**Agradecimientos**

Me gustaría agradecer este TFG a mis padres, ya que sin ellos no hubiese conseguido acabarlo. También a mi profesor Alberto, por aguantarme tantos años con el TFG a rastras.

**Resumen**

En este TFG se ha implementado una Aplicación Web con la que se podrá visualizar, recoger datos y probar combinaciones sobre un robot móvil con direccionamiento diferencial, cuando trata de seguir un circuito introducido por el usuario.

Para ello, se ha desarrollado la mecánica de este sistema a través de las ecuaciones que relacionan el giro de los motores con la posición y orientación de nuestro vehículo. A continuación, se ha analizado el entorno tecnológico sobre el que se trabajará, decidiendo un entorno Web.

Posteriormente, se han implementado dichas ecuaciones en lenguaje JavaScript y se ha procedido al modelado del objeto y el entorno. Finalmente, se han añadido distintas funcionalidades como la posibilidad de poder guardar los resultados de las pruebas, modificar el vehículo o cargar circuitos de manera dinámica.

# **Introducción**

## **Motivación**

Este trabajo nace motivado por un intento de mejorar una de las herramientas utilizadas en la asignatura “Robótica” de la titulación “Ingeniería Técnica en Informática de Sistemas” impartida en el antiguo Centro de Estudios Superiores Felipe Segundo, que actualmente constituye el Campus de Aranjuez de la Universidad Rey Juan Carlos.

Basado en la siguiente herramienta (MATLAB-Simulink), se pensó en mejorar la herramienta en un entorno Web, que se pudiera utilizar desde cualquier sitio. Además, al realizarlo con un lenguaje de programación tan amplio como JavaScript, en consonancia con WebGL, nos permitía realizar un modelo tridimensional del objeto y que el usuario tuviese diversos métodos para recoger datos.





## **Objetivos**

El principal objetivo es llegar a conseguir una visualización 3D del vehículo, el cual se podrá editar de manera simple con unos campos de texto. Esto afectará tanto al movimiento como a la capacidad de seguir el circuito.

Podremos manipular el vehículo de una manera sencilla utilizando solamente el teclado y el ratón.

También queremos poder visualizar el robot desde distintos puntos, por lo que necesitaremos diferentes tipos de cámaras: Ortogonal y Automática.

Además, podremos girar la cámara automática alrededor del vehículo.

Podremos añadir un circuito con un botón y un fichero de texto. Aparecerá un botón cuando añadamos el circuito que nos permitirá mover el coche automáticamente hasta la posición de salida y comenzará a contar número de toques y tiempo de vuelta.

Podremos registrar el número de toques y el tiempo de vuelta de todas las vueltas que queramos.

## **Estado del arte**

Actualmente en el mercado podemos encontrar una gran variedad de robots programados con Arduino que nos permiten llevar a cabo estos objetivos, pero cuestan dinero y hay que montarlos y programarlos. En cambio, si queremos encontrar una aplicación que nos resuelva este objetivo, cuesta encontrarla.

Además de programarla con Javascript y WebGL, podríamos llegar a utilizar otros lenguajes de programación que nos permitan diseñar y desarrollar visualizaciones tridimensionales, como por ejemplo Unity o OpenGL.

## **Estructura de la memoria**

En el Capítulo 2: Robótica móvil, comenzaremos explicando cómo se definen los vehículos con ruedas y que tipos o configuraciones de vehículos existen. Pasaremos a una explicación mediante el uso de fórmulas de como funciona nuestro vehículo en particular, como podemos calcular el direccionamiento de éste. Por último, comentaremos como podemos definir una navegación autónoma para nuestro robot.

En el capítulo 3: Entorno tecnológico explicaremos los lenguajes de programación que hemos utilizado, el porque y sus puntos fuertes y débiles. También explicaremos como funciona WebGL y la aplicación que le daremos en nuestro proyecto. En último lugar explicaremos como funciona GitHub y el uso que le hemos dado.

En el capítulo 4: Descripción de la aplicación explicaremos detalladamente cada zona de nuestra aplicación, tanto nuestro panel izquierdo donde veremos el robot, como las zonas de información y datos. También podremos observar los casos de uso de nuestra aplicación, con varios ejemplos de funcionamiento.

# **Robótica móvil**

## **Vehículos con ruedas**

“Los vehículos con ruedas son la solución más simple y eficiente para conseguir la movilidad en terrenos suficientemente duros y libres de obstáculos, permitiendo conseguir velocidades relativamente altas.

Como limitación más significativa cabe mencionar el deslizamiento al impulsar. Es posible que, dependiendo del terreno, aparezcan deslizamientos y vibraciones que desvíen la trayectoria del vehículo. La locomoción mediante ruedas es poco eficiente en terrenos blandos.

Por otra parte, excepto en configuraciones muy especiales, no es posible alterar internamente el margen de estabilidad para adaptarse a la configuración del terreno, lo que limita de forma importante los caminos aceptables del soporte.

Los robots móviles emplean diferentes tipos de locomoción mediante ruedas que les confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad. La mayor maniobrabilidad se consigue en vehículos omnidireccionales. Un vehículo omnidireccional en el plano es capaz de trasladarse simultánea e independientemente en cada eje del sistema de coordenadas y rotar según el eje perpendicular.”

## **Diferentes configuraciones**

El guiado diferencial es tan solo una entre las varias configuraciones existentes para los vehículos con ruedas. Puede haber diferentes configuraciones de vehículos con ruedas:

* Ackerman: Utilizado en vehículos de cuatro ruedas convencionales. Las dos ruedas delanteras son las que giran, siendo la rueda delantera interior la que gira en un ángulo ligeramente superior para evitar deslizamientos.
* Triciclo: Vehículo de tres ruedas, de las cuales la delantera proporciona la dirección y la tracción. Las dos ruedas traseras solamente proporcionan estabilidad, ya que giran libremente.
* Direccionamiento diferencial: Nuestro vehículo usará esta configuración. El direccionamiento viene dado por la diferencia de velocidades de las ruedas laterales, que también poseen la tracción.
* Skid Steer: Se disponen varias ruedas en cada lado del vehículo que actúan de forma simultánea. El movimiento es el resultado de combinar las velocidades de las ruedas de la izquierda con las de la derecha.
* Pistas de deslizamiento: Son vehículos tipo oruga en los que tanto la impulsión como el direccionamiento se consiguen mediante pistas de direccionamiento.
* Síncronas: Consiste en la actuación simultánea de todas las ruedas, que giran de forma síncrona. La transmisión se consigue mediante coronas de engranajes o con correas concéntricas.

## **Direccionamiento diferencial**

**Modelo cinemático**

Tal y como se muestra en la Figura 1, supóngase un sistema de referencia {G} y un sistema {L} con origen en el punto de guiado del vehículo y eje ZL en la dirección del eje longitudinal del vehículo.



. Modelo cinemático de un robot móvil

Entonces, si el vehículo tiene una velocidad de desplazamiento y de rotación con respecto a {L}, con respecto a {G} la velocidad es:

Así, si se conoce y , se podría calcular la trayectoria del vehículo (cinemática directa).

A la inversa, si se desea que el vehículo siga una determinada trayectoria se podría calcular cual debe ser la velocidad de las ruedas y el giro (cinemática inversa).

**Modelo discreto**

La derivada de una función puede aproximarse por el cociente incremental:

Esto se conoce como derivada discreta hacia adelante, pero también puede aproximarse con el cociente incremental hacia atrás, la aproximación centrada, u otras aproximaciones más complicadas.

El resultado es que, con una ecuación de este tipo, la nueva coordenada x (en t+1) se puede calcular a partir de la anterior (en t) mediante la ecuación:

Aplicando el mismo resultado al resto de coordenadas del modelo cinemático directo, se obtiene:

Por tanto, si se dispone de las coordenadas en cada instante de un determinado horizonte temporal, se puede calcular la nueva posición y orientación del robot en dicho horizonte.

El direccionamiento diferencial viene dado por la diferencia de velocidades de las ruedas laterales. La tracción se consigue con esas mismas ruedas. Adicionalmente, existen una o más ruedas para el soporte. En la figura 2 se muestra una imagen de dicho esquema. Nótese que para especificar la configuración hay que indicar los valores de las tres variables (x, z, ), siendo las variables de control son las velocidades de las ruedas laterales.



. Locomoción mediante guiado diferencial

Sean y las velocidades de giro de las ruedas izquierda y derecha, respectivamente. Si el radio de la rueda es , las velocidades lineales correspondientes son y . En este caso, la velocidad lineal y la velocidad angular correspondientes en el modelo vienen dadas por:

Sustituyendo estas expresiones en las obtenidas a partir de la Figura 1, se obtienen las velocidades de las coordenadas del robot en el sistema {G} a partir de la velocidad de giro de cada rueda:

Finalmente, utilizando el modelo discreto, se obtiene:

## **Navegación autónoma**

Con este vehículo, pretendemos realizar una navegación autónoma mediante el uso de unos sensores infrarrojos CNY70, los cuales detectan la claridad del objeto que hay bajo ellos.

Nuestro circuito será de color negro sobre fondo blanco, por lo que mediante estos sensores podemos detectar en qué momento nuestro vehículo se sitúa encima del circuito y frenar una de sus ruedas en consecuencia.

El vehículo avanzará hasta que uno de los sensores detecte un cambio de claridad. En ese momento, la rueda del mismo lado que el sensor que está detectando el cambio se frenará, proporcionando un giro al vehículo en la dirección que sigue el circuito.

En nuestra aplicación la colisión se calculará en cada momento mediante la diferencia de las posiciones de cada sensor con todos los puntos del circuito. Siendo la posición de un sensor (sx, sz) y el punto del circuito con el que calcular la distancia (px, pz) se calcula con la siguiente ecuación:

Cuando la distancia *d* obtenida sea menor que el radio del sensor, entonces se tomará como que el sensor ha detectado colisión con la línea del circuito, frenando la rueda necesaria.

# **Entorno tecnológico**

## **HTML/CSS/JS**

En este caso, hemos utilizado JavaScript para la parte de la interfaz gráfica porque nos permite incluir librerías que están ya implementadas (Por ejemplo, WebGL), además de ser un lenguaje de programación con una potencia increíble al ser dinámico y orientado a objetos. Nos permite crear objetos y modificarlos sobre la marcha, pudiendo extenderlos de una manera muy sencilla.

Además, JavaScript nos permite utilizar la interfaz gráfica usando la máquina del usuario, por lo que no colapsa un servidor con cálculos tan extensos como los necesarios para todo un sistema gráfico.

HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Corresponde con las siglas de *HyperText Markup Language*, es decir, Lenguaje de Marcas de Hipertexto. Estas marcas son mayormente conocidas como etiquetas, las cuales nos permiten ordenar nuestro texto con una estructura básica.



El término CSS corresponde a las siglas en inglés de “Hojas de Estilo en Cascada”. CSS es un lenguaje de diseño gráfico para estructurar y presentar un documento definido por un lenguaje de marcado como puede ser HTML. Utiliza clases que se añaden a las etiquetas HTML, que mediante un conjunto de reglas dan el diseño a la página Web.

JavaScript es un lenguaje de programación interpretado, que se usa principalmente en su forma del lado del cliente (client-side), es decir, que se ejecuta en la máquina del cliente. Actualmente todos los navegadores modernos soportan JavaScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Nos permite ampliar la funcionalidad de las páginas Web con animaciones, cálculos o modificaciones en la estructura.

En nuestra aplicación tendremos botones con diversas funcionalidades como cambiar la cámara, añadir un circuito o guardar los cambios modificados. Éstos funcionan mediante un método que escuchará los eventos de click en ellos, es decir, el método solo se ejecutará en el momento en el que se haga click sobre un botón.

Tendremos varios campos de texto numéricos que nos permitirán introducir los valores que deseemos para modificar la estructura de nuestro vehículo.

También podemos utilizar el teclado para usar la aplicación Web. Podremos acelerar o frenar el vehículo con las teclas *W* o *S*, respectivamente y podremos frenar solamente una de las dos ruedas delanteras con las teclas *A* o *D*.

La información recogida la podremos encontrar en unas etiquetas y una lista que se sitúan en la parte inferior de nuestra página Web.

## **WebGL**

En la Web podemos encontrar varias API’s disponibles para la visualización de gráficos por computador. Un ejemplo son las siguientes API’s:

* Chart.js: Api que permite visualizar conjuntos de datos mediante el uso de gráficos de tablas o mapas de puntos, con animaciones y preparado para utilizarse incluso en móviles.
* ArcGIS: Api diseñada para visualizar mapas en las aplicaciones Web y poder utilizarlos para visualizar datos en ellos.

Para visualizar gráficos renderizados por ordenador tenemos la API WebGL. Es una API basada en OpenGL ES 2.0 que permite llevar a cabo la representación 2D y 3D en un elemento canvas HTML. La representación se realiza mediante código JavaScript donde se describen tanto los objetos que vamos a representar como los cambios físicos que se van a aplicar en ellos (Translación, Rotación y Escalado).

Además del código JavaScript, se utiliza un código en lenguaje GLSL para implementar los shaders, que es un código de sombreado que permite simular la iluminación de focos, luces e iluminación ambiente al impactar sobre distintos objetos. Este código es ejecutado para cada vértice enviado a través de la API y para cada píxel rasterizado en pantalla. La implementación de WebGL compila estas instrucciones shader en código GPU, por lo que aumenta la velocidad de renderizado cuanta más potencia gráfica tengas.

La API de WebGL ha sido implementada por el consorcio de tecnología sin ánimo de lucro Kronos Group, los cuales consolidaron el WebGL Working Group. La versión 1.0 de las especificaciones WebGL fue lanzada en marzo de 2011. En 2013 comenzaron el desarrollo de las especificaciones de WebGL 2.0, las cuales están basadas en OpenGL ES 3.0. Concluyeron en enero de 2017.

Para usar WebGL hay que utilizar un elemento canvas. Este elemento lo inicializaremos desde javascript, utilizando una función específica de WebGL (*canvas.getContext(“webgl”)*). El valor resultante de esta función se lo asignaremos a una variable global (*“gl”* en nuestro caso).

La variable *“gl”* será aquella que utilizaremos para todas las funciones que conlleven un pintado o renderizado de elementos. Con ella podremos determinar los shaders que vamos a utilizar y asignar toda la configuración necesaria. También podremos crear buffers donde guardaremos los objetos completos que usaremos para formar los escenarios, por ejemplo, un cubo, un cilindro o una esfera.

## **GitHub**

En un principio, utilizamos Dropbox para almacenar y tener el código subido a la nube. Esto no nos era suficiente, ya que no podíamos llevar un control exhaustivo del código, ni tener diferentes versiones de la aplicación.

Debido a esto, comenzamos a utilizar Github, que es un sistema de control de versiones que está en la nube (nos permite tener el código en todo momento). Mediante Github, podemos tener el registro de cambios que realizamos en nuestro código.

Además de llevar el registro de cambios, podemos utilizar diferentes ramas para experimentar con GitHub, por ejemplo, para cambiar el sistema de shaders.

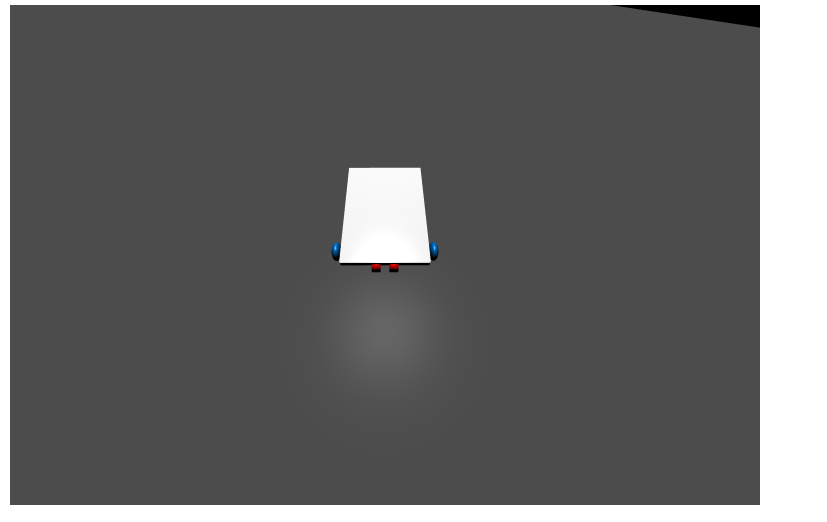
# **Descripción de la aplicación**

En la aplicación podemos observar tres zonas importantes: Viewport, campos de entrada de datos y la zona de información

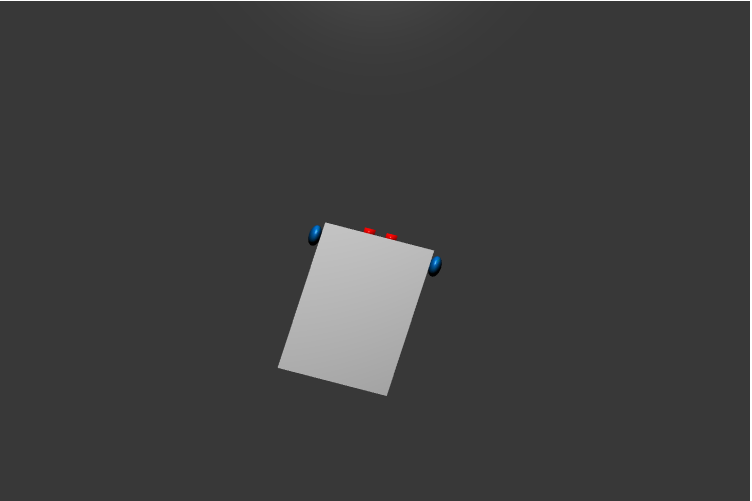
## **Zona izquierda: Viewport**

En la izquierda podemos observar el canvas, donde podemos observar a nuestro robot. En un inicio el robot estará sobre una superficie gris, para poder ver bien el movimiento y las iluminaciones sobre el coche.

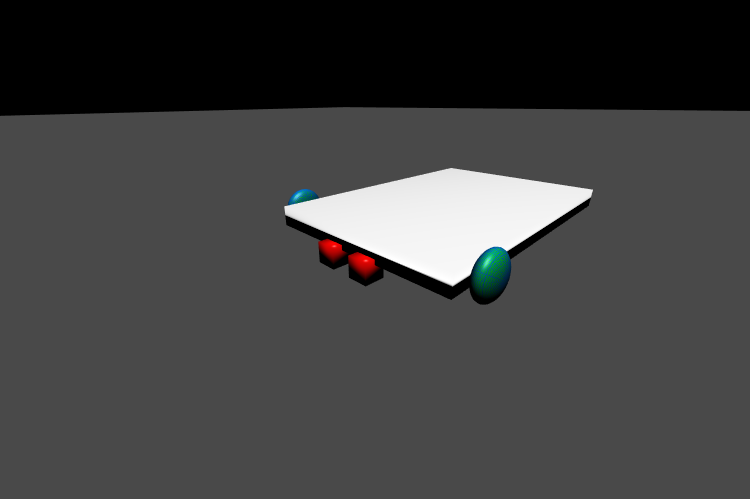
Al comienzo, la cámara estará en centrada en el coche en cualquier modo: “Automática”, “Ortográfica” o “Siguiendo” (Figuras 3, 4 y 5).



3. Cámara Automática

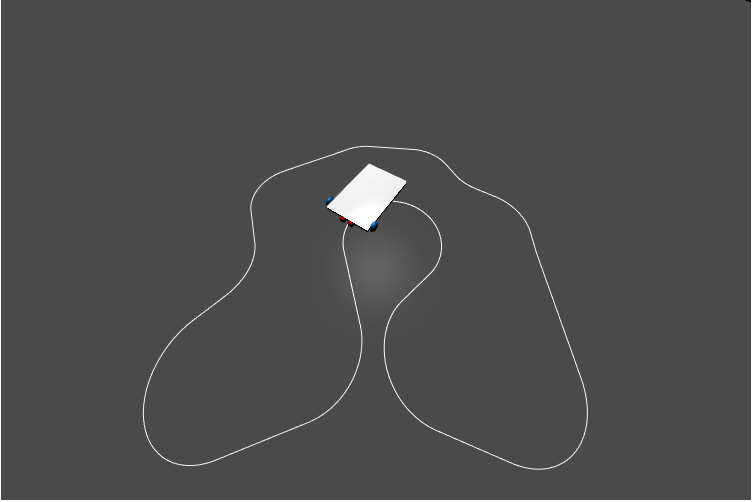


. Cámara Ortográfica

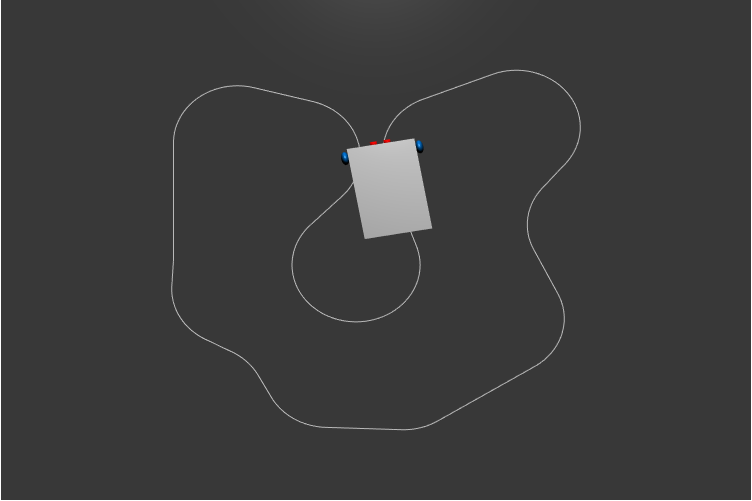


. Cámara Siguiendo

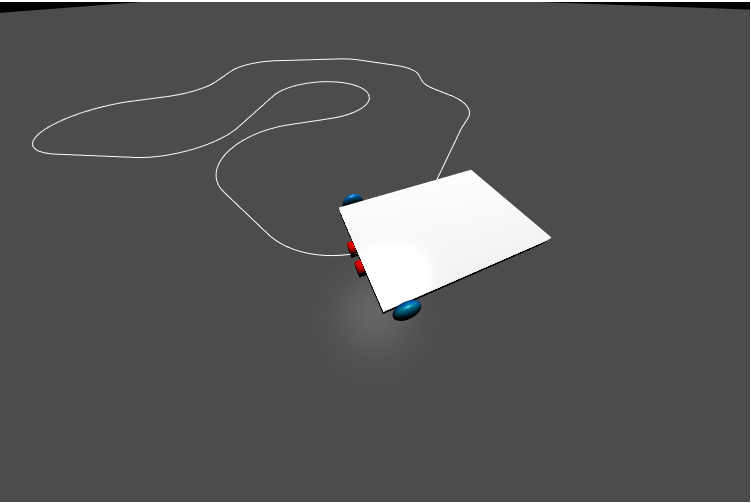
En el momento en el que añadamos un circuito los modos de cámara “Automática” y “Ortográfica” estarán centradas en el punto central del circuito (Figuras 6, 7 y 8).



. Cámara automática centrada



. Cámara Ortográfica Centrada



. Cámara Siguiendo con circuito

Podemos realizar varias acciones con el teclado y ratón sobre el canvas:

* Si hacemos click izquierdo sobre el canvas y arrastramos con el ratón podemos girar la cámara dependiendo del modo que tengamos activado.
* Podemos aumentar y disminuir la velocidad con las teclas “*W*” (Aumentar velocidad) y “*S*” (Disminuir velocidad).
* Podemos girar el robot manualmente con las teclas “*D*” (Derecha) y “*A*” (Izquierda).
* También podemos hacer zoom con la rueda del ratón.

Descripción de la parte izquierda de la aplicación (viewport), explicando cómo se ubican los diferentes elementos en la escena (centrado el circuito en el origen y posición de la cámara en sus diferentes vistas de acuerdo con los dos modelos de proyección).

Si finalmente conseguimos el control de la cámara mediante el ratón (posición y zoom) la descripción de dicho sistema iría aquí.

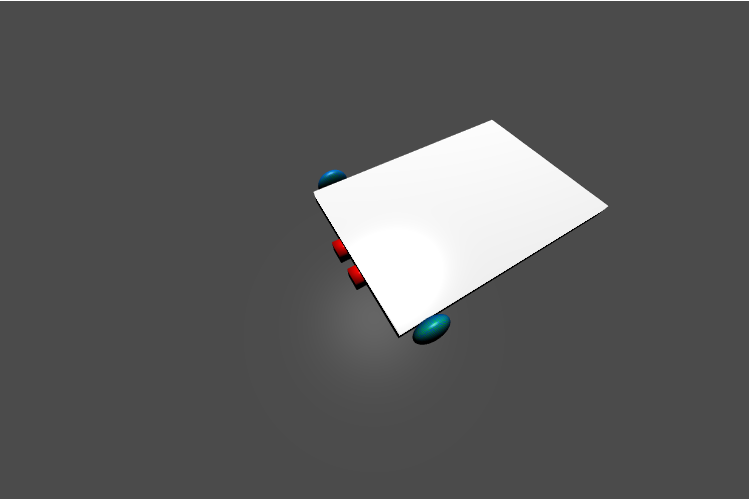
Finalmente, dicha zona también se aprovecha para mostrar todos los parámetros geométricos del robot sobre una imagen del mismo.

## **Zona derecha: Campos de entrada datos**

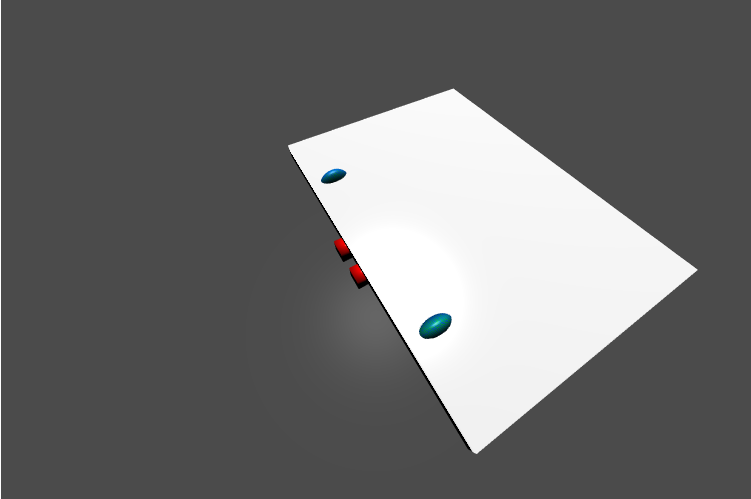
En nuestra zona derecha podemos observar distintos campos editables. Estos campos nos determinarán las características del coche que son editables. Todas estas características afectan de una manera u otra a las propiedades de movimiento del coche, tanto aumentando la velocidad como aumentando el ángulo máximo de giro.

Las características son las siguientes:

* Ancho: Nos permitirá ajustar el ancho del coche. Esta característica es solamente visible, ya que no es la que nos ajusta el ancho entre las ruedas. Figuras 9 y 10.

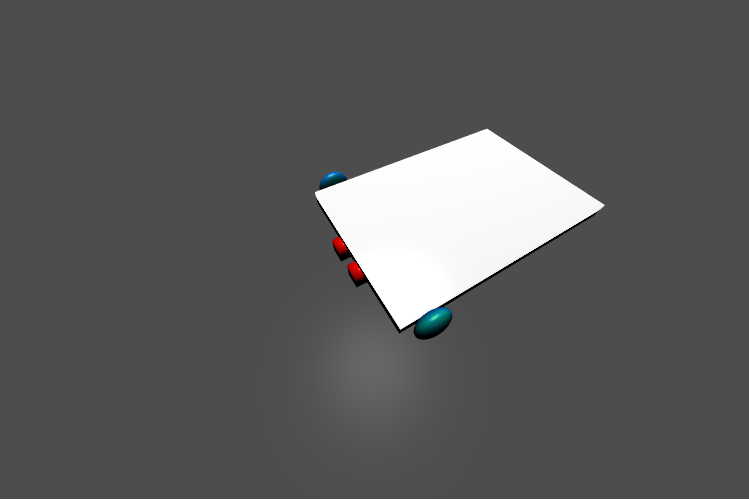


. Vehículo con ancho 10

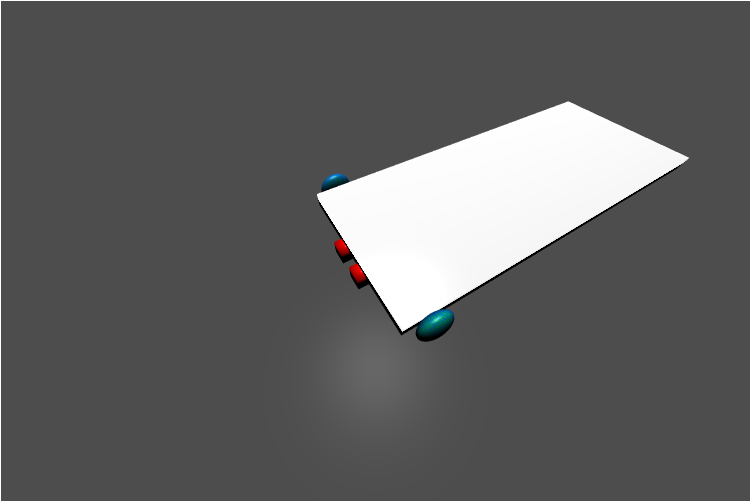


. Vehículo con ancho 20

* Largo: Nos permitirá ajustar el largo del coche. Esta característica no afecta al movimiento. Figuras 11 y 12.

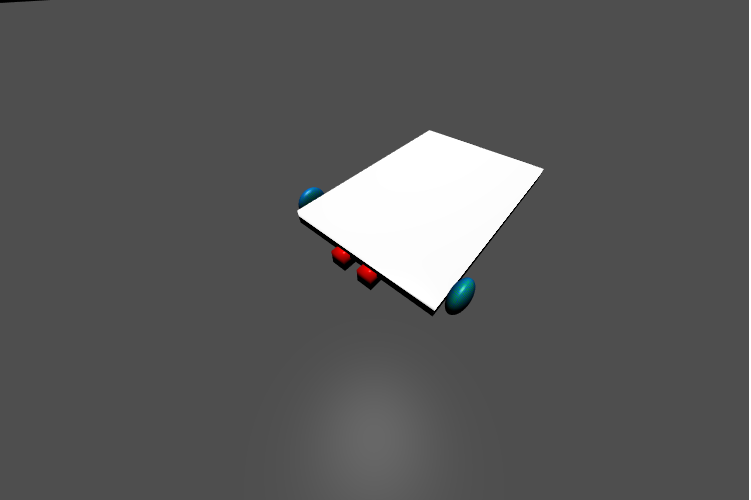


. Vehículo con largo 15

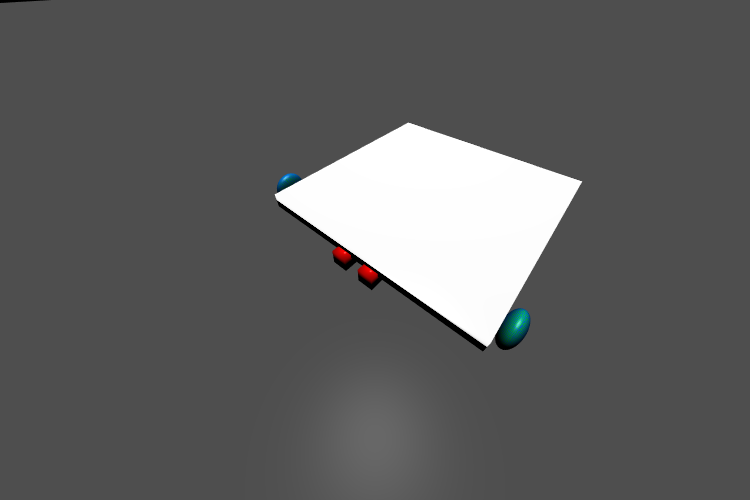


. Vehículo con largo 25

* Distancia entre ruedas: Nos permitirá asignarle una distancia entre las dos ruedas del coche. Al aumentar la distancia provocaremos que el coche gire menos, ya que al aumentar el radio de giro de las ruedas haríamos que la rueda contraria tuviese que recorrer más espacio para cambiar el ángulo. Figuras 13 y 14.

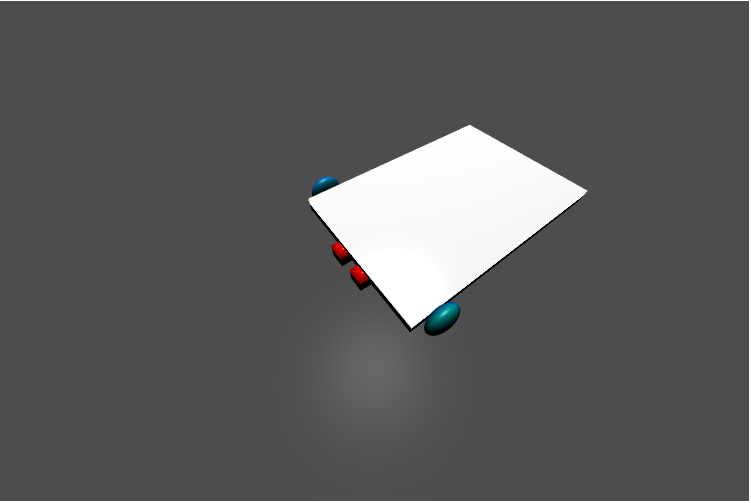


. Vehículo con Distancia entre ruedas 10

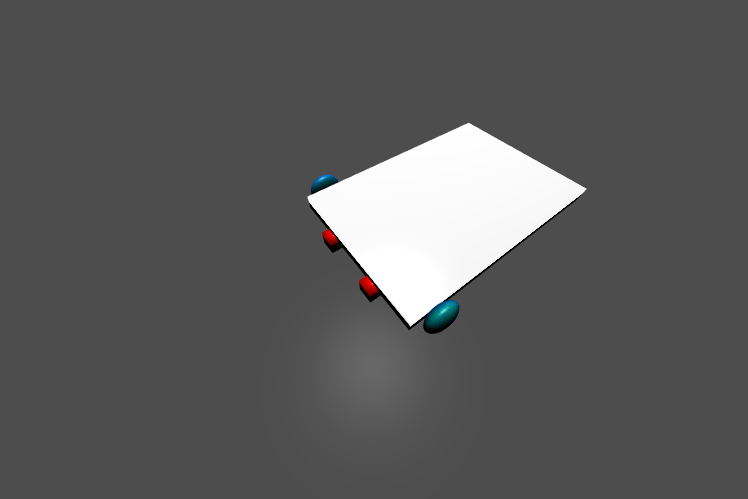


. Vehículo con Distancia entre ruedas 15

* Distancia entre sensores: Nos permitirá darle una distancia entre los dos sensores. Hay que tener en cuenta que los sensores miden 1cm en nuestro coche, por lo que si ponemos una distancia de menos de 2 estaría en todo momento tocando la línea y esto llevaría a resultados erróneos. Al aumentar la distancia provocaremos que el coche esté mas tiempo sin girar cuando llegue a una curva, o incluso recorra recto una sucesión de curvas si estas no giran lo suficiente; pero en caso de que haya una curva demasiado cerrada que luego pase de nuevo junto a otra carretera tendría la posibilidad de tocarse y llevar al coche hacia otro sitio. Figuras 15 y 16.

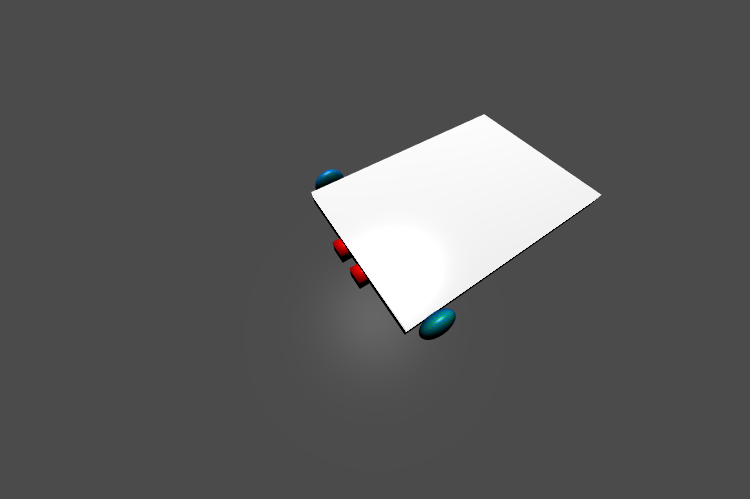


. Vehículo con distancia entre sensores 4

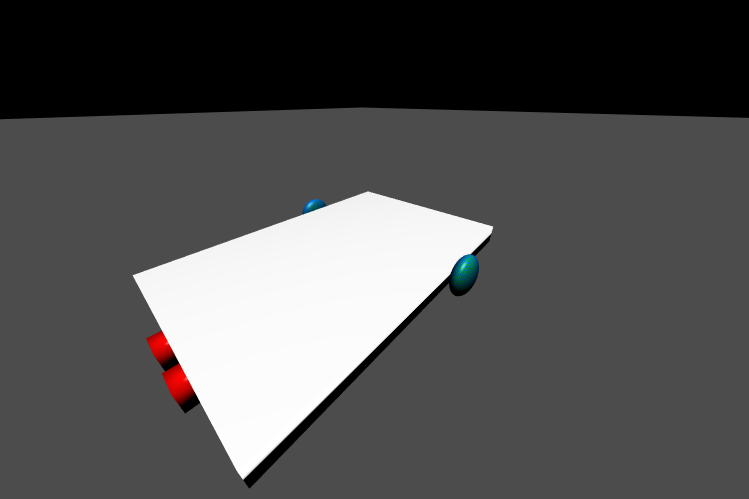


. Vehículo con distancia entre sensores 8

* Distancia Ruedas-Sensores: Ajustar esta opción nos resultará en una mayor distancia entre las ruedas y los sensores en torno a lo largo del vehículo. Aumentar este valor provocará que el coche tenga el punto de giro mucho mas atrás que el punto en el que detecta el punto del circuito, provocando que los giros lleguen antes. Figuras 17 y 18.

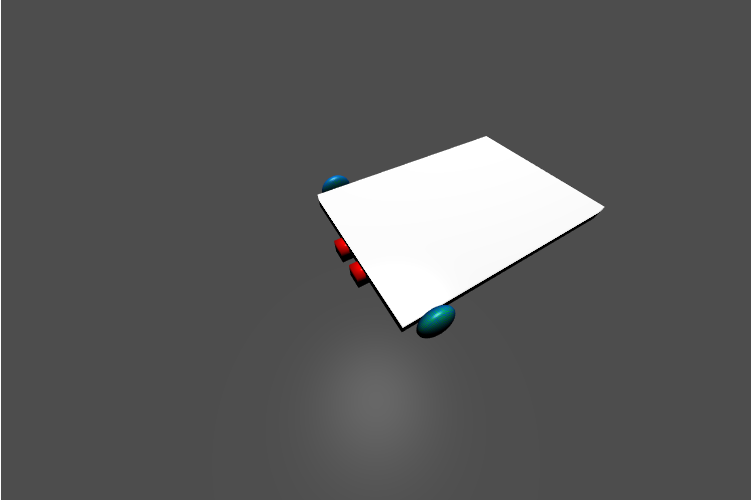


. Vehículo con distancia Ruedas-sensores 3

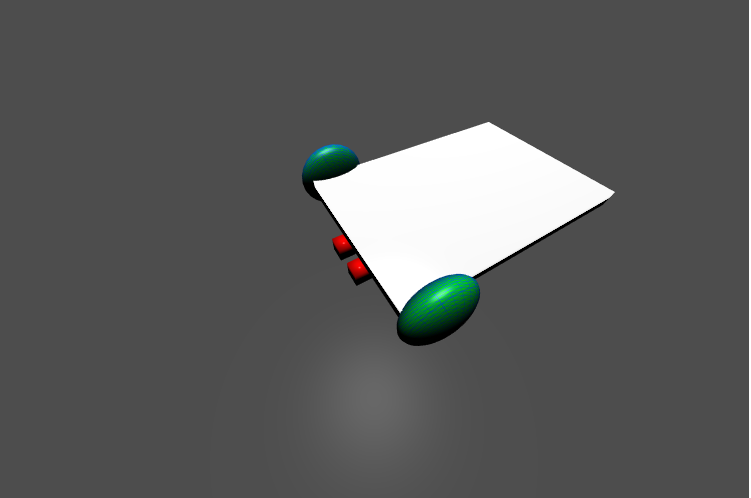


. Vehículo con distancia Ruedas-sensores 20

* Radio rueda: Ajustar esta opción aumentará o disminuirá el radio de la rueda. Esta característica no afecta al movimiento, ya que la velocidad que tenemos indicada es lineal. Figuras 19 y 20.



. Vehículo con radio de rueda 2



. Vehículo con radio de rueda 4

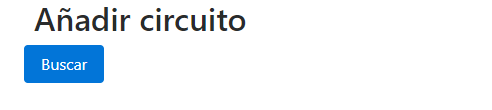
* Velocidad: Ajustar esta opción aumentará o disminuirá la velocidad lineal de las ruedas de nuestro vehículo. Si la aumentamos provocaremos que el vehículo aumente la velocidad.

Tras ajustar todos estos valores, podremos aplicarlos presionando la tecla *Enter* o apretando el botón *Guardar Cambios*.

## **Zona inferior. Información**

En la zona inferior de la pantalla podremos observar cuatro zonas:

* Cámaras: Permitirán cambiar entre las cámaras explicadas en el punto 1.
* Añadir circuito: Este botón nos permitirá buscar un archivo .txt en nuestro sistema que tenga el formato de un circuito, para añadirlo al mapa y que el coche lo pueda seguir.



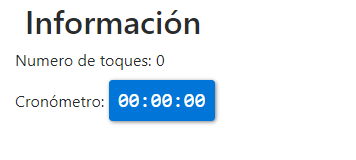
. Botonera añadir circuito

Tras añadirlo nos aparecerá un botón de empezar circuito, que nos colocará el coche en el inicio del circuito para que podamos probar. Seguirá el botón de buscar por si queremos cambiar de circuito.



. Botonera añadir circuito con empezar circuito

* Información: En esta sección nos aparecerá información sobre la vuelta actual que lleva el coche en el circuito. Solo funcionará cuando hayamos añadido un circuito.



. Sección información antes de añadir circuito.

El número de toques nos indicará la cantidad de veces que el sistema ha detectado que se ha tocado una línea, lo que nos indicará la eficiencia de cierta manera del vehículo.

El cronómetro nos servirá para observar cuanto tiempo lleva el vehículo realizando el circuito. Se reinicia a cada vuelta.



. Sección información durante el recorrido de un circuito.

* Vueltas: En esta sección nos aparecerán las vueltas que lleva nuestro vehículo, además de sus registros. De esta manera podemos comparar con los cambios que hagamos a nuestro vehículo.



. Sección Vueltas con ejemplos.

## **Casos de uso**

Hemos realizado varias pruebas con el vehículo para ver la eficiencia de nuestro vehículo en un circuito predefinido. Los resultados son los siguientes:

## Primera prueba

Para la primera prueba hemos dispuesto el coche con unos valores básicos y normales:



. Valores primera prueba

Con estos valores hemos dejado el vehículo durante tres vueltas y hemos obtenido los siguientes resultados:

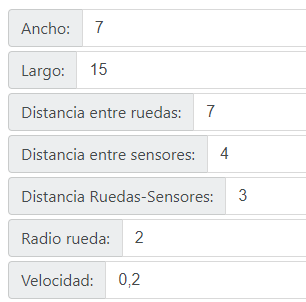


. Resultados primera prueba

Podemos definir estos resultados como los resultados base y a partir de los que vamos a comparar las siguientes pruebas. De estos resultados podemos observar la diferencia entre las dos primeras vueltas, lo que nos hace entender que, en la primera vuelta, al estar mejor colocado de salida (Sale recto) nos permite reducir el número de toques.

## Segunda prueba

En esta segunda prueba vamos a reducir la distancia entre ruedas, para ver cómo se comporta frente a un eje más pequeño. Los valores serán los siguientes:



. Valores segunda prueba

Con estos valores, hemos obtenido los siguientes resultados:



. Resultados segunda prueba

Podemos observar la gran diferencia de toques entre las dos pruebas. Al disminuir la distancia entre las ruedas hemos provocado que nuestro vehículo gire mejor, por lo que llega a chocar menos veces contra el circuito, disminuyendo de ese modo el tiempo medio que tarda en resolver una vuelta.

De esta prueba sacamos la conclusión de que una distancia entre ruedas baja podría ser la solución en caso de que nuestro vehículo girase demasiado poco.

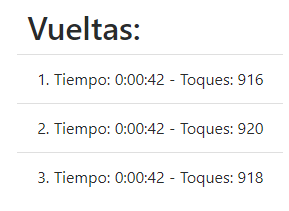
## Tercera prueba

En la tercera prueba vamos a reducir la distancia entre los sensores, de tal manera que el vehículo tenga más probabilidad de tocar, pero irá siguiendo el circuito de una manera más precisa.



. Valores tercera prueba

Con estos valores, hemos dejado el vehículo tres vueltas y hemos obtenido los siguientes resultados:



. Resultados tercera prueba

Con estos resultados podemos observar la precisión que ha obtenido nuestro vehículo realizando el circuito, de manera que tarda exactamente los mismos segundos y número de toques muy ajustado. Respecto a los valores base, podemos observar que la diferencia es mínima entre los dos.

## Cuarta Prueba

### 

Para esta prueba vamos a aumentar la distancia Ruedas-Sensores en una cantidad bastante grande, provocando que el vehículo gire anticipándose al circuito. Los valores serán los siguientes:



. Valores cuarta prueba

Con estos valores, hemos dejado el vehículo tres vueltas y hemos obtenido los siguientes resultados:



. Resultados cuarta prueba

Con estos resultados podemos observar una gran diferencia respecto a los anteriores resultados.

El hecho de provocar que el vehículo gire anticipándose a las curvas nos permite acortar espacio sobre todo en aquellos giros que son muy cerrados, ya que casi no hace falta entrar en el giro mientras que los sensores siguen la línea.

Hemos conseguido reducir tanto el número de toques como el tiempo en realizar la vuelta, lo que nos indica que aumentar la distancia Ruedas-Sensores podría darnos unos mejores resultados.

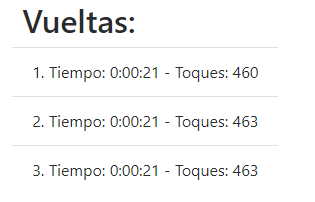
## Quinta prueba

Para esta prueba vamos a aumentar la velocidad lineal de nuestro vehículo, duplicándola, para observar que nos provocaría un aumento en la velocidad en nuestros resultados. Los valores serán los siguientes:



. Valores quinta prueba

Con estos valores, hemos dejado el vehículo tres vueltas y hemos obtenido los siguientes resultados:



. Resultados quinta prueba

Como podemos observar, aumentar la velocidad lineal de las ruedas de nuestro vehículo nos permite dividir entre dos tanto el tiempo como los toques que realizamos. El tiempo es entendible, ya que vamos el doble de rápido, pero ¿Porqué el número de toques?

Esto se debe a que lo que hemos aumentado es la velocidad lineal máxima de nuestras ruedas. Cuando nuestro robot detecta que un sensor está tocando una línea provoca que la rueda de ese lado frene, creando de esa manera un giro (ya que la rueda opuesta sigue girando).

Si aumentamos la velocidad lineal de las ruedas provocamos que, cuando una de las ruedas se para, la otra avanza un mayor espacio, girando el doble de rápido. Esto provoca que el vehículo deje de tocar la línea en la mitad de tiempo.

## Sexta Prueba

Para la sexta y última prueba, vamos a realizar una prueba múltiple, intentando sacar los mejores resultados variando nuestros valores. Daremos 5 vueltas para cada una de las recogidas de datos. En la siguiente tabla podemos observar los valores utilizados y los resultados obtenidos en tiempo y en toques. Los resultados que aparecen con un guión (-) significa que el vehículo ni siquiera ha conseguido terminar una vuelta.

Además, pondremos una velocidad fija de 0,5, ya que por pruebas anteriores podemos observar que aumentar la velocidad es una disminución asegurada del tiempo.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nº** | **Ancho** | **Largo** | **DR** | **DS** | **DR-S** | **RR** | **Velocidad** | **Toques** | **Tiempo** |
| 1 | 10 | 15 | 10 | 8 | 0 | 2 | 0,5 | - | - |
| 2 | 10 | 15 | 10 | 5 | 0 | 2 | 0,5 | - | - |
| 3 | 15 | 20 | 15 | 8 | 5 | 2 | 0,5 | 549 | 00:18,0 |
| 4 | 15 | 20 | 15 | 3 | 10 | 2 | 0,5 | 505,2 | 00:16,2 |
| 5 | 5 | 10 | 5 | 6 | 1 | 2 | 0,5 | 190 | 00:15,8 |
| 6 | 5 | 10 | 5 | 3 | 10 | 2 | 0,5 | 268,4 | 00:14,6 |
| 7 | 5 | 10 | 5 | 6 | 10 | 2 | 0,5 | 167,4 | 00:14,4 |
| 8 | 5 | 10 | 5 | 5 | 15 | 2 | 0,5 | 154 | 00:13,2 |
| 9 | 5 | 10 | 5 | 4 | 18 | 2 | 0,5 | 296,4 | 00:13,4 |
| 10 | 5 | 10 | 5 | 6 | 18 | 2 | 0,5 | 144,2 | 00:12,6 |

Tabla . Resultados prueba 6

De estos valores podremos apreciar cuales son los puntos fuertes si queremos reducir al máximo el tiempo que tarda en dar la vuelta nuestro vehículo.

Además, realizando una combinación entre dos o tres características podemos hacer que se reduzca aún más, ya que, fijándonos en las pruebas 9 y 10 podemos fijarnos que, si tenemos una distancia entre sensores baja y las ruedas muy por detrás de los sensores, provocarán que en el momento que el coche gire, los sensores recorran más espacio, por lo que tienen más probabilidades de tocar si están más juntos.

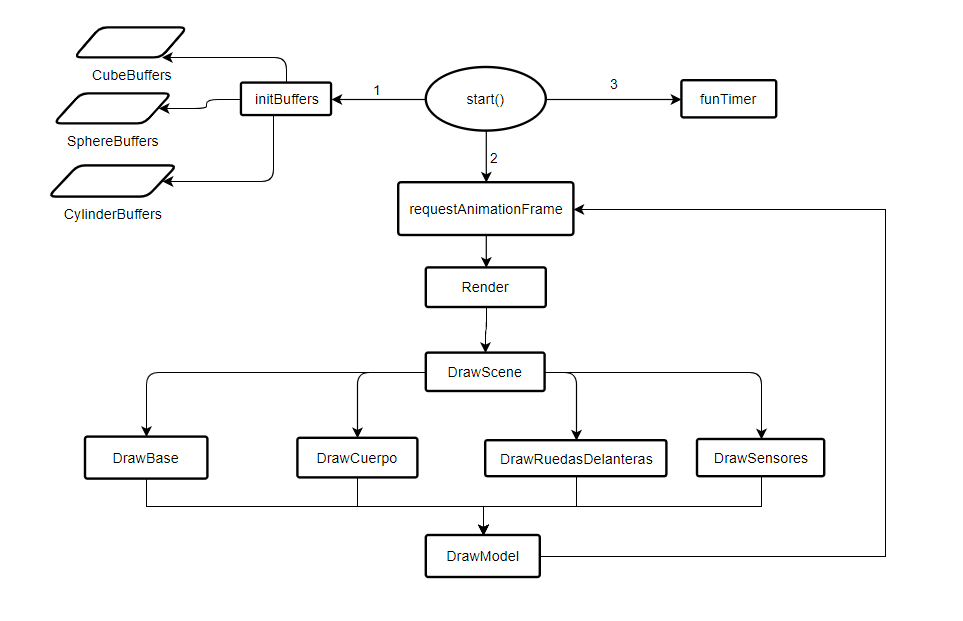
# **Conclusiones**

En conclusión, con esta aplicación desarrollada podemos ahorrar mucho tiempo y esfuerzo en obtener resultados bastante detallados sobre como podríamos construir nuestro robot físico.

Nos puede servir también para probar diferentes circuitos de una manera muy sencilla. Tan solo habría que definir los puntos y ya puedes probarlo en la aplicación.

Respecto a futuro, podríamos evolucionar esta herramienta mediante la adición de un editor avanzado del vehículo, que nos permitiese observar completamente como quedaría, y que también nos permitiese modificar otros valores que pudiesen afectar al comportamiento en circuito.

# **Apéndice**



. Diagrama de flujo del renderizado.

# **Bibliografía**

* Robótica. Manipuladores y robots móviles. *Aníbal Ollero Baturome*
* <https://es.wikipedia.org/wiki/JavaScript>
* <https://es.wikipedia.org/wiki/HTML>
* <https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada>
* <https://www.chartjs.org/>
* <https://developers.arcgis.com/javascript/3/>
* <https://es.wikipedia.org/wiki/WebGL>