado unos ingresos inferiores a 100.000$ dólares en los últimos 12 meses [20].

Para la codificación en C# nos serviremos Visual Studio Code como IDE. No es necesario utilizar entornos que soporten organigramas de proyectos o soluciones ya que los scripts se modifican a modo individual, asignando las dependencias entre clases y objetos en el propio entorno de Unity.

4.1.2 Hardware de seguimiento ocular

El dispositivo de seguimiento ocular a utilizar será Tobii Eye Tracker 4C, que implementa tecnología de infrarrojo cercano para seguir del movimiento de los ojos. No contempla funcionalidades de parpadeo o simulación de clics, por lo que la lógica de interacción recae de manera individualizada en el equipo de desarrollo de cada aplicación contra la que quiera hacerse uso.

Figura 8 - Dispositivo de seguimiento ocular Tobii 4C

El dispositivo consiste en una barra horizontal que se sitúa en la parte superior o inferior de un monitor mediante una combinación de anclajes adhesivos e imantados. Una vez se instalan los controladores y se completa el proceso de calibración, aparece en el monitor un cursor guiado por los ojos de la persona usuaria.

En lo que respecta a su integración con Unity, el dispositivo dispone de un SDK de descarga gratuita que incluye librerías para el motor de videojuegos. Asignando a los objetos en las escenas unos componentes de tipo Gaze, reconocerán el cursor de Tobii cuando éste pase por encima de ellos.

Las características técnicas de Tobii 4C son:

- Tamaño: 17 x 15 x 335 mm
- Peso: 95 gramos
- Tamaño de pantalla máximo recomendado: 27” con relación de aspecto 16:9, o 30” con relación de aspecto 21:9
- Distancia de funcionamiento: entre 50 y 95 cm
- Dimensiones de la caja de seguimiento: 40 x 30 cm a distancia de 75 cm
- Longitud del cable USB integrado: 80 cm
- Carga de CPU: promedio del 5% para Core i7, novena generación
- Consumo de energía: ~2 W, con picos de 5-6W en uso máximo
- Tasa de transferencia de datos USB: 100 KB / s

- Frecuencia de muestreo de imagen: 90 Hz
- Iluminadores: infrarrojo cercano (NIR 850)

4.1.3 Requerimientos técnicos

Los requisitos mínimos para ejecutar el entorno de Unity Editor son:

	Windows	**MacOs**	**Linux**
Sistema operativo	Windows 7 (SP1+), Windows 10 y Windows 11. Sólo versiones x64.	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04 y CentOS 7
CPU	Arquitecturas x64 con soporte SSE2	Arquitecturas x64 con soporte SSE2	Arquitecturas x64 con soporte SSE2
API de gráficos	GPUs con soporte DX10, DX11 y DX12	GPUs de AMD e Intel compatibles con Metal	GPUs de Nvidia y AMD con OpenGL 3.2+ o soporte para Vulkan
Otros	Drivers oficiales para otro hardware adicional	Drivers oficialmente soportados por Apple para otro hardware adicional	Entorno de escritorio Gnome sobre sistema de ventanas X11, controlador gráfico oficial de Nvidia o el controlador gráfico Mesa para AMD. Otra Configuración adicional dependerá de la distribución del SO.

A estos requisitos generales, se une una necesidad de hardware adicional, el dispositivo de seguimiento ocular Tobii, cuyas especificaciones recomendadas del sistema son:

- Sistema operativo: Windows 10 64x
- CPU: Intel i5 o i7 de 2,0 GHz
- Memoria RAM: 8 GB

- Conectividad USB: USB 2.0 (cable integrado, USB 2.0 BC 1.2)

Con estos requerimientos generales presentes y atendiendo a criterios de procesamiento, las características técnicas específicas del hardware utilizado como estación de desarrollo durante el proyecto han sido:

- Sistema operativo: Windows 10 Pro x64
- CPU: Intel Core i5-8600 3.10GHz (6 núcleos)
- Memoria RAM: 16 GB DDR4
- GPU: AMD Radeon RX 480 4GB
- Dispositivo de reconocimiento ocular: Tobii Eye Tracker 4C

4.2 Inventario y descripción de las herramientas utilizadas

Para la codificación, construcción y desarrollo del proyecto se han necesitado las siguientes aplicaciones:

- Unity Hub v2.1.3: Herramienta de administración para centralizar y gestionar todos los proyectos e instalaciones de Unity. Permite compatibilizar varias instalaciones de Unity Editor junto con sus componentes asociados, crear nuevos proyectos y abrir proyectos existentes.
- Unity Editor v2020.3.18f: Plataforma de desarrollo 3D en tiempo real correspondiente a la versión de Unity utilizada, que actúa como entorno visual para la creación de las escenas que componen un videojuego. Incorpora herramientas de previsualización, de gestión de recursos asociados al proyecto y utilidades de depuración.
- Unity Assets Store: repositorio de descarga recursos asociado a la cuenta de Unity empleada con Hub y Editor, que permite una integración inmediata en el juego de cualquier elemento adquirido.
- Tobii Unity SDK: proporciona las librerías y ejemplos necesarios para incorporar la tecnología de seguimiento ocular en aplicaciones y juegos de Unity. Incluye una variedad de scripts de muestra, tutoriales y documentación al respecto.
- Microsoft Visual Studio Community 2019 v16.3.4: entorno de desarrollo integrado (IDE) gratuito y extensible para sistemas operativos Windows, desarrollado y distribuido por Microsoft Corporation. Su función es codificar los scripts en C# que compondrán la lógica del videojuego.
- Microsoft Word 365 MSO versión 2110 bajo licencia académica: documentación y anotaciones durante el desarrollo.
- Google Chrome v96.0.4664.45 64 bits: búsqueda de documentación y descarga de recursos adicionales.

4.3 Inventario y descripción de componentes y recursos del juego

4.3.1 Recursos creados específicamente para el proyecto

En este grupo se engloban todos los elementos que han sido creados desde cero para cubrir necesidades del proyecto. Pueden dividirse en varias categorías según su funcionalidad y tipología:

4.3.1.1 Elementos gráficos

- **CountDownText:** Panel informativo para indicar al jugador la cuenta atrás que marca el inicio de la carrera.
- **FadeToBlack:** Fondo negro inicialmente invisible que sirve de transición entre escenas. Cuando se elige alguna opción o se desencadena algún evento que provoca un salto de escena o el reinicio de la escena actual, este fondo se hace gradualmente visible para suavizar estéticamente el cambio.
- **LapsIcons:** En el modo de carrera con rivales, entidades compuestas por iconos y texto que ofrecen información de las vueltas dadas alrededor del circuito por cada uno de los participantes.
- **PanelTime:** Cronómetro que recoge el tiempo transcurrido desde que comienza la carrera hasta que ésta finaliza.
- **ResultsBackground:** Panel informativo que aparece al finalizar la carrera, con mensajes sobre el rendimiento del jugador.
- **TitleImageBackground**: Imagen que aparece al fondo de la pantalla del título. Consiste en una captura del circuito en el que transcurren los modos de carrera, ligeramente desenfocada para ofrecer un lienzo sin distraer excesivamente la atención de los elementos interactivos o solaparlos.

4.3.1.2 Elementos de audio

No se han creado elementos propios relacionados con la faceta auditiva del juego, al margen de las entidades de reproducción Audiolistener que deben incorporarse a las escenas para activar la música o los efectos de sonido.

4.3.1.3 Elementos de interacción

- **BehindGoal:** Punto de control situado inmediatamente antes de la línea de salida / meta. Su función principal es comprobar que el jugador ha avanzado inmediatamente después del inicio de la carrera, sin intentar recorrer el circuito en sentido contrario.

- **CheckPoint:** Punto de control situado en el recorrido intermedio del circuito, para verificar que el jugador ha llegado hasta la mitad del trazado y por tanto deben verificarse otros puntos de control tanto antes como después de la línea de meta.
- **CheckPointRivals:** Punto de control utilizado los vehículos rivales para contabilizar las vueltas que dan al circuito. Si completan las vueltas necesarias antes que el vehículo principal del jugador, activa las condiciones de derrota.
- **CountDownController:** Entidad que dirige la cuenta atrás que marca el inicio de la carrera; hasta que ésta no finaliza, no se activan los scripts que rigen la IA, los menús o los paneles / botones de control.
- **CircularBar**: Barra de progreso circular que registra el tiempo durante el que debe mantenerse el cursor de seguimiento ocular sobre un botón para activarlo. Consiste en un disco que rellena su interior de forma radial a medida que el cursor se mantiene ininterrumpidamente sobre un botón; cuando dicho disco se completa, se desencadena la acción asociada al botón.
- **ControlRight**: Panel que orienta el vehículo principal hacia la derecha, girando el eje delantero hacia esa dirección mientras el cursor de seguimiento ocular esté sobre él.
- **ControlLeft**: Panel que orienta el vehículo principal hacia la izquierda, girando el eje delantero hacia esa dirección mientras el cursor de seguimiento ocular esté sobre él.
- **ExitGame:** Botón del menú principal que permite salir por completo del juego.
- **FinishRace:** En el modo de competición contra rivales, entidad que se activa cuando se cumplen las condiciones de victoria (el jugador ha sido el primero en completar tres vueltas al circuito) o derrota (alguno de los vehículos rivales ha sido el primero en completar tres vueltas al circuito) y da inmediatamente por finalizada la carrera.
- **Goal:** Punto de control en la línea de salida / meta que activa el final de la carrera para el modo de puntuación. Se activa cuando se han completado las vueltas necesarias para dar el juego por completado.
- **GoalLine:** Punto de control en la línea de salida / meta comprueba las vueltas dadas al circuito por el vehículo principal controlado por el jugador.
- **GroupWayPoints:** Entidad que contiene puntos de ruta invisibles, utilizados como puntos de aparición de los grupos de objetos en el modo de puntuación y para guiar el movimiento de los vehículos de IA en el modo de carrera con rivales.
- **MainMenu:** Botón que permite volver al menú principal. Aparece al final de la carrera o en el menú de pausa.

- **MenuOption:** Botón que aparece durante el transcurso de la carrera y abre el menú de pausa, durante el cual se puede reanudar la carrera, reiniciarla desde el principio o salir al menú principal.
- **PlayPoints:** Botón del menú principal que conduce al modo de juego por puntuación.
- **PlayRivals:** Botón del menú principal que conduce al modo de juego contra rivales.
- **RearMirror:** En el modo de carrera con rivales, superficie que hace las funciones de espejo retrovisor, valiéndose de una cámara adicional a la principal para captar los eventos que suceden detrás del vehículo principal.
- **RaceAgain:** Botón que permite reiniciar la carrera en el modo de juego actual sin tener que pasar por el menú principal. Aparece al finalizar la carrera o en el menú de pausa.
- **StartCheckPoint:** Punto de control situado al inicio del circuito. Se utiliza para comprobar que el jugador no trata de volver a la línea de salida/ meta circulando en sentido contrario.
- **Terrain:** Circuito dentro del que transcurren las carreras en los diferentes modos de juego. La zona en la que el jugador, los objetos y los vehículos rivales puede ubicarse se delimita mediante muros con elementos de colisión.
- **UnPauseButton:** Botón que permite salir del menú de pausa y reanudar la carrera.
- **Vehículo principal**: Entidad que representa al jugador, quien podrá dirigir el ángulo de giro del eje delantero mediante paneles de control. La velocidad de desplazamiento se establece automáticamente, con una aceleración progresiva desde el inicio de la carrera que alcanza un máximo prefijado.
- **Vehículos rivales**: Vehículos controlados por una IA. Cada uno se moverá a lo largo del trazado del circuito siguiendo unos puntos de ruta propios, no compartidos con el resto de los vehículos rivales.
- **WallsIner**: Los muros interiores que delimitan el trazado del circuito.
- **WallOuter**: Los muros exteriores que delimitan el trazado del circuito.

4.3.1.4 Scripts

- **AIPlayerCar**: Calcula la velocidad y aceleración que debe aplicarse a las ruedas del vehículo principal, transmitiendo también a éstas la dirección de giro que el jugador establece con los paneles de control.
- **AIRivalCar**: Calcula la velocidad y aceleración que debe aplicarse a las ruedas de un vehículo rival, comprobando la posición de los puntos de control asignados para trazar el movimiento a través del circuito.

- **CheckPointBehindGoal**: Verifica y modifica los estados de los puntos de control asociados al vehículo principal, si éste ha intentado comenzar la carrera en sentido contrario al partir de la línea de salida / meta.
- **CheckPointMiddle**: Verifica y modifica los estados de los puntos de control asociados al vehículo principal, si éste ha alcanzado el punto de control situado a medio camino del trazado del circuito.
- **CheckPointRivals:** En el modo de competición contra rivales, suma una vuelta al marcador de un vehículo rival cada vez que éste completa el trazado completo del circuito pasando por la línea de salida / meta. También contiene la lógica para activar el fin de la carrera con condición de derrota si algún vehículo rival completa tres vueltas antes que el jugador.
- **CheckPointStart:** Verifica y modifica los estados de los puntos de control asociados al vehículo principal, si éste cruza la línea de salida en el sentido correcto al inicio de la carrera.
- **CheckWalls**: Comprueba las colisiones entre los vehículos y los muros tanto interiores como exteriores que delimitan el circuito. En caso de choque, reorientan automáticamente al vehículo hacia el sentido correcto de la carrera.
- **EndRace**: En el modo de puntuación, cuando se cumplen las condiciones para activar el fin de la carrera se encarga de modificar el estado del juego apropiadamente. Esto es: desactiva los componentes de control sobre el vehículo principal, detiene el juego, muestra los paneles de resultados y presenta los botones para reiniciar la carrera o volver al menú principal.
- **EndRaceRivals**: En el modo de competición contra rivales, cuando se cumplen las condiciones para activar el fin de la carrera se encarga de modificar el estado del juego apropiadamente. Esto es: desactiva los componentes de control sobre el vehículo principal, desactiva la IA de los vehículos rivales, detiene el juego, muestra los paneles de resultados y presenta los botones para reiniciar la carrera o volver al menú principal. La información de resultados variará en función de si se han dado las condiciones de victoria o de derrota.
- **FollowWheel**: Orienta los modelos 3D que representan las ruedas en el lado derecho de los vehículos en función de los cálculos realizados por los scripts de control. Sin este script, el giro seguiría realizándose correctamente pero no se apreciaría estéticamente.
- **FollowWheelLeft**: Orienta los modelos 3D que representan las ruedas en el lado izquierda de los vehículos en función de los cálculos realizados por los scripts de control. Sin este script, el giro seguiría realizándose correctamente pero no se apreciaría estéticamente.
- **GazeOptionsMain**: En el menú principal, controla la opción seleccionada mediante el seguimiento ocular para mover el flujo del juego hacia la escena apropiada. También Se ocupa de calcular el porcentaje de la barra de progreso circular si se está seleccionado

alguna opción y también de reiniciarlo si se cambia de opción o no se está seleccionando ninguna.

- **GazeSteer**: Asignado a un panel de control, proporciona el multiplicador que debe aplicarse al ángulo de giro para orientar el vehículo principal en la dirección deseada. Este multiplicador será -1 para girar hacia la izquierda y 1 para girar hacia la derecha.
- **InitialCountDown**: En el modo de puntuación, muestra en orden sucesivo la cuenta atrás previa al inicio de la carrera. Cuando ésta finaliza, inicia el cronómetro y habilita tanto las entidades como scripts que permiten el movimiento del vehículo principal incluyendo los paneles de control para el jugador.
- **InitialCountDownRivals**: En el modo de competición contra rivales, muestra en orden sucesivo la cuenta atrás previa al inicio de la carrera. Cuando ésta finaliza, inicia el cronómetro y habilita tanto las entidades como scripts que permiten el movimiento de todos los vehículos incluyendo los paneles de control para el jugador.
- **ItemInteraction**: En el modo de puntuación, controla el comportamiento de un objeto cuando el vehículo del jugador colisiona con él y modifica el marcador de puntos según sus propiedades.
- **ItemObject:** M En el modo de puntuación, identifica a una entidad como objeto potencial a la que asignar propiedades de objeto: forma geométrica, color, valor en puntos, efecto de sonido y efecto de partículas. Los valores específicos de cada una se determinarán aleatoriamente en tiempo real al inicio de la carrera mediante el script ItemsManager.
- **ItemsManager**: En el modo de puntuación, los objetos se generan en grupos de tres ubicados alrededor de cada uno de los puntos de ruta situados a lo largo del trazado, conteniendo cada grupo exactamente un objeto positivo, un objeto negativo y un hueco sin objeto. Con este script se aleatorizan las posiciones que ocupa cada objeto dentro de su grupo y se les asignan las propiedades; como excepción, en el grupo situado frente a la línea de salida / meta el objeto positivo siempre ocupará el lugar central.
- **LapsCounter:** En el modo de competición contra rivales, suma una vuelta al marcador del jugador cada vez que éste completa el trazado completo del circuito, pasando por la línea de salida / meta. También contiene la lógica para activar el fin de la carrera con condición de victoria si el jugador completa tres vueltas antes que los vehículos rivales.
- **MenuOptionsRace:** Durante la carrera, comprueba si el jugador intenta abrir el menú de pausa situando el seguimiento ocular sobre el botón habilitado para ello. En caso de cumplirse la condición, detiene la carrera y muestra los botones para cancelar la pausa, reiniciar la carrera desde el principio o volver al menú principal.
- **NoMovementCamera:** Aplica ligeras correcciones en los ejes X y Z a la cámara que sigue al jugador. De no utilizarse, pueden producirse

efectos visuales de inestabilidad si se dan situaciones bruscas como por ejemplo colisiones contra los muros.

- **RestartRace:** En el menú que aparece al finalizar la carrera, controla la opción seleccionada mediante el seguimiento ocular para mover el flujo del juego hacia la escena apropiada. También Se ocupa de calcular el porcentaje de la barra de progreso circular si se está seleccionado alguna opción y también de reiniciarlo si se cambia de opción o no se está seleccionando ninguna.
- **SceneBeforeTitle:** Muestra una breve escena de transición con fundido a negro entre la *Splash Screen* de Unity y el menú principal.
- **TimeManager:** Contabiliza el tiempo transcurrido desde el inicio de la carrera, transcribiéndolo a la entidad que representa al cronómetro.
- **TrackPlayer:** Regula la velocidad de giro y la distancia de seguimiento de la cámara principal con respecto al vehículo del jugador.
- **UnPauseScript:** En el menú de pausa durante la carrera, reanuda la acción y oculta de nuevo los botones de opciones si el jugador elige reanudar el juego.

4.3.2 Recursos existentes incorporados al proyecto

Se incluyen aquí todos los elementos de terceros que han sido reutilizados en el desarrollo del juego. Consultar el Anexo II para una información más detallada con respecto a la acreditación y las licencias de uso pertinentes.

Considerando las mismas subcategorías utilizadas para los elementos de creación propia, tenemos:

4.3.2.1 Elementos gráficos

- **Asphalt_texture9:** Textura de suelo con apariencia de asfalto para decorar la zona transitable del circuito por la que pueden circular los vehículos.
- **baum_id1_mobile:** Árbol de baja resolución utilizado para decorar las zonas no transitables interiores y exteriores del circuito.
- **baum_id0:** Árbol de baja resolución utilizado para decorar las zonas no transitables interiores y exteriores del circuito.
- **baum_hd_pine_fbx:** Árbol de baja resolución utilizado para decorar las zonas no transitables interiores y exteriores del circuito.
- **CFX2_EnemyDeathSkull:** Efecto de partículas con aspecto de calavera *cartoon* que aparece cuando el vehículo principal colisiona con un objeto negativo.

- **CFX_Firework_Trails_Gravity:** Efecto de partículas consistente en fuegos artificiales que aparece cuando el vehículo principal colisiona con un objeto positivo.
- **GrassUV02:** Textura de suelo con apariencia de hierba para decorar las zonas no transitables interiores y exteriores del circuito.
- **Rx Sports**: Modelo 3D con forma de coche deportivo que representa visualmente a los vehículos que participan en la carrera.
- **SoftStar**: Modelo 3D con forma de estrella que representa visualmente a los objetos positivos.
- **SphereGemSmall:** Modelo 3D con forma de diamante con aristas que representa visualmente a los objetos negativos.

4.3.2.2 Elementos de audio

- **253614__xtrgamr__honda-cr-v-start:** Efecto de sonido que simula el motor de un coche, utilizado para indicar que se ha seleccionado una opción en los menús.
- **446125__justinvoke__collision-2:** Efecto de sonido de impacto leve utilizado para indicar que se ha colisionado con un objeto negativo.
- **446130__justinvoke__race-start-countdown:** Efecto de sonido que representa un pitido grave, utilizado durante la cuenta atrás al inicio de la carrera.
- **446142__justinvoke__race-start:** Efecto de sonido de alerta suave, utilizado cuando finaliza la cuenta atrás al inicio de la carrera y ésta da comienzo.
- **446145__justinvoke__freeze-hit:** Efecto de sonido de impacto leve utilizado para indicar que se ha colisionado con un objeto positivo.
- **POL-high-five-short:** Pista musical que se reproduce en bucle durante la carrera.
- **POL-living-lucky-short:** Pista musical que se reproduce en bucle tras la carrera, mientras se muestra la pantalla de resultados.
- **Tires Squealing-SoundBible.com-1814115127:** Efecto de sonido que simula un derrape, utilizado en la pantalla del título.

4.3.2.3 Elementos de interacción

- **Gazeplot:** El cursor utilizado por el jugador para interactuar con los elementos en el videojuego mediante el sensor de seguimiento ocular. Las librerías de Tobii proporcionan toda la lógica tanto para la transformación del movimiento de los ojos en coordenadas de pantalla como para la creación del elemento gráfico que da forma al cursor.

4.3.2.4 Scripts

- **GazeAware:** Debe añadirse a un objeto de Unity, que disponga de Collider, para ser reconocido como elemento interactuable por parte del cursor de seguimiento ocular.
- **Tobii Initializer:** Contiene las instrucciones de inicialización del dispositivo Tobii para el correcto reconocimiento e interacción de sus funcionalidades en las escenas.

4.4 Esquemas de arquitectura

4.4.1 Diagrama de flujo general

El flujo del programa y la interacción entre las escenas principales que componen el juego puede resumirse en el siguiente esquema:

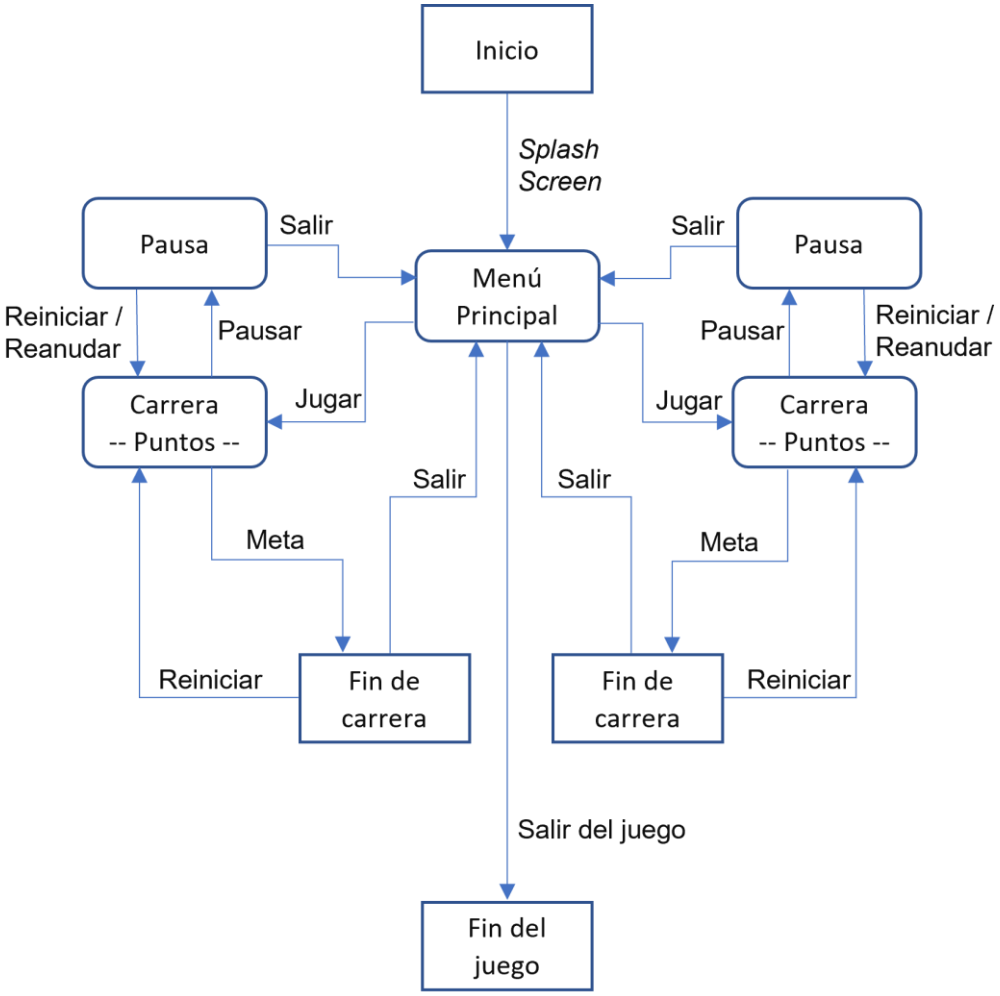


Figura 9 - Diagrama general de funcionamiento del juego

.

Una vez iniciado la aplicación, tras el *splash screen* se llega a la pantalla del título que contiene el menú principal. Desde aquí es posible salir de la aplicación o bien seleccionar uno de los dos modos de juego: carrera con puntuación o carrera contra rivales. En cada uno de estos dos modos, es posible pausar el juego para después reanudarlo desde el mismo punto, reiniciar la carrera desde el principio o volver al menú principal; si la carrera transcurre hasta lograr los objetivos o meta para su finalización, se permite reiniciar la carrera desde el principio o regresar al menú principal.

Describiremos a continuación la estructura y elementos participantes de cada una de las escenas principales, indicando los scripts que cada objeto pueda tener asociado. Se omitirán componentes comunes básicos a todas las escenas, a menos que incluyan algún script especial que sea necesario destacar. Estos componentes son:

- Cámara principal
- Reproductores de audio
- Script de inicialización de seguimiento ocular
- Cursor de seguimiento ocular
- Fuentes de iluminación
- Cortina de fundido a negro
- Barra de progreso circular para opciones
- Paneles de información
- Terreno
- Elementos decorativos (p.ej. árboles) sin función interactiva
- Muros delimitadores del circuito

4.4.2 Menú principal

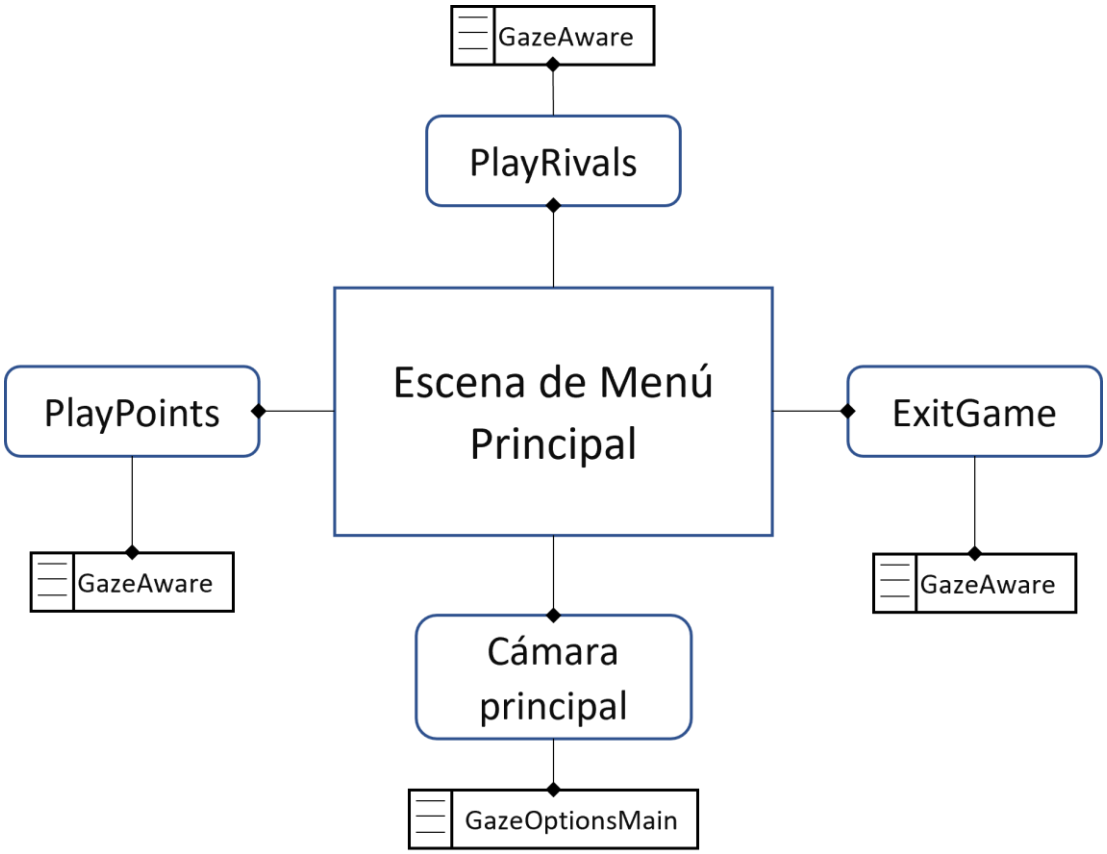


Figura 10 - Diagrama de la pantalla del menú principal

Esta escena supone el punto central de navegación al resto de escenas de juego principales y para salir de la aplicación. Sus componentes interactivos consisten en los botones que dan acceso a dichas escenas, los cuales deben tener asignado un script de tipo *GazeAware* para ser reconocidos por el seguimiento ocular.

El script *GazeOptionsMain* para gestionar la opción potencialmente seleccionada mediante el seguimiento ocular se adjunta a la cámara principal, por ser un elemento que siempre debe estar presente y que, al no verse influido por ningún tipo de interacción, no corre riesgo de verse alterado o modificado de ninguna forma que afecte al control de los otros tres componentes.

4.4.3 Carrera en modalidad de puntuación

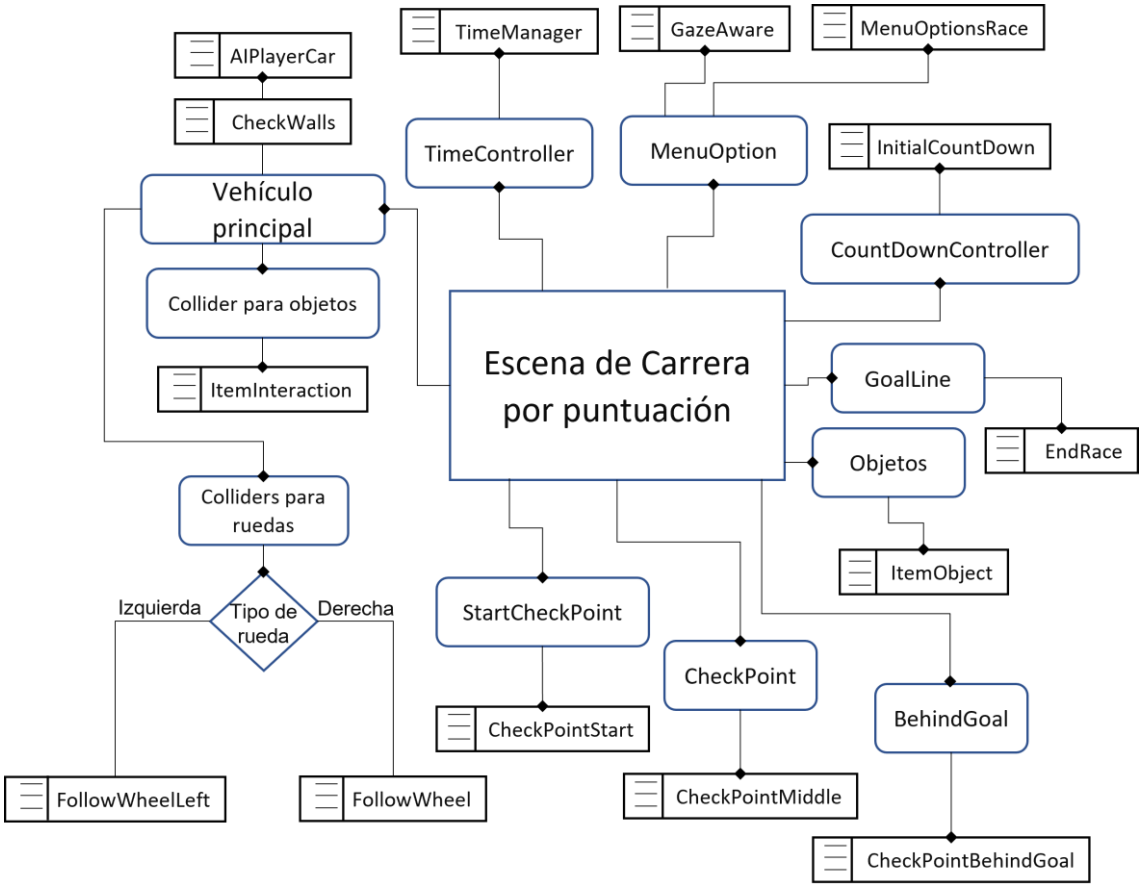


Figura 11 - Diagrama de carrera en modalidad de puntuación

En esta escena transcurre la acción correspondiente a la carrera en la que el jugador debe recoger objetos positivos que aumente su puntuación mientras esquiva los objetos negativos que decrementan el marcador.

El inicio de la carrera viene marcado por la cuenta atrás programada en el script *TimeManager*. Una vez ésta finaliza, se activan el resto de scripts para permitir el control del vehículo del jugador y el registro del tiempo transcurrido durante la carrera.

Para gestionar la colisión con los objetos, el vehículo principal dispone de un Collider dedicado que tiene asignado el script *ItemInteraction*, encargado de comprobar las propiedades establecidas mediante *ItemObject* y modificar el contador de puntos apropiadamente además de reproducir los efectos gráficos y sonoros asociados.

Varios puntos de control distribuidos a lo largo del circuito sirven para verificar que el vehículo principal sigue el trazado en el sentido apropiado. *CheckPointStart*, *CheckPoint* y *CheckPointBehindGoal* se ocupan de activar y desactivar estos puntos de control a lo largo del

trazado, de modo que el objeto que marca la meta, junto a su script *EndRace* que activa el final de la carrera, solo estará habilitado si se ha interactuado con todos ellos en el orden correcto.

En cualquier momento durante la carrera, puede activarse el objeto que contiene el *MenuOptionsRace* para pausar la acción y abrir el menú de opciones, que permite reiniciar desde el principio, salir al menú principal o reanudar sin cambios.

Cuando se activa el objeto que contiene *EndRace*, se paraliza el juego de modo similar a lo que ocurre al abrir el menú de opciones y se da por finalizada la carrera mostrando el panel de resultados. En esta ocasión no se presentará opción para reanudar, permitiendo únicamente reiniciar o salir al menú principal.

4.4.4 Carrera en modalidad contra rivales

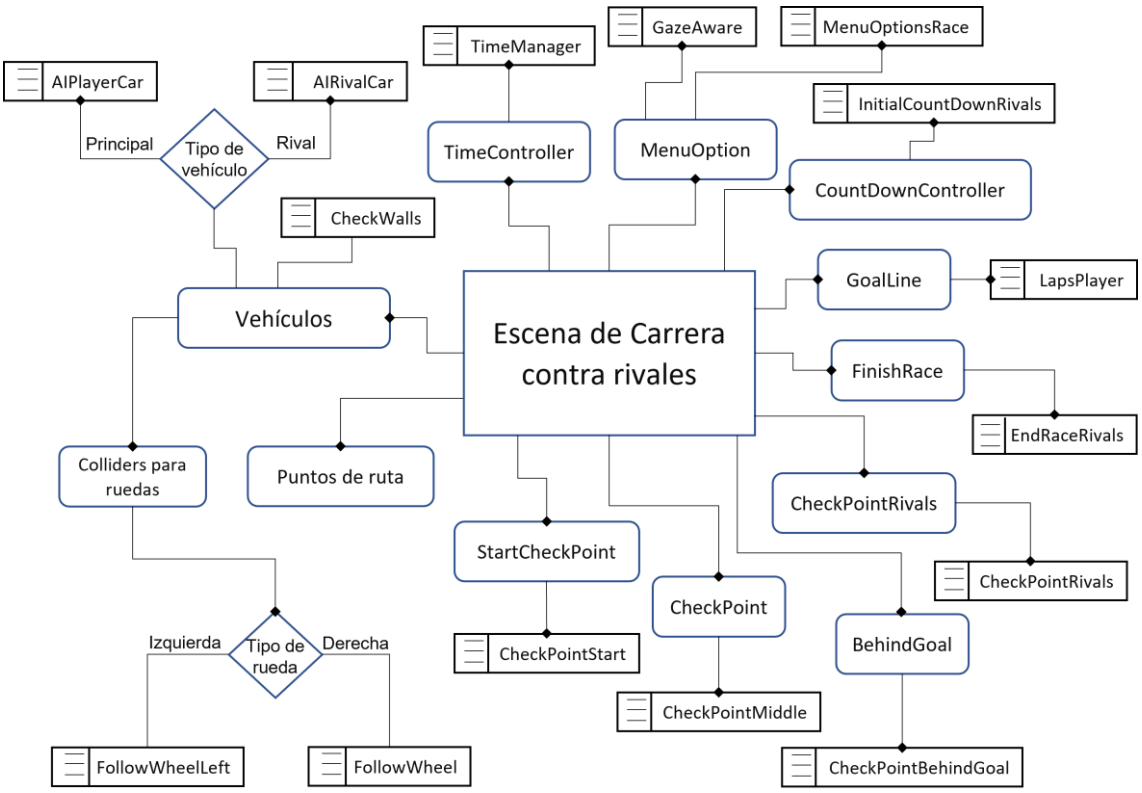


Figura 12 - Diagrama de carrera en modalidad contra rivales

La primera diferencia de la modalidad de carrera contra rivales con respecto a la de puntuación es la ausencia de objetos y de todos los scripts relacionados con sus funcionalidades.

Se incluyen una serie de vehículos adicionales al principal que actúan como rivales. Cada uno de ellos lleva asignado un script de tipo *AIRivalCar* que le hace seguir un grupo de puntos de ruta preestablecidos para cumplir su objetivo de dar tres vueltas completas al circuito antes que el jugador, además de llevar el registro del número de vueltas particulares que ese vehículo ha logrado.

A los puntos de control existentes que afectan al jugador, se añade uno nuevo con el script *CheckPointRivals* en la línea de salida/meta para sumar una vuelta a un vehículo rival cada vez que pasa por él. Si en algún momento algún vehículo rival logra acumular tres vueltas, se activará *FinishRace* para dar por finalizada la carrera mostrando el panel de resultados con condiciones de derrota. Se dará al jugador la opción de reiniciar la carrera o volver al menú principal.

En la línea de salida / meta también se produce un cambio que afecta al jugador, ya que *GoalLine* sustituye el script para finalizar la carrera por otro, *LapsPlayer*, para contabilizar las vueltas que el jugador ha dado al circuito. Si éste logra dar tres vueltas completas antes que alguno de los vehículos rivales, se activará *FinishRace* para dar por finalizada la carrera mostrando el panel de resultados con condiciones de victoria. Se dará al jugador la opción de reiniciar la carrera o volver al menú principal.

En cualquier momento durante la carrera, puede activarse el objeto que contiene el *MenuOptionsRace* para pausar la acción y abrir el menú de opciones, que permite reiniciar desde el principio, salir al menú principal o reanudar sin cambios.

4.4.5 Menú de pausa / fin de carrera

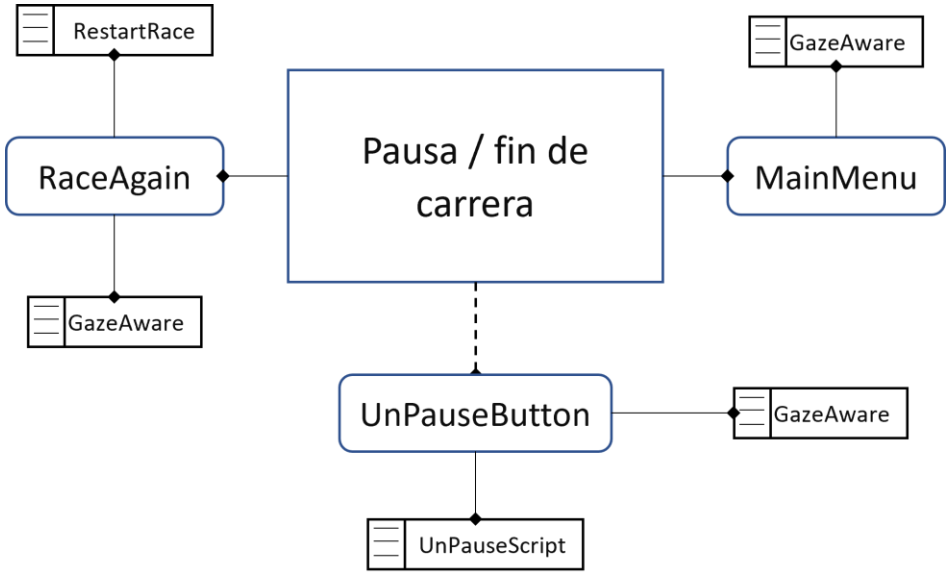


Figura 13 - Diagrama de pantalla de pausa

Si bien no se trata de una escena aislada sino de un subestado dentro de cada una de las escenas correspondientes a los modos de carrera, el menú de pausa y fin de carrera debe ser reseñado por suponer un estado que modifica el transcurso del programa y brinda nuevas opciones al tiempo que anula el resto de los elementos interactivos mientras dura.

Al activar el menú de pausa, se interrumpe la secuencia de la acción principal y aparecen en pantalla tres opciones seleccionables mediante seguimiento ocular. Dos de estas opciones, correspondientes a la elección para reiniciar la carrera desde el principio y a volver al menú principal, se controlan mediante *RestartRace* el cual se asocia al botón responsable de reiniciar. De elegirse alguna de ellas, se cargará la escena apropiada (ya sea la escena actual o bien la pantalla del título) con todas las posibles variables que gobiernan sobre los objetos y atributos situados en sus valores por defecto.

La tercera opción disponible permite reanudar el transcurso de la carrera desde el mismo instante en que fue interrumpida. La lógica que hace esto posible se encuentra en *UnPauseButton* asociada al botón para cerrar el menú de pausa.

El menú para pausar es en práctica el mismo que aparece cuando se cumplen las metas y condiciones que dan la carrera por finalizada. La diferencia es que en este último caso no se habilita el botón para cancelar la pausa, por lo que la secuencia de juego permanece indefinidamente interrumpida hasta que se seleccione alguna opción de entre reiniciar o salir al menú principal.

4.4.6 Paneles de control de giro del vehículo

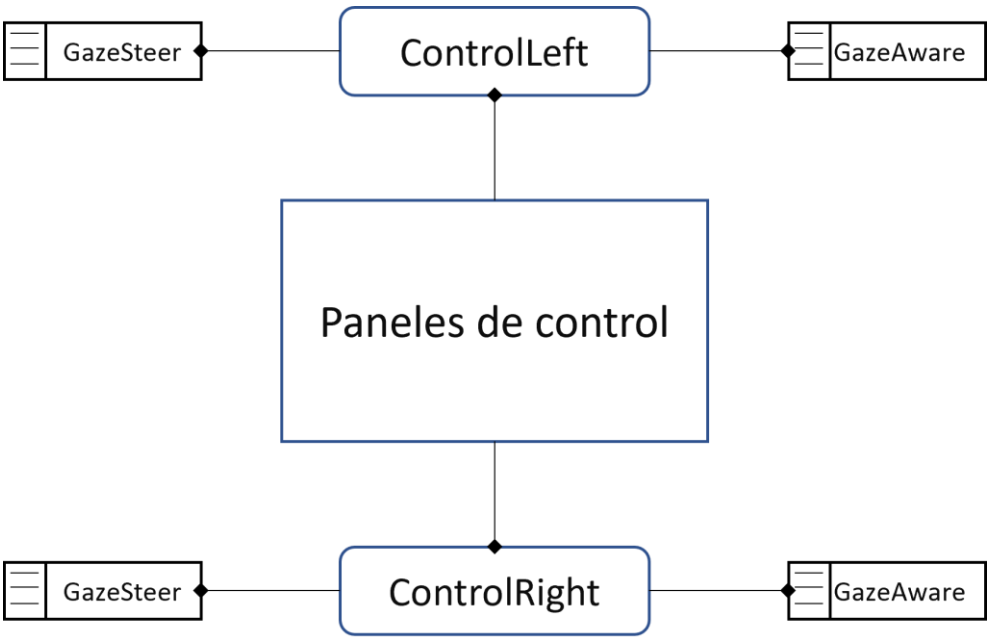


Figura 14 - Diagrama de paneles de control de vehículo principal

Los paneles de control que permiten el control sobre el giro del vehículo presentan la misma lógica, siendo la diferencia el valor del sentido de giro de las ruedas que se adjudica en *GazeSteer* y que se aplicará a los cálculos de la trayectoria en *AIPlayerCar*. Para el panel que permite girar hacia la izquierda (ControlLeft) el sentido se establece en -1 y para el panel que posibilita el giro hacia la derecha (ControlRigtht) será 1; si el seguimiento ocular no está sobre ningún panel, el sentido de giro será 0 y las ruedas se orientarán en línea recta hacia adelante.

4.5 Comentarios sobre botones y paneles de control

Las características de las librerías que Tobii proporciona para incorporar en Unity el seguimiento ocular han presentado un escollo principal a superar para poder implementar los interfaces de control.

El reconocimiento de un objeto como entidad capaz de interactuar con el cursor de seguimiento ocular se logra mediante la combinación de dos condiciones:

1) Tiene adjunto el script *GazeAware* incluido en las librerías de Tobii. Este fragmento de código proporciona al objeto la propiedad *HasGazeFocus*, transformándolo en reconocible por el foco del cursor.

```
//-----------------------------------------------------------------------
// Copyright 2016 Tobii AB (publ). All rights reserved.
//-----------------------------------------------------------------------

using Tobii.G2OM;
using UnityEngine;

namespace Tobii.Gaming
{
    /// <summary>
    /// Component that makes the game object GazeAware, meaning aware if the
    /// user's eye-gaze is on it or not.
    /// </summary>
    [AddComponentMenu("Tobii/Gaze Aware")]
    public class GazeAware : MonoBehaviour, IGazeFocusable
    {
        public bool HasGazeFocus { get; private set; }

        /// <summary>
        /// Function called from the gaze focus handler when the gaze focus for
        /// this object changes. Since the implementation is explicit, it will
        /// not be visible on instances of this component (unless cast to
        /// <see cref="IGazeFocusable"/>).
        /// </summary>
        /// <param name="hasFocus"></param>
        public void GazeFocusChanged(bool hasFocus)
        {
            HasGazeFocus = hasFocus;
        }
    }
}
```

Figura 15 - Script GazeAware de Tobii

2) Dispone de una propiedad Collider. Los límites de la caja de colisión son utilizados por el cursor para cambiar de estado con respecto al objeto; esto es, saber si está dentro o fuera del mismo.

Habitualmente los botones y métodos de control en Unity se crean como componentes anidados dentro del *Canvas*, que es el componente para los elementos de UI o interfaz. El *Canvas* está siempre superpuesto sobre la escena, es siempre visible con independencia de a dónde esté enfocando la cámara principal y los elementos en él no interactúan directamente con el resto de los objetos presentes, no pudiéndoseles asignar propiedades Collider; esta última particularidad hace por tanto imposible crear entidades de *Canvas* que puedan interactuar con el cursor de seguimiento ocular.

El modo de solventar estas limitaciones ha consistido en crear los elementos interactivos como objetos convencionales en la escena anidándolos a la cámara principal; de este modo, es posible simular componentes de interfaz con propiedades Collider que siempre estén visibles en la escena, ya que tendremos en pantalla botones y paneles que se moverán junto a la cámara. Para que el tamaño con el que aparecen en pantalla sea apropiado y equivalente al que tendrían de haber estado anidados al Canvas, y también evitar interacciones accidentales con otros objetos como el vehículo principal del jugador, se les han reducido las proporcionades a niveles mínimos y se han aproximado todo lo posible al foco de visión de la cámara teniendo en cuenta la posición y área jugable en que deben estar disponibles.

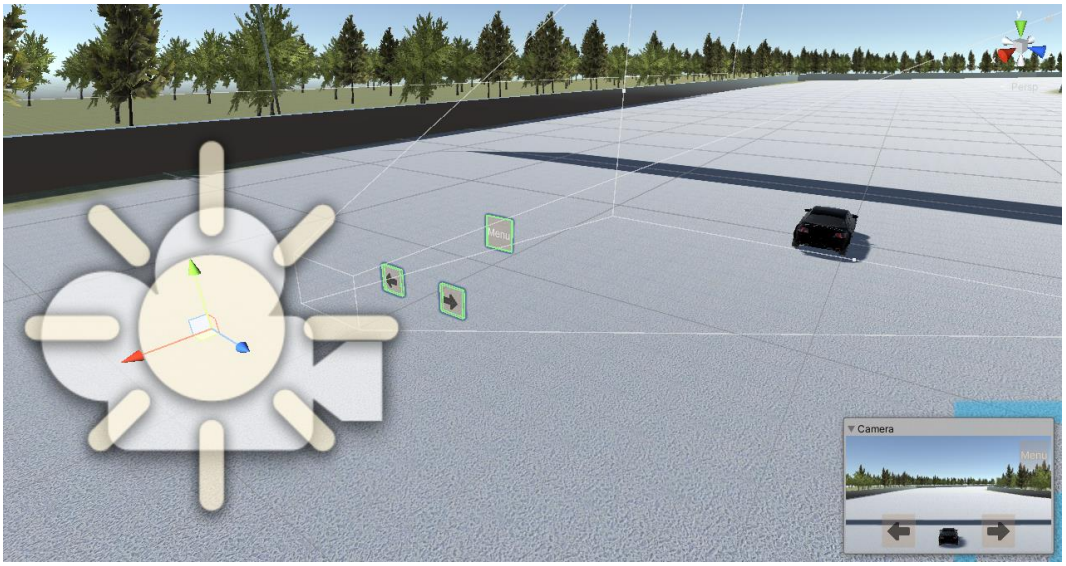


Figura 16 – Posición de los paneles de control respecto a la cámara

5. Diseño de niveles

Uno de los objetivos principales del proyecto ha sido demostrar las capacidades del seguimiento ocular para adaptar géneros de juegos que no están tradicionalmente desarrollados con este control en mente. Por ello, a la hora de diseñar el circuito se ha optado por una solución sencilla que no fuese en detrimento de la experiencia propuesta y que no complicase en exceso las labores de testeo.

El trazado del circuito sigue un patrón circular que comienza y finaliza en la misma línea de salida / meta delimitada por una banderola de posición; al inicio el jugador tan sólo percibirá la sombra que proyecta sobre el circuito por comenzar debajo pero su altura hará que sea gradualmente visible a medida que vuelva a aproximarse al final de la carrera.

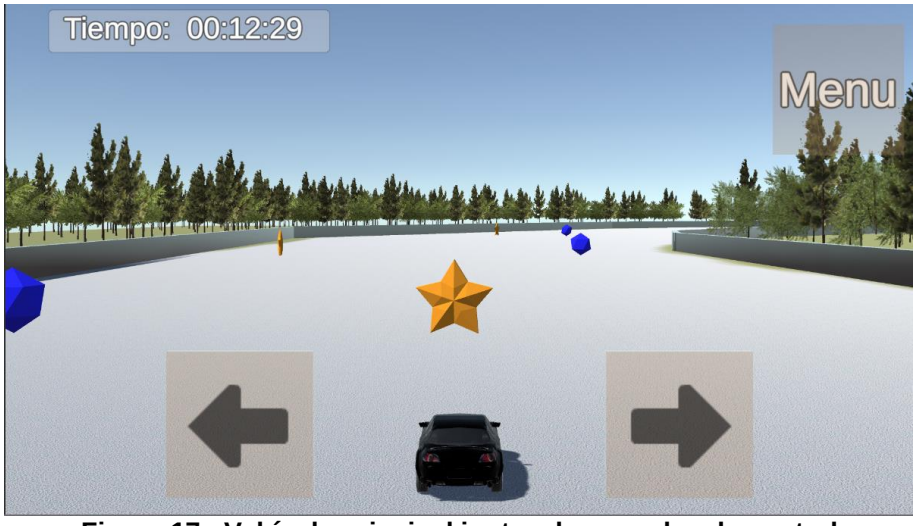


Figura 17 - Vehículo principal junto a los paneles de control

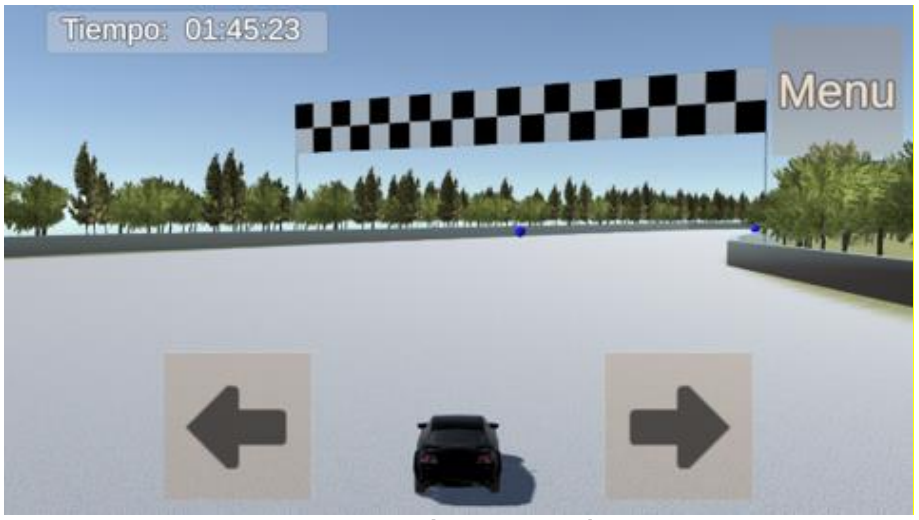


Figura 18 - Aproximándose a la línea de meta

El área transitable del circuito es identificable por la textura de suelo de pavimento asfaltado y está delimitada mediante formas geométricas con propiedades de Collider, que hacen las veces de muros o vallas de contención. En caso de choque contra estas paredes, para paliar la combinación de controles limitados para el giro, aceleración automática y ausencia de marcha atrás que podría convertir en una experiencia tediosa el volver a orientarse en el sentido correcto del trazado, el script *CheckWalls* reorienta al vehículo automáticamente hacia el vector cruzado calculado entre su dirección original en el momento del impacto y la dirección del objeto que representa al muro contra el que haya golpeado. Por motivos prácticos, este script se aplica también a los vehículos rivales en el modo de competición.

5.1 Puntos de ruta - Ubicación de los objetos en el modo de puntuación

En el modo de juego de puntuación, existen a lo largo del circuito varios puntos de ruta no visibles en posiciones prefijadas y orientados de forma perpendicular a la dirección del trazado. Cada uno de estos puntos de ruta sirve como punto de generación de un grupo de objetos.

Siempre se tendrán tantos grupos de objetos como puntos de ruta, y el contenido de cada grupo será un array de tres objetos que contiene una única instancia de cada uno de los siguientes elementos:

- Un objeto positivo, que añade puntos al marcador del jugador cuando éste colisiona con él. Se identifica de forma unívoca mediante un modelo 3D diferenciado y efectos tanto de partículas como de sonido que se reproducen al ser recolectados.
- Un objeto negativo que añade puntos al marcador del jugador cuando éste colisiona con él. Al igual que los objetos positivos, consta de un modelo 3D y efectos tanto de sonido como de partículas propios.
- Un hueco, materializado como un objeto nulo al que no se asocia ninguna propiedad válida ni se establece como activo (nótese que en la siguiente figura se ha representado el hueco mediante su caja de colisión invisible).

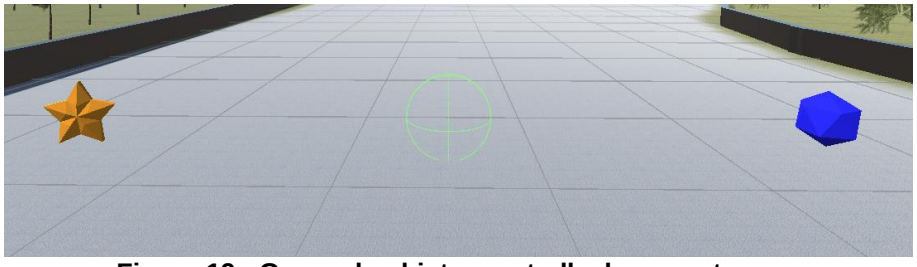


Figura 19 - Grupo de objetos: estrella, hueco y trampa

La posición que cada objeto ocupa en el array se determina aleatoriamente durante la generación del circuito, exceptuando el grupo situado frente a la línea de salida; en éste el objeto positivo siempre está en el centro para coincidir con la trayectoria inicial del jugador y solamente el objeto negativo y el hueco serán reordenados.

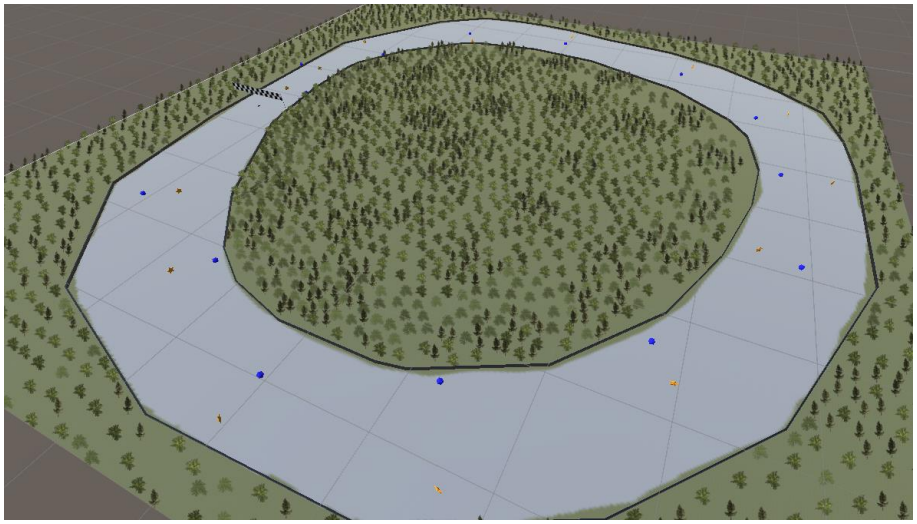


Figura 20 - Vista aérea de objetos distribuidos por el circuito

Al establecer un número de objetos con posiciones prefijadas y un contenido total no variable se busca establecer un sistema de puntuación justo que brinde al jugador la posibilidad de conseguir siempre la máxima puntuación si logra dominar el juego, al tiempo que aleatorizar las ubicaciones internas de cada objeto en el grupo intentan que no baste con aprenderse el trazado del circuito para lograrlo y cada partida sea dinámica.

5.2 Puntos de ruta – Trayectoria de los vehículos rivales en el modo de competición

En el modo de carrera contra rivales, cada vehículo rival tiene asociado un grupo de puntos de ruta no visibles que determinan las posiciones que marcará como objetivos para ir completando vueltas al circuito.

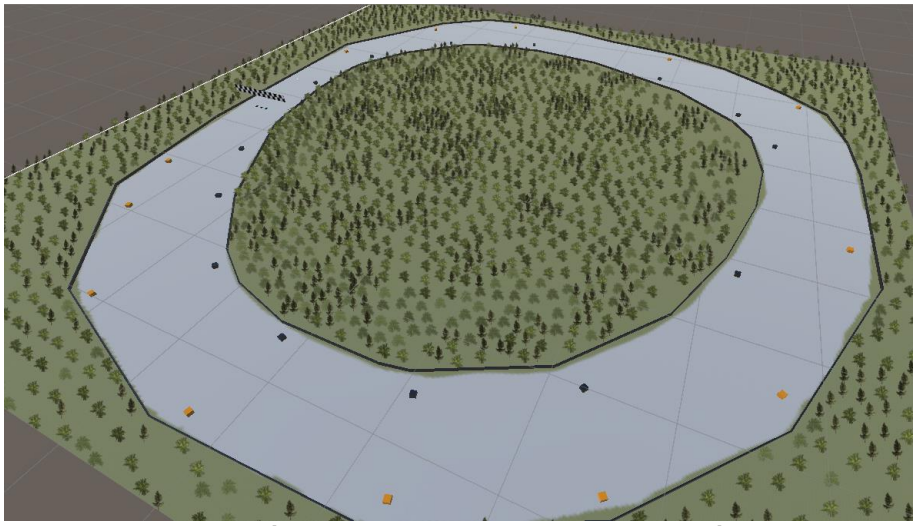


Figura 21 - Vista aérea de puntos de ruta para los vehículos rivales

Cada grupo de puntos de ruta contiene un array de objetos en posiciones prefijadas que se colocan durante la generación del circuito, marcando como punto inicial a seguir el primero situado inmediatamente delante de la línea de salida. El vehículo rival acelera automáticamente y orienta su dirección hacia el punto de ruta establecido como objetivo actual; una vez que lo alcanza, se toma el siguiente punto de ruta en el array y se marca como nuevo objetivo a seguir, repitiendo el proceso cada vez que vuelve a alcanzarlo para hacer que recorra el trazado designado. Una vez alcanza el último punto de ruta, ubicado inmediatamente antes de la línea de meta, se designa como objetivo el primer punto del array para volver a comenzar el ciclo.

Estos puntos de ruta no tienen ningún efecto sobre el jugador. Si el vehículo principal toca o pasa a través de alguno, no se producirá efecto alguno.

6. Análisis de costes

6.1 Equipamiento utilizado

Las especificaciones de la estación de trabajo utilizada como equipo de desarrollo y pruebas se corresponden con las descritas en el capítulo dedicado los requerimientos técnicos. Ampliando la información con componentes adicionales necesarios para el funcionamiento del hardware y visualización de salida, o conexión a internet para subidas sucesivas del progreso:

- CPU: Intel Core i5-8600 3.10GHz (6 núcleos)
- Memoria RAM: 16 GB DDR4
- GPU: AMD Radeon RX 480 4GB
- Dispositivo de reconocimiento ocular: Tobii Eye Tracker 4C
- Fuente de alimentación de 650W
- Monitor M227WDP
- Disco de almacenamiento interno Seagate Firecude SSHD 2TB
- Conexión de fibra simétrica 1Gbps

6.2 Coste del desarrollo

6.2.1 Cálculo de horas de trabajo y coste/persona

El equipo de desarrollo durante todo el transcurso del proyecto ha consistido en **un (1) ingeniero técnico en informática** trabajando, de promedio, en el siguiente horario entre el **15-09-2021** y **15-12-2021**:

- Lunes a jueves: 4 cada día (c/d)
- Viernes: 5 horas c/d
- Sábado y domingo: 8 horas c/d

Total de horas semanales: 37 horas

De acuerdo con la información proporcionada por el Instituto Nacional de Estadística en el periodo de tiempo registrado más reciente, correspondiente al segundo trimestre del año 2021, el coste / hora salarial medio en el total del territorio nacional es de 15,83€ [21] mientras que el portal de búsqueda de empleo Talent tasa esta cantidad en 13,08€ para los ingenieros técnicos en particular [22]. Tomaremos el

promedio de estos dos datos, es decir **14,56€**, para estimar cual ha sido el coste de trabajo total.

Tratando todas las horas por igual, sin tener en cuenta posibles consideraciones relativas a horas extra, contribuciones a cotizaciones o sobrecostes por fines de semana:

1) 37 horas semanas x 4 semanas al mes = 148 horas / mes
2) 148 horas / mes x 3 meses completos (período 15 de septiembre a 15 de diciembre) = 444 horas de trabajo total
3) 444 horas x 14.56€ coste / hora = 6.464,64€ coste total

El coste total salarial promedio estimado es de **6.464,64€**

6.2.2 Coste energético y equivalencia en CO2

Para el cálculo del coste energético y huella de carbono equivalente, el primer paso será introducir las especificaciones del equipo utilizado durante el proyecto en una utilidad online que permite estimar el consumo promedio en carga de vatios [23]:

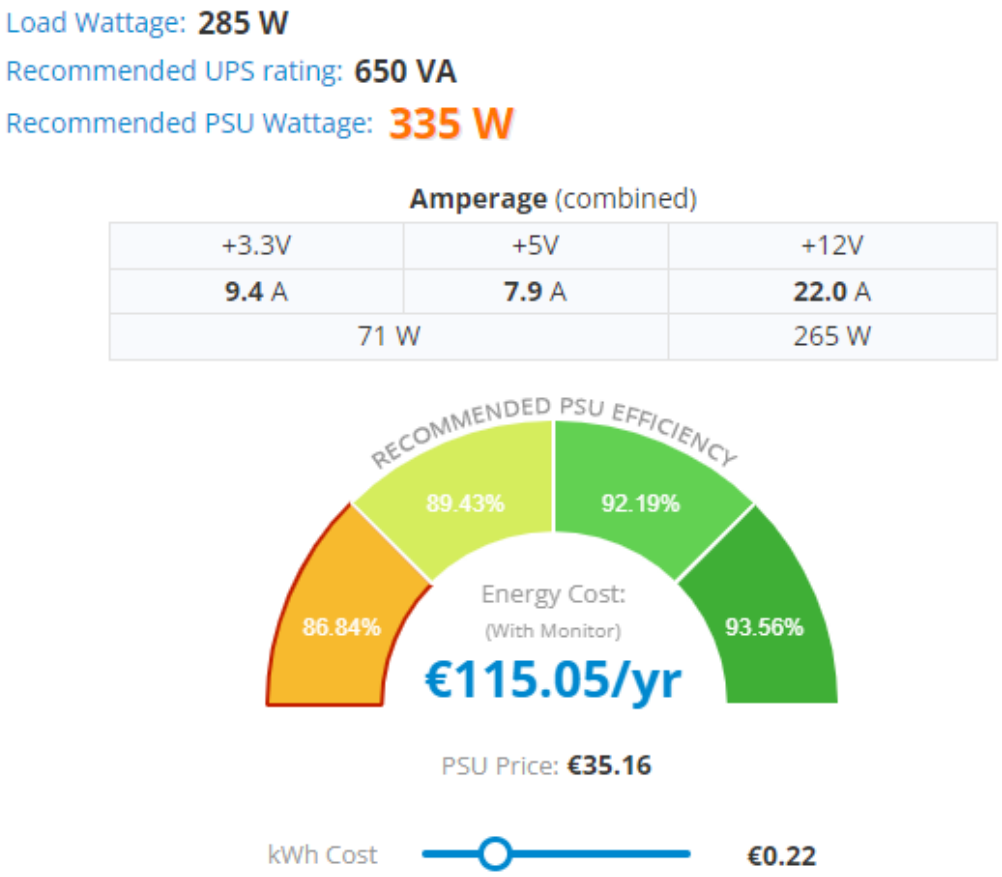


Figura 22 - Consumo energético en kWh del equipo de desarrollo

La carga estimada es de **285 W**. Teniendo en cuenta las horas de trabajo consideradas para el coste/hora, el consumo total en kWh del desarrollo del proyecto puede calcularse como:

(Consumo en W x nº de horas) / 1.000 = Coste total en kWh

Aplicando los valores:

(285 x 444) / 1.000 = 126,54 kWh

El consumo total del proyecto ha sido de **126,54 kWh**. Para su transformación en huella de carbono, utilizaremos otra herramienta online que estima esta equivalencia [24]. Utilizaremos la aproximación pesimista, sin considerar posibles contrataciones de energía verde:

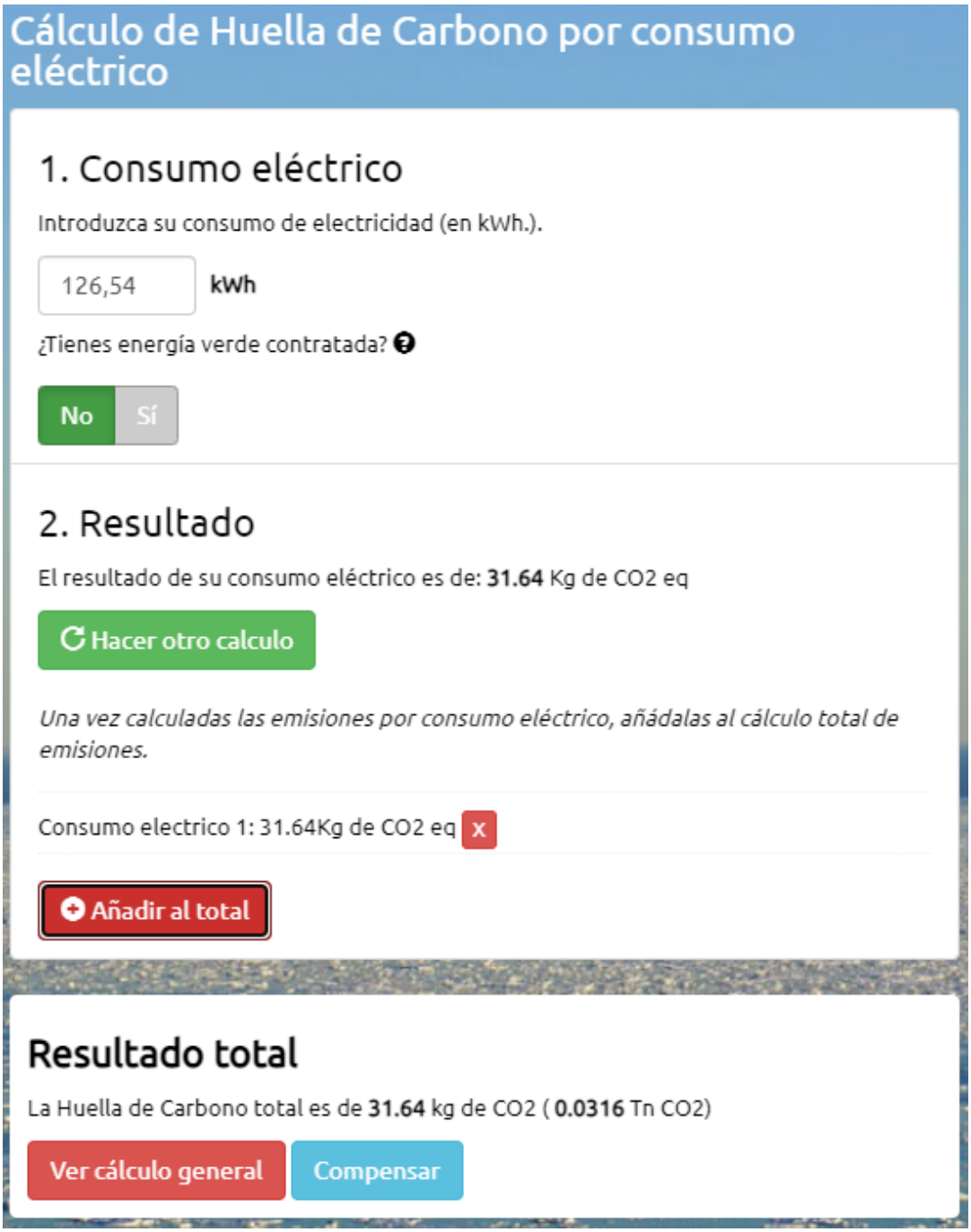


Figura 23 - Cálculo del impacto en CO2 del equipo de desarrollo

El desarrollo del proyecto ha supuesto la emisión de **31,64 kg de CO2**.

6.2.3 Viabilidad del proyecto

La idea de partida del proyecto nace de una carencia actual en el sector del videojuego manifestada por su propio sector objetivo, el de las personas con movilidad limitada o reducida. De acuerdo con los datos proporcionados por la Confederación Española de Asociaciones de Atención a las Personas con Parálisis Cerebral (ASPACE) en España existen alrededor de **120.000 personas** con esta afectación [25].

Considerar a todos y cada uno de los integrantes de este nicho comercial como un comprador potencial del videojuego supondría unas perspectivas poco realistas; incluso en ese supuesto estaríamos considerando unas expectativas alejadas de las cifras de ventas manejadas por los títulos AAA y más próximas a las de un juego *indie*. Desde esa perspectiva, según la plataforma de análisis de mercado Video Games Insights más del 50% de los videojuegos indies nunca consiguen ganar más de 4000$ durante todo su ciclo de vida [26], por lo que teniendo en cuesta un coste estimado de desarrollo de 6.464,64€ (que no implica costes indirectos como publicidad o coberturas asociadas a contratación) obtenemos unas previsiones a priori deficitarias.

El modo más viable de llevar a cabo el videojuego del presente trabajo como un desarrollo profesional sería su presentación como idea en alguna de las convocatorias anuales a fondos de ayuda destinados a proyectos sociales, bajo la perspectiva de cómo este tipo de ocio puede contribuir a la ludificación de las terapias, la expansión personal en el tiempo libre y la mejora en el bienestar emocional. Algunas de estas iniciativas anuales son:

- Convocatoria estatal a cargo a los fondos del 0,7% del IRPF.
- Convocatorias sociales de Fundación CaixaBank.
- Convocatoria de ayudas Fundación Solidaridad Carrefour.
- Ayudas a proyectos sociales Fundación Mapfre.

7. Conclusiones

7.1 Evolución del proyecto y discusión

Como se ha mencionado varias veces a lo largo de la documentación, el objetivo principal del proyecto era demostrar cómo pueden adaptarse ciertos géneros de videojuegos a un control que utilice exclusivamente seguimiento ocular, por ser éste el único método de control al que muchas personas tienen acceso por motivos de limitaciones físicas o diversidad funcional. Se ha realizado tomando como caso un título del género de conducción por ser una petición transmitida en primera persona al equipo de desarrollo.

Repasaremos a continuación los objetivos planteados al inicio, argumentando para cada uno en qué medida han conseguido completarse:

1) Implementación de la interfaz de control mediante Tobii, que a su vez se divide en:

1-1) **Interacción en las escenas de juego:** El control del vehículo se realiza únicamente mediante reconocimiento ocular sin necesidad de ningún otro modo de control o periférico.

1-2) **Interacción con menús y componentes secundarios:** Una vez se haya conseguido una jugabilidad viable, se puede complementar la experiencia con pantallas de título o menús de selección de opciones que también tendrán que ser controlables exclusivamente con el seguimiento ocular.

2) Desarrollo de los componentes y la lógica de las escenas de juego, atendiendo a:

2-1) **Vehículo controlado por el jugador, atendiendo:** El flujo del juego transcurre como consecuencia de la interacción del vehículo del jugador con el resto de componentes, ya sea sumando puntós por recol·lectar objetos, alcanzando checkpoints invisibles al jugador para activar o desactivar eventos, mostrando mensajes al alcanzar la meta, etc

2-2) **Movimiento semiguiado del jugador por el entorno:** La aceleración aplicada a las ruedas que impulsa el movimiento a lo largo del trazado del circuito se realiza de manera automática, sin intervención del jugador. En manos de éste queda determinar el encaramiento del vehículo mediante los paneles que gobiernan el eje de giro de las ruedas delanteras.

2-3) **Puntos de ruta:** La disposición de los objetos en el circuito y el movimiento mediante IA de los vehículos rivales se realiza

automáticamente por medio de una serie de puntos de ruta prefijados.

2-4) **Cámara:** La cámara sigue en todo momento al jugador, controlando mediante scripts que lo haga de modo suave y sin movimiento bruscos.

2-5) **Objetos y puntuación:** En el modo de juego por puntos, se sitúan grupos de objetos en posiciones prefijadas a lo largo del circuito. Cada grupo se compone de un objeto positivo, un objeto negativo y un hueco distribuidos aleatoriamente, con la excepción del grupo inicial frente a la línea de salida que siempre presenta el objeto positivo frente al jugador.

2-6) **Vehículos adversarios:** En el modo de juego contra rivales se han implementado dos vehículos controlados por IA que siguen una serie de puntos de ruta para intentar completar tres vueltas antes que el jugador, variando las condiciones de victoria con respecto al modo por puntuación.

2-7) **Elementos adicionales:** La interfaz incorpora un cronómetro y un menú de pausa/opciones durante la carrera. Para el modo de juego con rivales, además, existe un espejo retrovisor y un contador de vueltas.

3) Presencia, estética, fluidez y entorno *user-friendly*, que complementen la experiencia jugable con una presentación completa y agradable, comprendiendo:

3-1) **Construcción de pantalla de título y / o presentación para no comenzar la experiencia jugable abruptamente:** La transición entre pantallas y modos de juegos se suaviza mediante cortinas de fundidos a negro.

3-2) **Entorno gráfico:** Los colores de los objetos, botones y vehículos se han establecido atendiendo a criterios de accesibilidad en cuanto a combinaciones de colores. Se ha procurado que el tamaño de los elementos interactivos sea lo suficientemente grande y la disposición práctica como para que su uso resulte lo más fácil posible a todo tipo de usuarios, sin que vaya en detrimento de un interfaz eficiente.

3-3) **Música y efectos de sonido:** Los diferentes estados de la carrera (en trascurso o finalizada) se denotan con diferentes melodies de fondo. La interacción con objetos y elementos de interfaz se ha reforzado con efectos de sonido.

3-4) **Menús de configuración o selección modos de juego:** la pantalla del título, menú de pausa durante la carrera y menú de finalización de carrera disponen de varios elementos interactivos con los que seleccionar escena o modo de juego deseado.

3-5) **Opción de volver a la pantalla principal o jugar de nuevo:** incorporados en los menús de pausa y el menú de fin de carrera.

La jugabilidad se planteó inicialmente de modo que el vehículo principal recorriese el circuito siguiendo un camino completamente prefijado, con el trazado dividido en carriles discretizados entre los que el jugador podía elegir donde situarse hasta llegar a la meta; el objetivo era suplir las limitaciones que el seguimiento ocular conlleva en comparación con respecto al carácter analógico de los controles tradicionales (como un volante o gamepad) utilizados en el género de conducción. Si bien esta decisión lograba el objetivo, limitaba las posibilidades de la jugabilidad y presentaba desventajas a la hora de un seguimiento de cámara fluido o planificar el comportamiento de los vehículos rivales; por ello se modificó esta base para mantener el desplazamiento automático en cuanto a velocidad de desplazamiento, pero permitiendo un giro libre del eje delantero del vehículo principal que otorgue más libertad de decisión al jugador.

El cambio de paradigma obligó a reestructuraciones no sólo en el manejo del vehículo principal sino también en el propio circuito, ya que un movimiento libre obligaba a establecer mecanismos de control para que el vehículo principal no abandonase el trazado del circuito previsto; por otro lado, simplificaba el diseño del nivel al suprimir la necesidad de implementar puntos de ruta para guiar el movimiento del jugador sin que el esfuerzo invertido a este respecto fuese en vano, ya que el sistema se reutilizaría para los vehículos rivales. Si bien puede parecer que es una solución que debía haberse adoptado desde el principio, fue una conclusión que llegó como consecuencia de explorar las posibilidades del seguimiento ocular durante el transcurso del proyecto.

Si bien el proyecto desarrollado presenta limitaciones y existe mucho margen de mejora, se considera demostrada la premisa de que el seguimiento ocular es una opción viable para acercar determinados géneros de videojuegos, como puede ser la conducción, a personas que por determinadas circunstancias no tienen acceso a los títulos tradicionalmente desarrollados.

7.2 Puntos de mejora y líneas de trabajo futuro

La experiencia propuesta en el proyecto puede mejorarse expandiendo características ya presentes o introduciendo otras nuevas. Algunas de estas ideas resultarían imprescindibles si se diese el paso a una propuesta comercial, pero quedan fuera del trabajo presentado por restricciones en la ventana de tiempo disponible y / o requerir una inversión económica profesional:

- Adaptar el funcionamiento a una gama más amplia de sensores de seguimiento ocular, no solamente teniendo en cuenta modelos más sofisticados de la gama de productos Tobii sino también otras marcas.
- Permitir al jugador elegir vehículos con diferentes características, como velocidad, suavidad del giro o tipo de tracción.

- Más circuitos que representen distintos niveles de dificultad.
- Físicas revisadas que permitan maniobras de conducción avanzada, como por ejemplo derrapes.
- Condiciones climáticas que modifiquen el comportamiento de los controles.
- Diferentes momentos del día y / o noche con varios estados de iluminación.
- Menús de opciones para personalizar interfaz, selección musical, aspecto de los paneles de control, etc.
- Creación de perfiles de jugador, con guardado de los registros que permita implementar tablas de puntuación.
- Aumentar el número de vehículos rivales.
- Implementar más modos de juego, como carrera contrarreloj o modo espejo en que debe circularse en sentido contrario al habitual.
- Diseño de elementos de audio y vídeo creados específicamente para el videojuego en lugar de utilizar *assets* genéricos de terceros.

8. Glosario

- *Asset:* Cualquier fichero o archivo perteneciente al videojuego.
- *Canvas:* Área sobreimpresa a la escena en la que se sitúan todos los elementos de interfaz o U.
- *Carrera contra rivales:* Modo de juego en el que el jugador debe completar tres vueltas al circuito antes de que lo logren otros vehículos rivales controlados por una Inteligencia Artificial.
- *Carrera por puntuación:* Modo de juego en el que el jugador debe intentar conseguir el mayor marcador de puntuación posible recogiendo objetos positivos y esquivando objetos negativos.
- *Circuito*: El trazado del escenario en el que transcurre la carrera, delimitado por paredes y con textura de suelo de asfalto.
- *Cursor*: El puntero sobreimpreso en pantalla que corresponde a la posición de los ojos captada por el seguimiento ocular.
- *Escena*: Cada uno de los niveles o pantallas en las que se subdivide un videojuego en Unity, que contienen los objetos, entornos y menús del juego. Se crean con una cámara y una luz direccional por defecto
- *Eye-tracking*: Véase *seguimiento ocular*.
- *Grupo de objetos*: Conjunto formado por un objeto positivo, un objeto negativo y un hueco vacío que se ubican en la posición marcada por un punto de ruta.
- *Menú de fin de carrera*: Menú que aparece durante la carrera cuando se cumplen las condiciones para darla por finalizada. Interrumpe el transcurso del juego y ofrece dos opciones: reiniciar la carrera desde el principio o salir al menú principal
- *Menú de pausa*: Menú que puede desplegarse durante la carrera. Interrumpe el transcurso del juego y ofrece tres opciones: continuar sin cambios, reiniciar la carrera desde el principio o salir al menú principal
- *Menú principal*: La pantalla con el título del juego que presenta las opciones principales: jugar en modo de carrera por puntos, jugar en modo de carrera contra rivales, o salir de la aplicación.
- *Objetos positivos*: Objetos presentes en el interior del circuito en el modo de carrera por puntuación. Suman puntos en el marcador cuando el vehículo principal colisiona con ellos.
- *Objetos negativos*: Objetos presentes en el interior del circuito en el modo de carrera por puntuación. Restan puntos en el marcador cuando el vehículo principal colisiona con ellos.
- *Pantalla del título*: Véase *Menú principal.*

- *Puntos de ruta*: Marcadores de posición no visibles que determinan coordenadas preestablecidas. Se utilizan para guiar el movimiento de los vehículos rivales y ubicar los grupos de objetos.
- *Seguimiento ocular*: Tecnología consiste en un receptor / emisor que se coloca en el monitor y permite seleccionar objetivos situados en pantalla.
- *Script*: Un asset con fragmento de código en C# que se adjunta a un elemento de la escena para implementar lógica.
- *Vehículos adversarios*: Véase *vehículos rivales*.
- *Vehículo principal*: El vehículo controlado por el jugador que le representa en el entorno de juego.
- *Vehículos rivales*: Los vehículos controlados por una Inteligencia Artificial que compiten contra el jugador en el modo de carrera contra rivales.
- *UI*: Abreviatura de interfaz de usuario, del inglés User Interface. Véase *Canvas*.

9. Bibliografía

1) **El videojuego en España factura 1.747 millones de euros en 2020 y aumenta su base hasta casi los 16 millones de usuarios**.

URL: *https://www.europapress.es/portaltic/videojuegos/noticia-videojuego-espana-factura-1747-millones-euros-2020-aumenta-base-casi-16-millones-usuarios-20210429122400.html*.

Fecha de visita: 03/12/2021

2) Płużyczka, Monika. **The First Hundred Years: a History of Eye Tracking as a Research Method**. Applied Linguistics Papers. Pp 101-116. 4/2018. 2018.

DOI: 10.32612/uw.25449354.2018.4.pp.101-116

3) **A Brief History of Eye-Tracking**.

URL: https://www.uxbooth.com/articles/a-brief-history-of-eye-tracking

Fecha de visita: 03/12/2021

4) **Reading Plus - A History of Innovation**.

URL: https://www.readingplus.co.uk/history

Fecha de visita: 03/12/2021

5) Gibaldi, Agostino & Vanegas, Mauricio & Bex, Peter & Maiello, Guido. **Evaluation of the Tobii EyeX Eye tracking controller and Matlab toolkit for research**. Behavior Research Methods. Pp 923–946. 49/2017. 2017.

DOI: https://doi.org/10.3758/s13428-016-0762-9

6) Białowąs, Sylwester & Szyszka, Adrianna. **Eye-tracking in Marketing Research**. Managing Economic Innovations - Methods and Instruments. Pp 91-104. 2019.

DOI: 10.3758/s13428-016-0762-9

7) Uzma Samadani, Robert Ritlop, Marleen Reyes, Elena Nehrbass, Meng Li, Elizabeth Lamm, Julia Schneider, David Shimunov, Maria Sava, Radek Kolecki, Paige Burris, Lindsey Altomare, Talha Mehmood, Theodore Smith, Jason H Huang, Christopher McStay, S Rob Todd, Meng Qian, Douglas Kondziolka, Stephen Wall, Paul Huang. **Eye tracking detects disconjugate eye movements associated with structural traumatic brain injury and concussion**. Journal of Neurotrauma. Pp. 548-556. Volume: 32 Issue 8. 2015

DOI: 10.1089/neu.2014.3687

8) Kunka, B.; Kosikowski, R.; Barlinn, J. and Kozak, K. **Brain Rehabilitation in Clinical Trials Setup by Eye-Tracking**. Proceedings of the Fifth International Conference on Telecommunications and Remote Sensing. Pp 89-94. 2016.

DOI: 10.5220/0006227500890094

9) **The secret world of disabled gamers**.

URL: https://www.technologyreview.com/2018/07/03/240440/the-secret-world-of-disabled-gamers/

Fecha de visita: 03/12/2021

10) L. Ballaz et al. **Active video games and children with cerebral palsy: the future of rehabilitation?**. International Conference on Virtual Rehabilitation. Pp 1-2. 2011.

DOI: 10.1109/ICVR.2011.5971808.

11) Archambault D., Gaudy T., Miesenberger K., Natkin S., Ossmann R. **Towards Generalised Accessibility of Computer Games**. In: Pan Z., Zhang X., El Rhalibi A., Woo W., Li Y. (eds) Technologies for E-Learning and Digital Entertainment. Pp 518-527. Lecture Notes in Computer Science, vol 5093, Edutainment 2008. 2008.

DOI: 10.1007/978-3-540-69736-7_55

12) **Videojuegos y accesibilidad: esta es la odisea de las personas con discapacidad**.

URL: https://www.3djuegospc.com/reportajes/videojuegos-accesibilidad-esta-odisea-personas-discapacidad

Fecha de visita: 04/12/2021

13) **Young gamers are inventing their own controllers to get around their disabilities**.

URL: https://theconversation.com/young-gamers-are-inventing-their-own-controllers-to-get-around-their-disabilities-75665

Fecha de visita: 04/12/2021

14) **The AbleGamers Charity – sitio web official**.

URL: https://ablegamers.org/

Fecha de visita: 04/12/2021

15) **SpecialEffect – sitio web official**.

URL: https://www.specialeffect.org.uk/

Fecha de visita: 04/12/2021

16) **Xbox Adaptive Controller – página de especificaciones oficial**.

URL: https://www.xbox.com/es-ES/accessories/controllers/xbox-adaptive-controller

Fecha de visita: 04/12/2021

17) **Tobii Eye Tracking enabled games**.

URL: https://gaming.tobii.com/games/

Fecha de visita: 04/12/2021

18) **Captura de Wipeout HD (By Eurogramer, Fair use)**.

URL: https://en.wikipedia.org/w/index.php?curid=53099770

Fecha de visita: 04/12/2021

19) **Diseñar con el daltonismo en mente, mejorará el diseño para todos**.

URL: https://medium.com/@invisionenespanol/dise%C3%B1ar-con-el-daltonismo-en-mente-mejorar%C3%A1-el-dise%C3%B1o-para-todos-2f2e35a01744

Fecha de visita: 28/11/2021

20) **Unity Personal – Página de especificaciones**.

URL: https://store.unity.com/es/products/unity-personal

Fecha de visita: 04/12/2021

21) **Instituto Nacional de Estadística**.

URL: https://www.ine.es/index.htm

Fecha de visita: 08/12/2021

22) **Salario medio para Ingeniero Técnico en España, 2021**.

URL: https://es.talent.com/salary?job=Ingeniero%20T%C3%A9cnico

Fecha de visita: 08/12/2021

23) **OuterVision® Power Supply Calculator**.

URL: https://outervision.com/power-supply-calculator

Fecha de visita: 08/12/2021

24) **Cero CO2 – Calculadoras de emisiones de Gases de Efecto Invernadero (GEI)**.

URL: https://www.ceroco2.org/calculadoras/electrico

Fecha de visita: 08/12/2021

25) **ASPACE – Parálisis cerebral – Algunos datos**.

URL: https://aspace.org/algunos-datos

Fecha de visita: 08/12/2021

26) **Infographic: Indie game revenues on Steam**

https://vginsights.com/insights/article/infographic-indie-game-revenues-on-steam

Fecha de visita: 08/12/2021

10. Anexos

10.1 Anexo I – Créditos de recursos externos

Los elementos gráficos y de sonido utilizados en el proyecto corresponden a recursos gratuitos que pueden ser empleados en proyectos no comerciales y/o mediante la debida acreditación:

- **Textura de Asfalto del circuito**

  Título: Asphalt materials

  Creador: VK GameDev

  Fuente: https://assetstore.unity.com/packages/2d/textures-materials/roads/asphalt-materials-141036

  Licencia: Standard Unity Asset Store EULA

- **Objetos positivos y negativos**

  Título: Simple Gems Ultimate Animated Customizable Pack

  Creador: AurynSky

  Fuente: https://assetstore.unity.com/packages/3d/props/simple-gems-ultimate-animated-customizable-pack-73764

  Licencia: Standard Unity Asset Store EULA

- **Modelos de vehículos**:

  Título: Lowpoly Cars: Amazing car pack for mobiles

  Creador: Fun Assets

  Fuente: https://assetstore.unity.com/packages/3d/vehicles/lowpoly-cars-amazing-car-pack-for-mobiles-194643

  Licencia: Standard Unity Asset Store EULA

- **Efectos de partículas**

  Título: Cartoon FX Free

  Creador: Jean Moreno (JMO)

  URL de descarga: https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565

  Licencia: Standard Unity Asset Store EULA

- **FX de entrada de título**

Título: Tires Squealing
Creador: Mike Koenig
Fuente: https://soundbible.com/1178-Tires-Squealing.html
Licencia: CC Attribution 3.0

- **FX de Confirmación de opción seleccionada**
  Título: HONDA CR-V START
  Creador: Blake N. (xtrgamr)
  Fuente: https://freesound.org/people/xtrgamr/sounds/253614/
  Licencia: CC Attribution 3.0

- **FX de Cuenta atrás**
  Título: Race Start Countdown
  Creador: Justin Couch (JustInvoke)
  Fuente: https://freesound.org/people/JustInvoke/sounds/446130/
  Licencia: CC Attribution 3.0

- **FX de Iniciar carrera**
  Título: Race Start
  Creador: Justin Couch (JustInvoke)
  Fuente: https://freesound.org/people/JustInvoke/sounds/446142/
  Licencia: CC Attribution 3.0

- **Música durante la carrera**
  Título: High Five
  Creador: PlayOnLoop
  Fuente: https://www.playonloop.com/2016-music-loops/high-five/
  Licencia: CC Attribution 4.0

- **Música durante el fin de la carrera**:
  Título: Living Lucky
  Creador: PlayOnLoop
  Fuente: https://www.playonloop.com/2015-music-loops/living-lucky/
  Licencia: CC Attribution 4.0

10.2 Anexo II – Manual de usuario

10.2.1 Requisitos mínimos

- Sistema operativo Windows 7 (SP1+) x64, Windows 10 x64 o Windows 11 x64.
- CPU con arquitectura x64 con soporte para instrucciones de tipo SSE2.
- GPU: Tarjeta gráfica compatible con DX10, DX11 y DX12.
- 150 MB de espacio disponible en disco duro.
- Dispositivo de seguimiento ocular Tobii.

10.2.2 Menú principal / pantalla del título

Menú principal / Pantalla del título

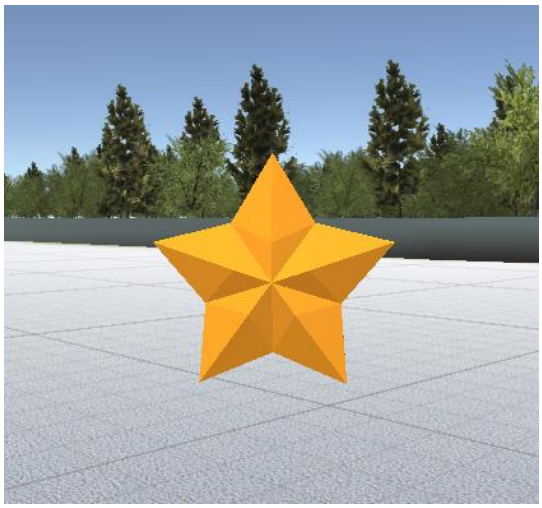
En la pantalla del título se ofrecen tres botones, correspondientes a las opciones para elegir cada uno de los dos modos de juegos disponibles y para salir de la aplicación:

- Carrera – Puntos: Inicia el modo de carrera por puntuación.
- Carrera – Rivales: Inicia el modo de carrera contra rivales.
- Salir: Cierra el juego y vuelve al sistema operativo.

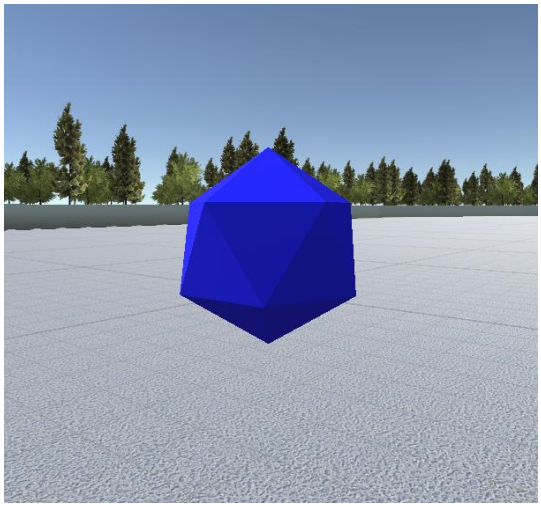
Para seleccionar una de las opciones, debe situarse el cursor de seguimiento ocular sobre el botón que correspondiente y esperar hasta que se complete la barra de progreso circular que aparecerá.

10.2.3 Modo de carrera por puntuación

En este modo, el jugador debe completar una vuelta al circuito mientras recoge las estrellas que hacen aumentar su puntuación y esquiva las trampas que la disminuyen.

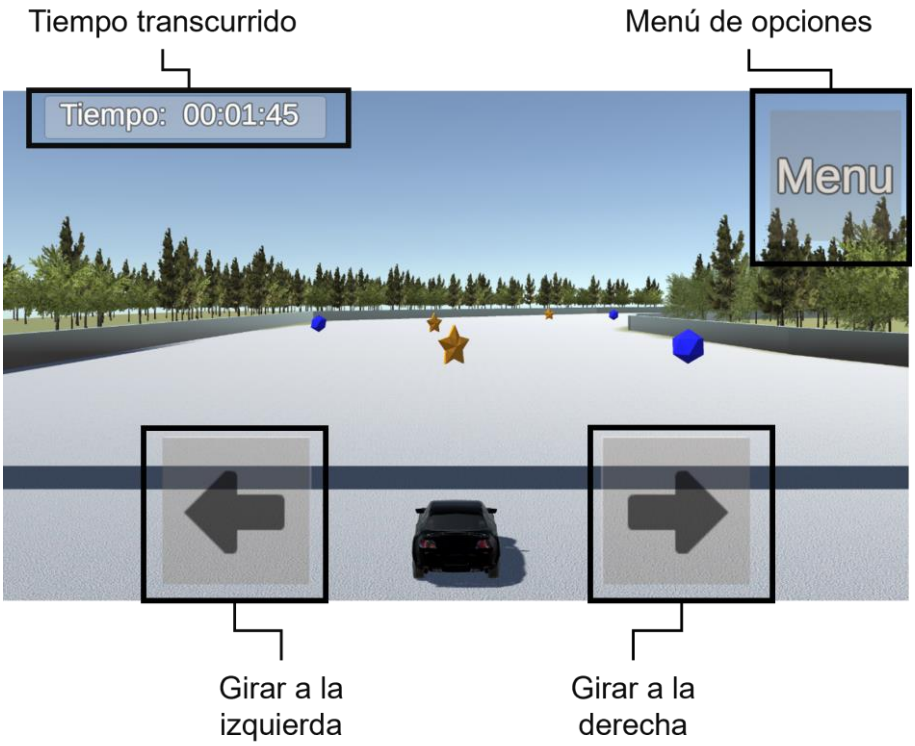


Objeto estrella

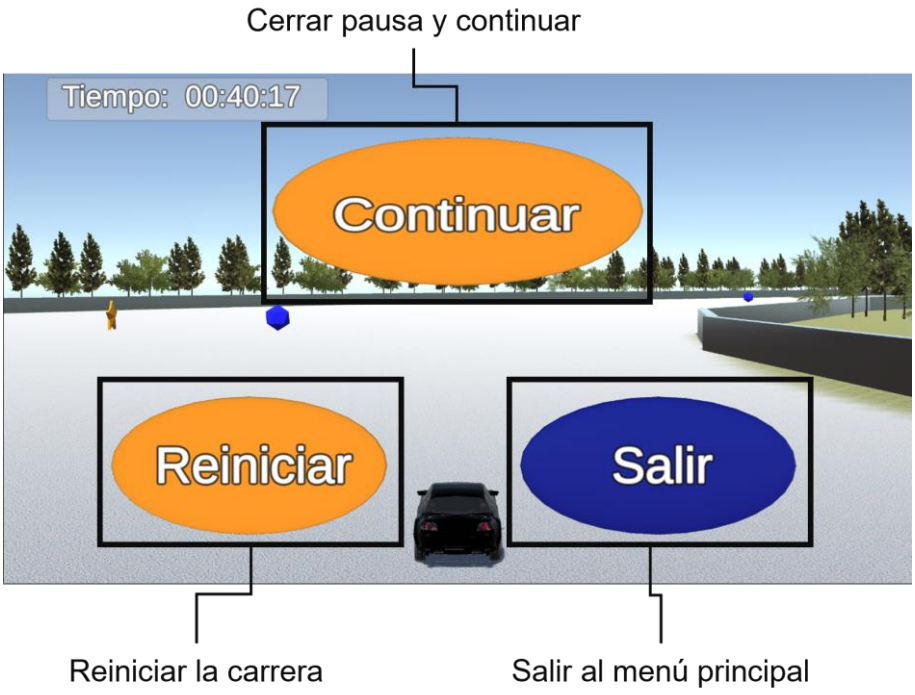


Objeto trampa

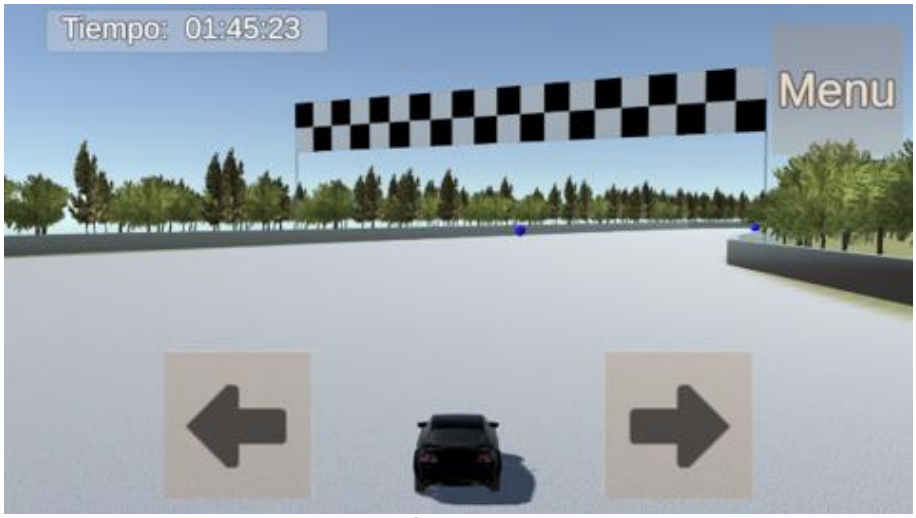
El vehículo del jugador acelera y avanza automáticamente. Para variar el giro del eje delantero que determina el encaramiento, debe situarse el cursor de seguimiento ocular sobre el panel correspondiente al sentido deseado de entre los situados en la parte inferior de la pantalla.



En cualquier momento, puede pausarse la partida y abrir un menú de opciones si se mantiene el seguimiento ocular sobre el panel situado en la esquina superior derecha de la pantalla. En este nuevo menú puede reiniciarse la carrera desde el principio, salir a la pantalla del título o reanudar la carrera de nuevo, manteniendo el seguimiento ocular sobre el botón correspondiente hasta que se rellene la barra de progreso circular.



Una vez se llegue al final del circuito, delimitado por la bandera de meta, aparecerá la pantalla con el marcador de puntuación final. Aquí será posible reiniciar la carrera desde el principio o volver al menú principal.



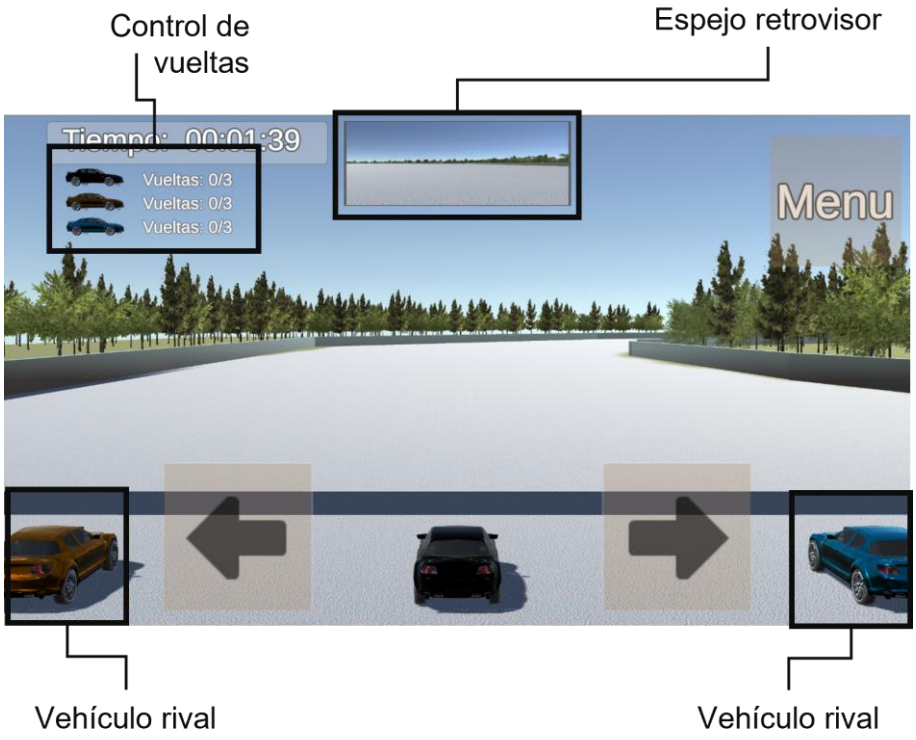
Aproximándose a la meta

Reiniciar la carrera

Salir al menú principal

10.2.4 Modo de carrera contra rivales

En esta modalidad, el objetivo es completar tres vueltas completas al circuito antes de que lo haga alguno de los vehículos rivales presentes. Los controles y el menú de opciones disponibles son los mismos que en el modo de carrera por puntuación.



Cuando alguno de los vehículos participantes haya logrado completar las tres vueltas, aparecerá la pantalla de fin del juego. De nuevo, igual que en el modo de juego por puntuación, se podrá elegir entre reiniciar la carrera o salir al menú principal.