



Con formato: Sangría: Izquierda: 2,67 cm, Espacio Después: 12,2 pto

Con formato: Derecha

Con formato: Fuente: 18 pto

Con formato: Sangría: Izquierda: 0 cm

Tabla con formato

Con formato: Fuente: 26 pto

Con formato: Izquierda, Sangría: Izquierda: -0,06 cm, Primera línea: 0,06 cm

Con formato: Centrado, Sangría: Izquierda: 0 cm

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

**DOBLE GRADO EN INGENIERIA INFORMATICA E INGENIERIA
SOFTWARE**

Curso Académico 2017/2018

Trabajo Fin de Grado

Con formato: Espacio Después: 11,2 pto

Con formato: Espacio Después: 11,15 pto

SIMULACIÓN DE UN ROBOT SIGUE **LINEASDESARROLLO DE UNA APLICACIÓN PARA** **LA SIMULACIÓN DE UN ROBOT MÓVIL CON** **DIRECCIONAMIENTO DIFERENCIAL**

Autor: David Vacas Miguel

Director/Tutor: Alberto Herrán González

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sangría: Izquierda: 0 cm, Interlineado: 1,5 líneas

Con formato: Fuente: 18 pto, Negrita, Color de fuente: Negro

Con formato: Espacio Después: 10,95 pto

Con formato: Espacio Después: 12,9 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Índice

ÍNDICE	3
AGRADECIMIENTOS.....	5
RESUMEN.....	6
CAPÍTULO 1 INTRODUCCIÓN	7
1. MOTIVACIÓN	7
2. OBJETIVOS.....	10
3. ESTADO DEL ARTE.....	13
4. ESTRUCTURA DE LA MEMORIA	14
CAPÍTULO 2 ROBÓTICA MÓVIL	15
1. ROBÓTICA MÓVIL.....	15
2. VEHÍCULOS CON RUEDAS	18
3. MODELO FÍSICO DIRECCIONAMIENTO DIFERENCIAL.....	22
4. NAVEGACIÓN AUTÓNOMA	29
CAPÍTULO 3 ENTORNO TECNOLÓGICO	32
1. OPENGL	33
2. ETAPA GEOMÉTRICA.....	38
3. QT	43
4. METODOLOGÍA ÁGIL	49
CAPÍTULO 4 DESCRIPCIÓN DE LA APLICACIÓN	51
1. ZONA IZQUIERDA: VIEWPORT	51
2. ZONA DERECHA: CAMPOS DE ENTRADA DE DATOS	57
3. CASOS DE USO	62
CONCLUSIONES	66
BIBLIOGRAFÍA.....	67
ÍNDICE	2
AGRADECIMIENTOS.....	3
RESUMEN.....	4
CAPÍTULO 1 INTRODUCCIÓN	5
1. MOTIVACIÓN	5
2. OBJETIVOS.....	8
3. ESTADO DEL ARTE.....	10
4. ESTRUCTURA DE LA MEMORIA	11
CAPÍTULO 2 ROBÓTICA MÓVIL	12
1. VEHÍCULOS CON RUEDAS	12
2. DIFERENTES CONFIGURACIONES	15
3. DIRECCIONAMIENTO DIFERENCIAL	18
4. NAVEGACIÓN AUTÓNOMA	20
CAPÍTULO 3 ENTORNO TECNOLÓGICO	21
1. QT	21

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente de párrafo predeter., Fuente: Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Fuente: Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Fuente: Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Fuente: Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Fuente: Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Fuente: Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva



2.	OPENGL	23
3.	GLM	26
4.	METODOLOGÍA ÁGIL	27
CAPÍTULO 4 DESCRIPCIÓN DE LA APLICACIÓN		29
1.	ZONA IZQUIERDA: VIEWPORT	29
2.	ZONA DERECHA: CAMPOS DE ENTRADA DE DATOS	32
3.	CASOS DE USO	36
CONCLUSIONES		39
BIBLIOGRAFÍA		40

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Fuente:
Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Versalitas

Con formato: Fuente de párrafo predeter., Fuente:
Negrita, Mayúsculas

Con formato: Fuente de párrafo predeter., Fuente:
Negrita, Mayúsculas

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Agradecimientos

Me gustaría agradecer a la universidad por el conocimiento recibido y en especial a mi tutor, Alberto Herrán, por su ayuda y apoyo durante el desarrollo del proyecto.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Resumen

En este TFG se ha desarrollado una aplicación con la que poder simular y visualizar el funcionamiento de un robot móvil con direccionamiento diferencial cuando trata de seguir ~~un el~~ recorrido marcado por una línea negra sobre un fondo blanco.

Para ello, se ha comenzado estudiando la mecánica de tal sistema a través de las ecuaciones que relacionan el giro de los motores con la posición y orientación del ~~mismo robot~~. A ~~continuación~~ continuación, se han analizado el entorno tecnológico que ha sido utilizado de las diferentes tecnologías disponibles para la implementación, habiéndose ~~elegido Qt~~ elegido Qt y OpenGL ~~para la interacción y simulación~~. Posteriormente se ha realizado la implementación de la simulación del comportamiento del robot y su visualización sobre el circuito, además de y la interacción con el usuario final mediante inputs en la aplicación. Finalmente, se han implementado opciones adicionales para la mejora ~~de la aplicación de la experiencia del usuario y de la aplicación~~.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Capítulo 1 Introducción

1. Motivación

Este trabajo nace motivado por un intento de mejorar una de las herramientas que se utilizaban utilizadas en la antigua 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.

En esta asignatura los alumnos, en grupos, debían realizar construir robots de distintos tipos: sigue líneas, velocistas, etc. Al final del curso el profesor realizaba un circuito y una competición en la que se usaban los robots que se habían construido para ver cuál era el más rápido, creando así una competición de robots entre los alumnos. El tiempo en el que el robot realizaba el circuito venía dado por las medidas que los alumnos utilizasen a la hora de construir el mismo En esta competición se cronometraba el tiempo que tardaba cada robot en realizar el circuito, este tiempo dependía de ciertos parámetros que los alumnos elegían a la hora de crear el robot, por ejemplo un robot con una separación de sensores mayor realiza menos giros pero de mayor amplitud sin embargo un robot con una separación menor realiza más giros pero de menor magnitud y esto junto con los demás parámetros influye en la velocidad para realizar el circuito.

El objetivo de este proyecto es desarrollar una aplicación que permita simular y visualizar la dinámica de este robot móvil con direccionamiento diferencial. La aplicación servirá de banco de pruebas con el que analizar el comportamiento del robot sigue líneas antes de su construcción real.

Como se ha mencionado anteriormente antes, actualmente se dispone de un desarrollo previo en MATLAB-Simulink, en la figura 1 se puede observar el esquema de dicho desarrollo, en el que se puede ver los diferentes componentes que se utilizan en la aplicación ver figura 1 y figura 2n. La figura 2 muestra el programa en ejecución.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

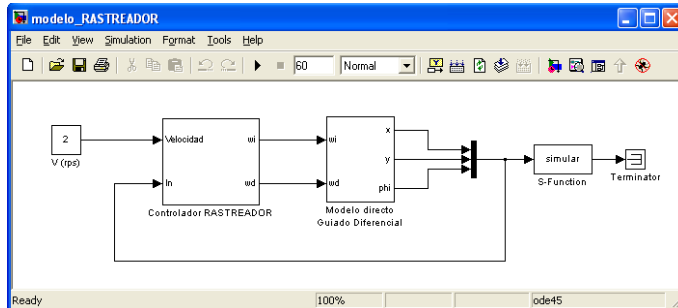


Figura 1. Modelo del desarrollo hecho en MATLAB-Simulink.

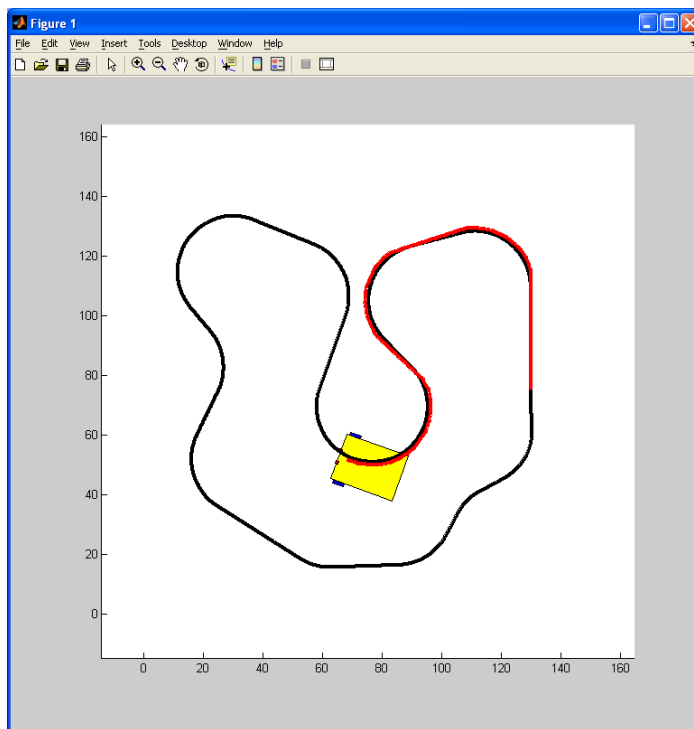


Figura 2. Vista de la simulación desarrollada en MATLAB-Simulink.

Sin embargo, este tiene varios inconvenientes:

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Centrado, Espacio Antes: Automático, Después: 6 pto

Con formato: Sangría: Izquierda: 0 cm, Sangría francesa: 1,25 cm, Espacio Antes: Automático, Después: 6 pto

Con formato: Fuente: 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

- a) Se debe tener instalado ~~dic~~ dic ~~ho software~~ MATLAB-Simulink para su uso.
- b) No se permite cambiar los parámetros del robot de una forma sencilla, estos parámetros se debían escribir en un vector cuya estructura debes conocer, como se puede ver en la figura 3.

```
>> robot
robot =
    22    16     3     2    -2     3     2     3     1
```

Figura 3. Se puede observar el vector en el cual se deben poner los parámetros del robot.

- c) No ofrece todas las funcionalidades requeridas, estas se especifican más adelante en el punto 2 Objetivos.

```
>> robot
robot =
    22    16     3     2    -2     3     2     3     1
```

Figura 3. Vector en el cual se deben poner los parámetros del robot.

Por ello, se ha desarrollado ~~dic~~ dic ~~ho software creando~~ una aplicación (integrando las tecnologías C++, Qt y OpenGL) cuyo único requisito para su uso es el de tener instaladas en el sistema las librerías necesarias para hacerlo funcionar.

El destinatario de la aplicación son alumnos de una asignatura en la que se pretende que jueguen con el simulador antes de la construcción real del robot ~~real~~.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sangría: Izquierda: 1,25 cm

Con formato: Espacio Antes: Automático, Después: 6 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

2. Objetivos

Una vez se han visto las carencias de la aplicación que se tenía anteriormente, los objetivos que servirían para subsanarlas, además de otros objetivos adicionales serían los siguientes:

- **Eliminar la necesidad de tener software instalado:** con esta aplicación no será necesario disponer de ningún software adicional instalado, más allá de las librerías necesarias para la aplicación, por lo que será más fácil su descarga y utilización inicial.

- **Facilidad a la hora de realizar cambios:** el mayor problema a la hora de crear el robot se hallaba en que una vez se decidían las medidas y se compraban o creaban las piezas en cuestión era complicado cambiar estas para utilizar distintas. Por ejemplo si se decide que el radio de las ruedas sea de Xcm una vez se compraba la rueda, en caso de querer cambiarla se debía comprar una nueva, otro ejemplo podría ser el tamaño del robot, si se decide porque el robot tuviera unas dimensiones de (Y,Z)Y-Zcm pero más tarde se daba cuenta que sería mejor tener un robot con unas dimensiones mayores, estos debían volver a crear el soporte del robot puesto que este es el que indica las dimensiones.

Esto se soluciona en la aplicación con una simple interfaz user-friendly en la que se podrá cambiar tantas veces como se quiera las medidas de su robot, de forma sencilla y natural.

- **Realización de pruebas con facilidad:** debido a lo costoso de realizar un robot y la necesidad de crear un circuito por el cual el robot tenga que correr se hacía complicado poder probar el robot creado, o diferentes opciones para el mismo.

Con esta aplicación se podrán realizar la cantidad de pruebas que se desee puesto que en unos segundos se podrá tener un robot simulado corriendo por el circuito dado sin necesidad de crear físicamente los mismos.

- **Mejora del aprendizaje:** igual que en el anterior punto, puesto que los alumnos no podían realizar pruebas con el robot de forma sencilla, no podían observar

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Negrita, Sin subrayado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

detenidamente los cambios que se creaban en el movimiento del robot utilizando al utilizar diferentes medidas.

Puesto que aquí se tiene una forma rápida de simular el movimiento del robot, se puede experimentar haciendo cambios en las medidas para poder observar qué cambios se realizan en el movimiento del robot y por lo tanto ir aprendiendo con la visualización del mismo de este.

-

Facilidad a la hora de realizar cambios: el mayor problema a la hora de crear el robot se hallaba en que una vez se decidían las medidas y se compraban e creaban las piezas en cuestión era complicado cambiar estas para utilizar distintas. Por ejemplo si se decidía que el radio de las ruedas sería de X una vez se compraba la rueda, en caso de querer cambiarla se debía comprar una nueva, otro ejemplo podría ser el tamaño del robot, si se decidía porque el robot tuviera unas dimensiones de (Y, Z) pero más tarde se daba cuenta que sería mejor tener un robot con unas dimensiones mayores, estos debían volver a crear el soporte del robot puesto que este es el indica las dimensiones.

—

Esto se soluciona en la aplicación con una simple interfaz user friendly en la que se podrá cambiar tantas veces como se quiera las medidas de su robot de forma sencilla y natural.

Realización de pruebas con facilidad: debido a lo costoso de realizar un robot y la necesidad de crear un circuito por el cual el robot tenga que correr se hacía complicado poder probar el robot creado, o diferentes opciones para el mismo.

=====

Con esta aplicación se podrán realizar la cantidad de pruebas que se desee puesto que en unos segundos se podrá tener un robot simulado corriendo por el circuito dado sin necesidad de crear físicamente los mismos.

—

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sangría: Izquierda: 1,27 cm, Sin viñetas ni numeración

Con formato: Fuente: Negrita, Sin subrayado

Con formato: Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Fuente: Negrita

Con formato: Subrayado

Con formato: Fuente: Negrita, Sin subrayado

Con formato: Fuente: Negrita

Con formato: Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Subrayado

Con formato: Sangría: Izquierda: 1,27 cm, Sin viñetas ni numeración

Con formato: Fuente: Negrita, Sin subrayado

Con formato: Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Sangría: Izquierda: 0 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sangría: Izquierda: -0,63 cm, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Subrayado

Con formato: Sin viñetas ni numeración

Con formato: Sangría: Izquierda: 1,27 cm, Primera línea: 0 cm

~~Mejora del aprendizaje: igual que en el anterior punto, puesto que los alumnos no podían realizar pruebas con el robot de forma sencilla, no podían observar detenidamente los cambios que se creaban en el movimiento del robot utilizando diferentes medidas.~~

~~Puesto que aquí se tiene una forma rápida de simular el movimiento del robot, se puede experimentar haciendo cambios en las medidas para poder observar qué cambios se realizan en el movimiento del robot y por lo tanto ir aprendiendo con la visualización del mismo.~~

Por lo ~~tantetanto~~, los requisitos que debe tener la aplicación son:

- Facilidad de cambiar los parámetros del robot.
- Visualización de la simulación del robot de forma correcta.
- —
- Integración de la interfaz y de la simulación en la aplicación.
- —
- Interfaz user-friendly, fácil de utilizar para todos.
- —
- Diseño simple de la simulación para poder visualizar mejor los movimientos de tu robot.
- —
-

Con formato: TFG título 2, Sangría: Izquierda: 0 cm, Sangría francesa: 0,75 cm, Numerado + Nivel: 1 + Estilo de numeración: 1, 2, 3, ... + Iniciar en: 1 + Alineación: Izquierda + Alineación: 0,63 cm + Sangría:

Con formato: Fuente: 16 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Estado del arte

En cuanto a las herramientas que se pueden encontrar en el mercado para simular un robot de este tipo se pueden encontrar de dos tipos:

- Aplicaciones creadas por una persona ~~las que~~ ~~cuales~~ muestran cómo realizarlas o dejan la aplicación en un repositorio público (pocas de estas realizan esto último). Estas aplicaciones suelen tener dos problemas: la dificultad para su descarga y utilización, que no ~~están-se hallan~~ preparadas para su uso puesto que están en las IDE correspondientes y la interfaz no es fácil de usar, en caso de que haya interfaz y no se tenga que cambiar los parámetros por código.

- Aplicaciones creadas por empresas. En este caso muchas de estas aplicaciones tienen una carencia en la UI, no se puede cambiar de ~~manera~~ ~~sencilla~~ manera sencilla los parámetros del robot. Lo bueno que tenía la herramienta encontrada es que no solo servía para un tipo de ~~robot~~ robot, sino que incluía en la aplicación varios tipos de robots. Ejemplos de este tipo son: Hemero, una herramienta en Matlab-simulink para la enseñanza de la robótica. El problema que posee esta herramienta es la falta de parte grafica; RoboWorks, se trata de una aplicación de modelado, simulación y animación de sistemas físicos, sin embargo, se utiliza solo para robots manipuladores. RoboDK, sucede lo mismo que con la anterior, se trata de una aplicación para simular robots, sin embargo, está basada en robots manipuladores.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Fuente: 11 pto

Con formato: Fuente: 11 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

6.4. Estructura de la memoria

~~Los capítulos que se abarcan a continuación son:~~ A continuación se describe brevemente la estructura del resto del documento:

En el Capítulo 2, **Robótica móvil**, se comienza explicando qué es un robot y en específico un vehículo con ruedas y diferentes configuraciones del mismo. A ~~continuación~~ continuación, se entra en detalle en el funcionamiento del direccionamiento diferencial y se realiza el cálculo de las ecuaciones necesarias para simular un robot de este tipo. ~~Finalmente~~ Finalmente, se le aplica una navegación autónoma a un vehículo con ruedas con direccionamiento diferencial.

En el Capítulo 3, **Entorno tecnológico**, se ~~informa~~ habla mediante una descripción sobre las diferentes tecnologías usadas: Qt, OpenGL, freeglut y GLM y GLM. Además De estas se realiza una explicación sobre su utilización en el proyecto. También se explica de manera breve el cauce gráfico y la etapa geométrica con un poco más de detalle. Además de estas también se explican otras tecnologías menos importantes para la aplicación, pero importantes para el desarrollo del proyecto como son Trello y GitHub ~~el proyecto~~.

En el Capítulo 4, **Descripción de la aplicación**, ~~en este apartado~~, se expone la aplicación al completo, comenzando por detalles sobre la simulación y visualización del robot, pasando por la explicación de la interfaz y finalizando con casos de uso de la aplicación.

Finalmente, en el Capítulo 5, **Conclusiones**, se presentan tanto las conclusiones del trabajo como su posible ampliación y mejora.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Negrita, Cursiva

Con formato: Fuente: Negrita, Cursiva

Con formato: Fuente: Negrita, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Capítulo 2 Robótica móvil

1. Vehículos con ruedas Robótica móvil

Los robots móviles son robots que, gracias a los progresos entre la inteligencia artificial y los robots físicos, son capaces de moverse a través de cualquier entorno físico. Estos normalmente son controlados por software y usan sensores y otros equipos para identificar la zona de su alrededor

Los robots móviles se pueden diferenciar en dos tipos, autónomos y no autónomos. Los robots móviles autónomos pueden explorar el entorno sin ninguna directriz externa a él, sin embargo, los robots móviles no autónomos necesitan algún tipo de sistema de guía para moverse. Los vehículos con ruedas son una solución simple y eficiente para conseguir movilidad sobre terrenos duros y libres de obstáculos, con los que se permite conseguir velocidades más o menos altas.

Su limitación más importante es el deslizamiento en la impulsión, además dependiendo del tipo de terreno, puede aparecer deslizamiento y vibraciones en el mismo. Como se ha dicho en la definición, estos vehículos son eficientes en terrenos duros y libres de obstáculos, por lo tanto en terrenos blandos son poco eficientes.

Otro problema que tienen este tipo de vehículos se haya en que no es posible modificar la estabilidad para adaptarse al terreno, excepto en configuraciones muy especiales, lo que limita los terrenos sobre los que es aceptable el vehículo.

A continuación se pasa a calcular las ecuaciones del modelo cinemático y discreto.

Según se ve en la figura 4, se supone un sistema de referencia (C) y un sistema (L) con origen en el punto de guiado del vehículo y eje Y_L en la dirección del eje longitudinal del vehículo.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: TFG titulo 2

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

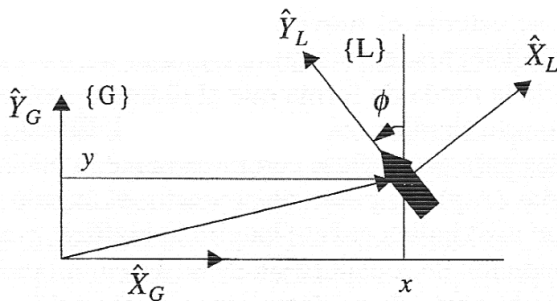


Figura 4. Modelo cinemático.

Por lo tanto, si el vehículo tiene una velocidad de desplazamiento v y de rotación w con respecto a $\{L\}$, con respecto a $\{G\}$ la velocidad es:

$$v_x = v \cdot \sin(\phi)$$

$$v_y = v \cdot \cos(\phi) \quad (1)$$

$$v_\phi = w$$

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

$$v_x = \frac{dx}{dt} = \frac{x(t+1) - x(t)}{\Delta t} \quad (2)$$

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:

$$x(t+1) = x(t) + v_x \cdot \Delta t \quad (3)$$

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Letra normal TFG, Izquierda

Con formato: Justificado

Con formato: Punto de tabulación: No en 16 cm

Con formato: Punto de tabulación: No en 16 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva



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

$$x(t+1) = x(t) + v_x \cdot \Delta t$$

$$y(t+1) = y(t) + v_y \cdot \Delta t \quad (4)$$

$$\varphi(t+1) = \varphi(t) + v_\varphi \cdot \Delta t$$

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

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

1. Diferentes configuraciones

2. Vehículos con ruedas

Los vehículos con ruedas son un tipo de robot móvil los cuales proporcionan una solución simple y eficiente para conseguir movilidad sobre terrenos duros y libres de obstáculos, con los que se permite conseguir velocidades más o menos altas.

Su limitación más importante es el deslizamiento en la impulsión, además dependiendo del tipo de terreno, puede aparecer deslizamiento y vibraciones en el mismo. Como se ha dicho en la definición, estos vehículos son eficientes en terrenos duros y libres de obstáculos, por lo ~~tant~~ tanto, en terrenos blandos son poco eficientes.

Otro problema que tienen este tipo de vehículos se ~~halla~~ ya en que no es posible modificar la estabilidad para adaptarse al terreno, excepto en configuraciones muy especiales, lo que limita los terrenos sobre los que es aceptable el vehículo.

Los vehículos con ruedas emplean distintos tipos de locomoción que les da unas características y propiedades diferentes entre ellos en cuanto a eficiencia energética, dimensiones, maniobrabilidad y carga útil. A ~~continuación~~ continuación, se pasa a mencionar algunas de estas configuraciones:

- **Ackerman:** Es el habitual de los vehículos de cuatro ruedas. Las ruedas delanteras son las que se ocupan del giro, sin ~~embargo~~ embargo, la rueda interior gira un poco más que la exterior lo que hace que se elimine el deslizamiento. El centro de rotación del vehículo se haya en el corte de las prolongaciones de las ruedas delanteras y las ruedas traseras como se muestra en la figura 54.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Letra normal TFG, Sin viñetas ni numeración

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

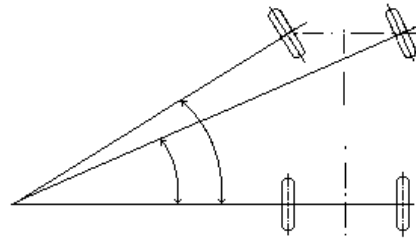


Figura 45. Configuración ackerman y centro de rotación.

- Triciclo:** Se compone por tres ruedas, una delantera central y dos traseras. La rueda delantera actúa tanto para tracción como para direccionamiento, las ruedas traseras son pasivas. Tiene una mayor maniobrabilidad que la configuración anterior, sin embargo, posee una menor estabilidad sobre terrenos difíciles. El centro de gravedad tiende a perderse cuando se desplaza por una pendiente, perdiendo tracción. El centro de rotación se puede calcular igual que en la configuración anterior, es decir, prolongando el eje de la rueda delantera y las traseras y este se encuentra en el punto de corte.

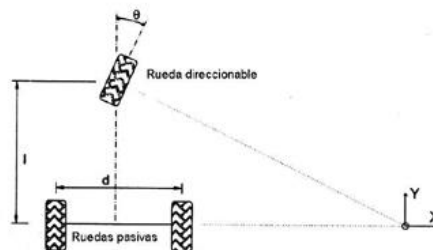


Figura 56. Configuración de triciclo y centro de rotación.

- Skid Steer:** Varias ruedas a cada lado del vehículo que actúan de forma simultánea. La dirección del vehículo resulta de combinar las velocidades de las ruedas izquierdas con las de la derecha.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

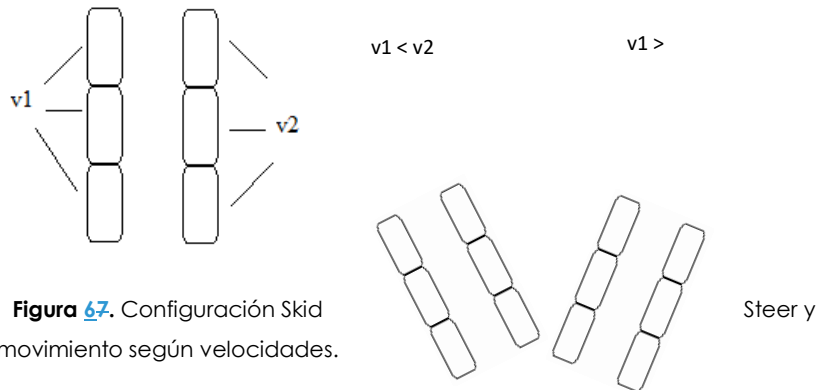


Figura 67. Configuración Skid movimiento según velocidades.

- **Pistas de deslizamiento:** Funcionalmente análogo a la configuración Skid Steer. Tanto la impulsión como el direccionamiento se realiza mediante pistas de desplazamiento, estas pistas actúan de forma similar a como lo harían ruedas de gran diámetro. Esta configuración es útil en terrenos irregulares.



Figura 78. Pistas de desplazamiento.

- **Síncronas:** Se trata de una configuración en la que todas las ruedas actúan de forma simultánea y, por lo tanto, giran de forma síncrona.
- **Direccionamiento diferencial:** Esta configuración es la que utiliza el robot sigue líneas. 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 9 se muestra una imagen de dicho esquema. A continuación se entra en detalle sobre su funcionamiento. En este sistema las ruedas se sitúan de forma paralela y no realizan giros. El direccionamiento diferencial viene dado por la diferencia de velocidades de las ruedas laterales. La tracción se consigue con

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Sin Negrita

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva



esas mismas ruedas. Adicionalmente, existen una o más ruedas para el soporte.
En la figura 10 se muestra una imagen de dicho esquema. ↗

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

2. 3. Modelo físicoD direccionamiento diferencial

Para poder realizar la simulación del robot y pintarlo se debe conocer primeramente el estado del sistema. El sistema de un robot viene dado por su posición y orientación. Puesto que el robot se va a mover se necesita saber el estado en los diferentes instantes, por lo tanto, se debe definir un modelo de cambios de estado, es decir, una ecuación que a partir del estado actual (t) y unas entradas se pase al estado en el siguiente instante ($t+1$). En la ecuación X se puede observar la ecuación que define el sistema, siendo \bar{s} el estado y \bar{r} las entradas. Asimismo, en esta misma ecuación se muestra la ecuación del estado de un robot móvil, definida por su posición $x(t)$ $z(t)$ y su orientación $\varphi(t)$ en ese instante:

$$\begin{aligned}\bar{s}(t+1) &= f(\bar{s}(t), \bar{r}(t)) \\ \bar{s} &= (x(t), z(t), \varphi(t))\end{aligned}\quad (X)$$

Sin embargo, la ecuación anterior es genérica para robots móviles, para obtener la ecuación de cambio de estado para un robot con direccionamiento diferencial se debe definir las entradas como las velocidades de cada rueda como se explica en el punto anterior. Por lo tanto, la ecuación de las entradas resultante para un robot con direccionamiento diferencial, siendo w_i la velocidad de la rueda izquierda y w_d la velocidad de la rueda derecha, es la siguiente:

$$\bar{r} = (w_i(t), w_d(t))\quad (X)$$

En la figura 8 se pueden observar los diferentes datos de entrada y del sistema puestos sobre un robot con direccionamiento diferencial.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Espacio Después: Automático, Sin viñetas ni numeración

Con formato: TFG título 2, Espacio Después: 5 pto

Código de campo cambiado

Código de campo cambiado

Código de campo cambiado

Con formato: Fuente: Cursiva

Código de campo cambiado

Código de campo cambiado

Código de campo cambiado

Código de campo cambiado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

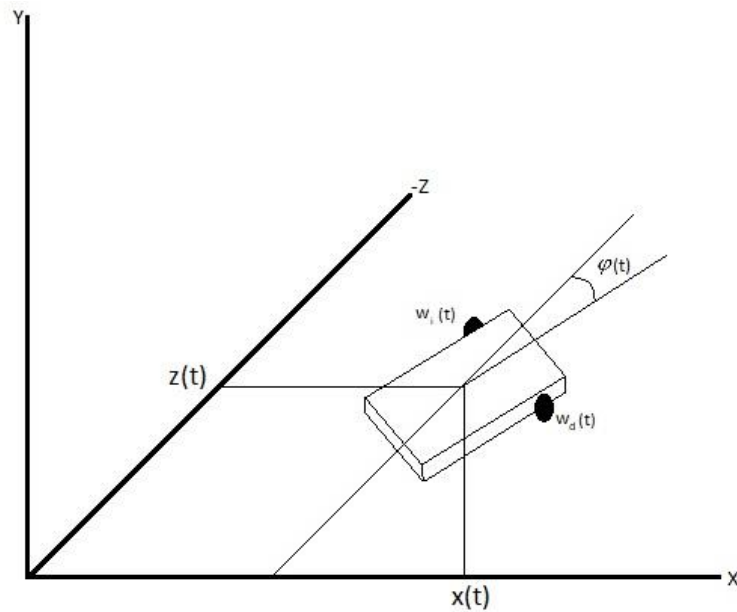


Figura 8. Sistema 3D de un robot con direccionamiento diferencial.

Una vez obtenido esto vamos a realizar el cálculo de las ecuaciones específicas. A continuación se pasa a calcular las ecuaciones del modelo cinemático y discreto.

Según se ve en la figura 94, se supone un sistema de referencia $\{G\}$ y un sistema $\{L\}$ con origen en el punto de guiado del vehículo y eje Y_L en la dirección del eje longitudinal del vehículo.

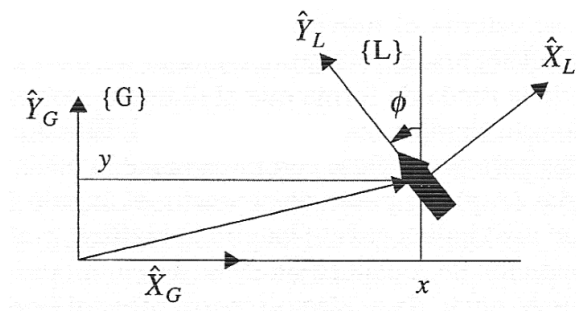


Figura 94. Modelo cinemático.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Centrado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Por lo tanto, si el vehículo tiene una velocidad de desplazamiento v y de rotación w con respecto a $\{L\}$, con respecto a $\{G\}$ la velocidad es:

$$v_x = -v \cdot \sin(\varphi)$$

$$v_y = v \cdot \cos(\varphi) \quad (1)$$

$$v_x = -v \cdot \sin(\varphi)$$

$$v_y = v \cdot \cos(\varphi) \quad (1) \quad v_\varphi = w$$

$$v_\varphi = w$$

Código de campo cambiado

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

$$v_x = \frac{dx}{dt} = \frac{x(t+1) - x(t)}{\Delta t} \quad v_x = \frac{dx}{dt} = \frac{x(t+1) - x(t)}{\Delta t} \quad (2)$$

Código de campo cambiado

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:

$$x(t+1) = x(t) + v_x \cdot \Delta t \quad (3)$$

$$x(t+1) = x(t) + v_x \cdot \Delta t \quad (3)$$

Código de campo cambiado

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

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

$$\begin{aligned}x(t+1) &= x(t) + v_x \cdot \Delta t \\y(t+1) &= y(t) + v_y \cdot \Delta t \quad (4) \\ \varphi(t+1) &= \varphi(t) + v_\varphi \cdot \Delta t\end{aligned}$$

$$x(t+1) = x(t) + v_x \cdot \Delta t$$

$$y(t+1) = y(t) + v_y \cdot \Delta t \quad (4)$$

$$\varphi(t+1) = \varphi(t) + v_\varphi \cdot \Delta t$$

Por tanto, si se dispone de las coordenadas (v_x, v_y, v_φ) 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 9 se muestra una imagen de dicho esquema. Nótese que para especificar la configuración hay que indicar los valores de las tres variables (x, z, φ) , siendo las variables de control las velocidades de las ruedas laterales.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Código de campo cambiado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

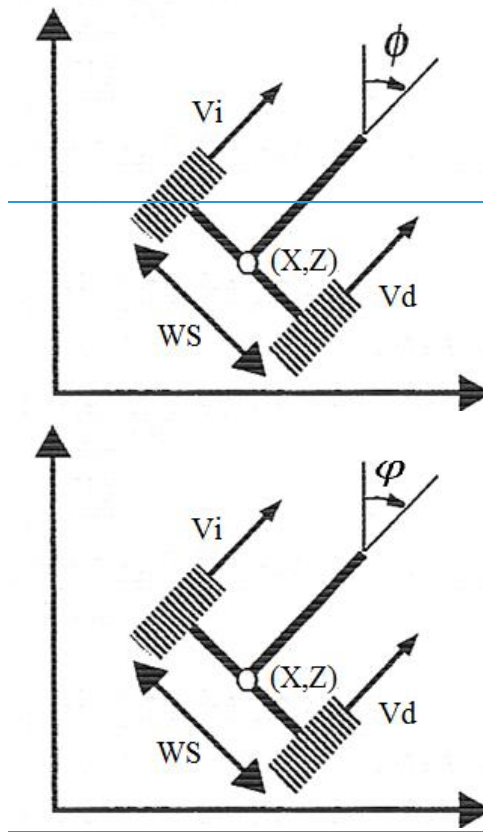


Figura 109. Locomoción mediante guiado diferencial.

Sean w_i y w_d las velocidades de giro de las ruedas izquierda y derecha, respectivamente. Si el radio de la rueda es WR , las velocidades lineales correspondientes son $v_i = w_i \cdot WR$ y $v_d = w_d \cdot WR$. En este caso, la velocidad lineal y velocidad angular correspondientes en el modelo vienen dadas por:

$$v = \frac{v_d + v_i}{2} = \frac{(v_d + v_i) \cdot WR}{2}$$

$$w = \frac{v_d - v_i}{WS} = \frac{(w_d - w_i) \cdot WR}{WS}$$

(5) $v = \frac{v_d + v_i}{2} = \frac{(v_d + v_i) \cdot WR}{2}$

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Fuente: (Predeterminada) Century Gothic, Sin Cursiva

Código de campo cambiado

Con formato: Fuente:

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

$$w = \frac{v_d - v_i}{WS} = \frac{(w_d - w_i) \cdot WR}{WS} \quad (5)$$

Sustituyendo estas expresiones en las obtenidas a partir de la Figura 24, se obtienen las velocidades de las coordenadas del robot en el sistema {G} a partir de la velocidad de giro de cada rueda:

$$v_x = \frac{(w_d + w_i) \cdot WR}{2} \cdot \sin(\varphi) = -(w_d + w_i) \cdot \frac{WR \cdot \sin(\varphi)}{2}$$

$$v_z = \frac{(w_d + w_i) \cdot WR}{2} \cdot \cos(\varphi) = (w_d + w_i) \cdot \frac{WR \cdot \cos(\varphi)}{2} \quad (6)$$

$$v_\varphi = \frac{(w_d - w_i) \cdot WR}{WS} = (w_d - w_i) \cdot \frac{WR}{WS}$$

$$v_z = \frac{(w_d + w_i) \cdot WR}{2} \cdot \cos(\varphi) = (w_d + w_i) \cdot \frac{WR \cdot \cos(\varphi)}{2} \quad (6)$$

$$v_\varphi = \frac{(w_d - w_i) \cdot WR}{WS} = (w_d - w_i) \cdot \frac{WR}{WS}$$

Finalmente, utilizando el modelo discreto, se obtiene:

$$x(t+1) = x(t) - (w_d + w_i) \cdot \frac{WR \cdot \sin(\varphi(t))}{2} \cdot \Delta t$$

$$z(t+1) = z(t) + (w_d + w_i) \cdot \frac{WR \cdot \cos(\varphi(t))}{2} \cdot \Delta t \quad (7) \quad x(t+1) = x(t) - (w_d + w_i) \cdot$$

$$\varphi(t+1) = \varphi(t) + (w_d - w_i) \cdot \frac{WR}{WS} \cdot \Delta t$$

$$\frac{WR \cdot \sin(\varphi(t))}{2} \cdot \Delta t$$

$$z(t+1) = z(t) + (w_d + w_i) \cdot \frac{WR \cdot \cos(\varphi(t))}{2} \cdot \Delta t \quad (7)$$

$$\varphi(t+1) = \varphi(t) + (w_d - w_i) \cdot \frac{WR}{WS} \cdot \Delta t$$

Estas últimas ecuaciones se utilizarán en el proyecto para el cálculo de la posición del robot y por lo tanto para el cálculo de la matriz model mediante los métodos de la

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Código de campo cambiado

Con formato: Fuente:

Con formato: Fuente: (Predeterminada) Century Gothic, Sin Cursiva

Código de campo cambiado

Con formato: Fuente:

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva



librería GLM que en este caso estos métodos realizan los mismos cálculos que los métodos de OpenGL: `translate` (para el cálculo de la matriz de translación, cuyo método en OpenGL sería `glTranslate3f`) y `rotate` (para el cálculo de la matriz de rotación, cuyo método en OpenGL sería `glRotate3f`).

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

3. 4. Navegación autónoma

El robot sigue líneas que se ha implementado realiza su movimiento de manera autónoma, se coloca el robot sobre un fondo blanco con una línea negra que representa el circuito, como se muestra en la figura 11, y este deberá recorrer el circuito sin salirse del mismo. Esto se puede realizar gracias a dos sensores que son implantados en la parte delantera del robot los cuales son responsables de la detección de la línea del circuito. En función de lo que estos sensores recojan (están sobre el circuito o no) el robot realiza cambios en la velocidad de sus ruedas resultando en un movimiento recto, rotatorio hacia la izquierda o rotatorio hacia la derecha.

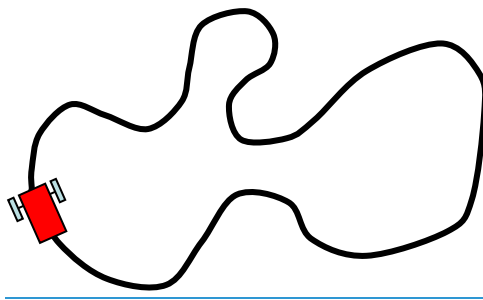


Figura 11. Colocación del robot en un circuito negro sobre fondo blanco.

Los sensores que se usan en este tipo de robots son sensores CNY70, los cuales se muestran en la Figura 12.

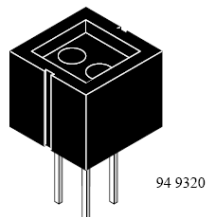


Figura 12. Sensor CNY70.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sin viñetas ni numeración

Con formato: TFG título 2

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Estos son sensores ópticos reflexivos de corto alcance basados en un diodo de emisión de luz infrarroja y un receptor formado por un fototransistor que ambos apuntan en la misma dirección, esta estructura de forma simplificada se puede observar en la figura 13. Cuando el sensor se haya sobre una línea negra la luz es absorbida y el fototransistor envía una señal (ya sea alta o baja dependiendo del montaje del sensor), sin embargo, cuando se haya sobre fondo blanco la luz es reflejada y por lo tanto el fototransistor envía la señal contraria a la enviada al estar sobre negro.

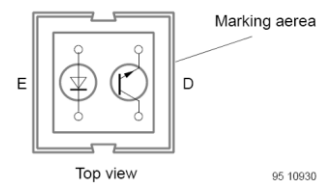
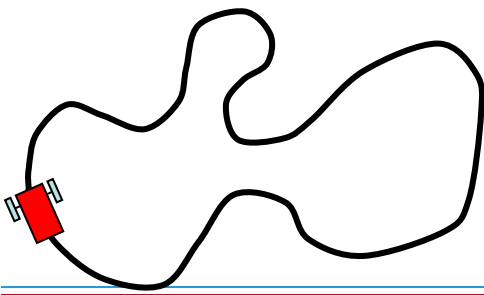


Figura 13. Estructura simplificada del sensor CNY70.

El robot sigue líneas que se ha implementado realiza su movimiento de manera autónoma, se coloca el robot sobre un fondo blanco con una línea negra que representa el circuito y este deberá recorrer el circuito sin salirse del mismo. Esto se puede realizar gracias a dos sensores que son implantados en la parte posterior del robot los cuales son responsables de la detección de la línea del circuito. En función de lo que estos sensores recojan (están sobre el circuito o no) el robot realizará cambios en la velocidad de sus ruedas resultando en un movimiento recto, rotatorio hacia la izquierda o rotatorio hacia la derecha.



Los sensores que se implementan en este tipo de robots son sensores CNY70 los cuales son sensores ópticos reflexivos de corto alcance basados en un diodo de emisión de luz infrarroja y un receptor formado por un fototransistor que ambos apuntan en la misma dirección. Cuando el sensor se haya sobre una línea negra la luz es absorbida

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

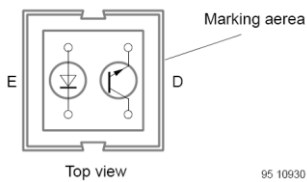
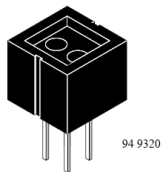
Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

y el fototransistor envía una señal (ya sea alta o baja dependiendo del montaje del sensor), sin embargo cuando se haya sobre fondo blanco la luz es reflejada y por lo tanto el fototransistor envía la señal contraria a la enviada al estar sobre negro.

La simulación de estos sensores se basa en el conocimiento de la posición de los sensores en todo momento y de los puntos que conforman el circuito, pudiendo así comprobar si cualquiera de los dos sensores está situado sobre el circuito.



Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Capítulo 3 Entorno tecnológico

Para el desarrollo del trabajo se han utilizado diferentes tecnologías: -bla bla bla...



Logo C++

Qt para la interacción con el usuario.



OpenGL para implementar el cauce gráfico.



Glm para el cálculo de matrices.



Freeglut para crear modelos 3D.

1. OpenGLQt

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 11 pto, Sin Negrita

Con formato: Fuente: 11 pto, Sin Negrita

Con formato: Letra normal TFG, Sin viñetas ni numeración

Con formato: Letra normal TFG, Sin viñetas ni numeración

Con formato: Sin viñetas ni numeración

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Qt es un framework de desarrollo de aplicaciones multiplataforma para ordenador, embebido y móvil que en gran parte se suele utilizar para programas que utilicen interfaz gráfica.

Internamente se utiliza C++ con alguna extensión para funciones como Signals y Slots, por lo tanto, se utiliza orientación a objetos. Puesto que se utiliza C++, en los proyectos se encontrarán archivos de 4 tipos:

- .pro: Solo habrá un archivo .pro en el proyecto. Este archivo contiene toda la información necesaria para realizar la build de la aplicación.
- .h: Estos archivos incluyen la declaración de variables y las cabeceras de las funciones de la clase correspondiente, tanto pública como privada.
- .cpp: Son los archivos en los cuales se sitúa el código fuente de la clase.
- .ui: Este archivo es el correspondiente a la interfaz gráfica.

Para el desarrollo de la interfaz gráfica se puede escribir en C++ utilizando el módulo Widget, además de esto, Qt tiene una herramienta gráfica llamada Qt Designer que es un generador de código basado en Widgets.

A continuación se muestra el "Hello World" en qt para poder observar un ejemplo sencillo. La estructura se puede ver en la figura 10. Como se puede observar, main.cpp será el que inicialice el programa, y mainwindow será el que se encargue de contener el código para mostrar en la aplicación "Hello World".



Figura 10. Estructura "Hello World".

El fichero HelloWorld.pro es el que contiene la configuración del proyecto.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

En primer lugar, se crean dos widgets, un botón y un espacio de texto, en el fichero `mainwindow.ui` (se puede utilizar la herramienta Qt Designer como se muestra en la figura 11) y se les nombra como se desea, en este caso el botón se llama "saluda" y el cuadro de texto "textEdit".

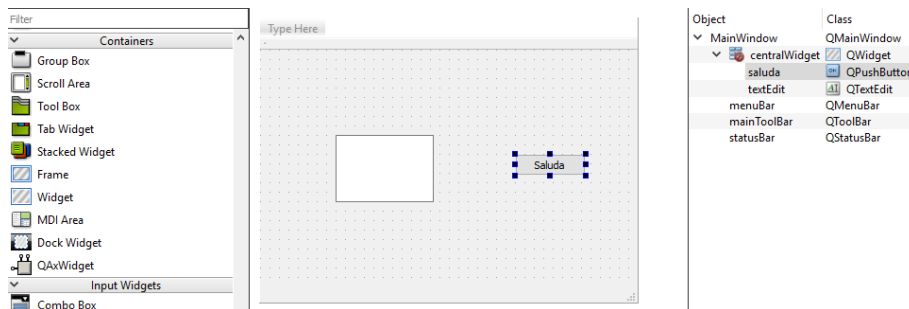


Figura 11. MainWindow.ui.

El fichero `mainwindow.cpp` contiene el método `on_saluda_clicked()` que se llamará cuando se pulse el botón "saluda". Este contiene el código que accederá al cuadro de texto y escribirá en él "Hello World", este método se puede observar en la figura 12.

```
void MainWindow::on_saluda_clicked()
{
    ui->textEdit->setText("hello world");
}
```

Figura 12. Método que escribirá "Hello World" al pulsar el botón.

Solo se necesitará añadir la cabecera del método en `mainwindow.h` y el programa estará terminado.

Para poder implementar el cauce gráfico, que se explica al final de este punto y el siguiente, existen diferentes APIs que se pueden utilizar como pueden ser Direct3D (para Windows), Mantle (para tarjetas AMD), Vulkan (basado en Mantle y multiplataforma) o OpenGL. Se decidió utilizar OpenGL puesto que es la librería más conocida, pública y multiplataforma, además es fácil integrarla en la mayoría de los proyectos.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

OpenGL es una API multilenguaje y multiplataforma que se utiliza para el desarrollo de aplicaciones en las que se utilicen gráficos 2D y 3D. Este te proporciona funciones con las cuales podrás realizar imágenes, animaciones, juegos, simulaciones, etc. OpenGL te permite realizar entre otras muchas cosas:

- Construir formas geométricas a partir de las primitivas que este te proporciona.
- Ubicar los objetos en la escena.
- Ubicar el punto desde el que se visualiza la escena.
- Poner color o texturas.
- Crear luces.
- Realizar la rasterización.

Las funciones básicas y de mayor importancia que nos proporciona OpenGL con Qt son las siguientes:

- **initializeGL():** Esta función se ejecuta una sola vez y antes que las otras dos funciones. Por lo tanto, se utiliza para inicializar y configurar todo lo necesario para la utilización de OpenGL u otros.
- **paintGL():** Esta función es la que renderiza la escena de OpenGL y se llama siempre que el widget necesite ser actualizado. Además, este método será en el cual se modificará la posición de la cámara y la posición del robot, es decir, las matrices view y model. Gracias a los cambios en esta última matriz se realiza la animación de movimiento del robot.
- **resizeGL(int w, int h):** En este método se debe configurar el viewport (los parámetros de entrada w y h son el ancho y el alto respectivamente de la zona donde se podrá visualizar el código desarrollado en OpenGL), el tipo de vista (perspectiva u ortográfica), etc. Es llamado por primera vez cuando el widget se crea (siempre después de initializeGL) y siempre que el widget sea reescalado. Este método es el responsable de crear la matriz projection.

OpenGL tiene diferentes versiones que siguen pudiendo ser utilizadas hoy en día, a continuación, realizaré una muy breve explicación de las más relevantes:

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

- **OpenGL 1.X:** en las diferentes actualizaciones fueron haciendo extensiones al núcleo de la API.
- **OpenGL 2.X:** se incorporó GLSL (OpenGL Shading Language), con el cual se podía programar las etapas de transformación y rasterizado del cauce gráfico.
- **OpenGL 3.X:** en la primera etapa (OpenGL 3.0) se nombra ciertas funciones como obsoletas, que serán marcadas para ser eliminadas en futuras versiones (la mayoría de ellas en la versión 3.1).
- **OpenGL 4.X:** actualmente la última versión de OpenGL (4.6) lanzada este mismo año en 2017. Se añaden una gran cantidad de funcionalidades.

A pesar de esto OpenGL tiene un problema, no es fácil crear una interfaz con la que un usuario pueda interactuar, por ello y para solucionar este problema se utiliza Qt en este proyecto.

El cauce gráfico es el conjunto de transformaciones y procesados de la imagen que se realiza a los elementos que definen la escena hasta llegar a la imagen resultante final. Actualmente la generación de estas imágenes sigue una serie de pasos ya definidos.

El cauce gráfico se divide en varios pasos o etapas que se conectan entre ellas, es decir la salida de la primera será la entrada de la segunda, y así sucesivamente. En la figura 14 podemos observar las diferentes etapas y como se conectan como hemos citado anteriormente.

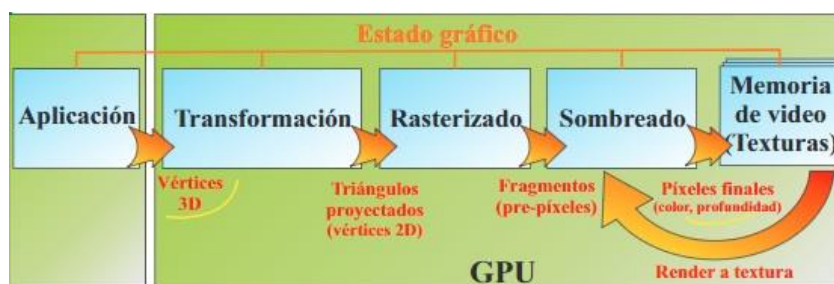


Figura 14. Etapas del cauce gráfico.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Las tres etapas principales del cauce gráfico son: transformación/geométrica, rasterizado y sombreado.

La integración de estas herramientas (OpenGL y Qt) se realiza de forma muy sencilla puesto que Qt tiene la API de OpenGL y por lo tanto solo hace falta decirle al IDE que estás utilizando que vas a utilizar esas librerías incluyendo lo siguiente "#include <QOpenGLWidget>". Además en caso de que se esté utilizando como IDE Qt Creator se debe ir al fichero .ui y en el widget en el que se muestra la simulación, promoverle a la clase que tenga el código de la simulación. En caso de utilizar librerías externas se debe incluir en el fichero .pro la palabra reservada LIBS seguido del path donde se encuentre la librería: LIBS += pathDeMiLibreria y en la clase en la que se utiliza realizar el include correspondiente.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

2. Etapa Geométrica

En esta etapa se transforma las coordenadas de los vértices de los objetos de su sistema local a su proyección en 2D. Esto se realiza mediante cálculos de matrices en los que una vez se han realizado todos los cálculos se consiguen las coordenadas en la proyección. Si se quiere utilizar en 3D, las matrices que se utilizan para los cálculos son matrices 4x4. Para obtener las coordenadas donde se encuentra cada punto del objeto dentro de la escena es necesario calcular la matriz model-view-projection (MVP), como su nombre indica, esta matriz viene dada por la multiplicación de tres matrices distintas las cuales realizan cálculos para la obtención de diferentes funcionalidades, es decir, la ecuación que se implementa en esta etapa es

$$v' = M_{projection} * M_{view} * M_{model} * v$$

Donde v' es el vector de posición resultante del objeto en cuestión y v el vector de coordenadas en el instante anterior.

Para la creación de los modelos 3D se utiliza la librería freeglut. Estos modelos son los que contienen las coordenadas, es decir, cada vértice de los modelos es un vector de coordenadas v .

En lo referido a la matriz MVP, OpenGL no ofrece todo el control que se desea, por ello se ha utilizado la librería GLM para realizar los cálculos pertinentes sobre esta matriz. Se ha optado por la utilización de GLM como librería para el cálculo de la matriz MVP puesto que es la librería recomendada por OpenGL y por lo tanto es la más apta para este funcionamiento. A continuación, se pasa a explicar las diferentes matrices por separado y las funciones utilizadas de la librería GLM en cada una de ellas.

- **(P) Projection:** Esta matriz define como se realiza la proyección, poniendo en esta matriz si la proyección es en perspectiva u ortográfica. Además, permite realizar el clipping (desactivar la renderización) de los vértices que no son visibles.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Código de campo cambiado

Código de campo cambiado

Código de campo cambiado

Código de campo cambiado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

La diferencia principal entre estos dos tipos de vistas (proyección y ortográfica) se basa en que la vista en perspectiva mantiene las dimensiones reales de los objetos si nos acercamos o nos alejamos de ellos, por lo que se acerca más a la vista de una persona. En la figura 15 y 16 podemos observar como dos cámaras, una de cada tipo de vista, generan el plano de proyección a partir de los puntos de los objetos del espacio.

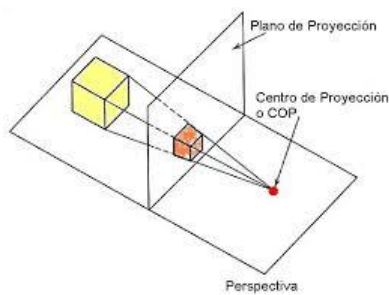


Figura 15. Vista en perspectiva.

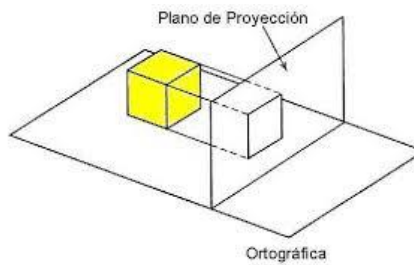


Figura 16. Vista ortográfica.

Esto se debe a que en una vista ortográfica los puntos de los objetos se proyectan de forma perpendicular al plano de proyección, mientras que en la vista en perspectiva los puntos se dirigen al punto central de la cámara o centro de proyección.

Las funciones de GLM que se utilizan en el cálculo de esta matriz son `perspective(fov, aspectRatio, near, far)` para el cálculo de la matriz en perspectiva y `ortho(left, right, bottom, top, near, far)` para el cálculo de la matriz en ortográfica.

- **(V) View:** Esta matriz hace las funciones de cámara, necesitando para poder usar esta matriz la posición, el punto hacia el que mira y la orientación de la cámara. Además, esta matriz no solo sitúa la cámara, sino que también realiza los cálculos para simular los movimientos que la cámara realizaría. Puesto que no se puede mover la cámara, lo que se realiza es mover el mundo en concordancia con esto, es decir se aplica al mundo la inversa del movimiento que la cámara realizaría, consiguiendo así el efecto de que la cámara se está

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

moviendo. Por consiguiente, la posición de la cámara también es ilusoria puesto que la cámara siempre está en un punto fijo.

La función de GLM que se utiliza para esta matriz es `lookAt(eye, center, up)` cuyos parámetros indican `eye` la posición de la cámara, `center` el punto hacia donde mira y `up` la orientación.

Al iniciarse esta aplicación se dibuja el circuito y se hace un cálculo para posicionar automáticamente la cámara sobre el mismo. Sabiendo que la cámara se sitúa mirando hacia $y=0$ y la parte de arriba de la cámara está situada hacia $-z$, como se muestra en la figura 17, se buscan los puntos del circuito más externos, es decir, el punto situado más a la izquierda (x menor), más a la derecha (x mayor), más arriba (z mayor) y más abajo (z menor).

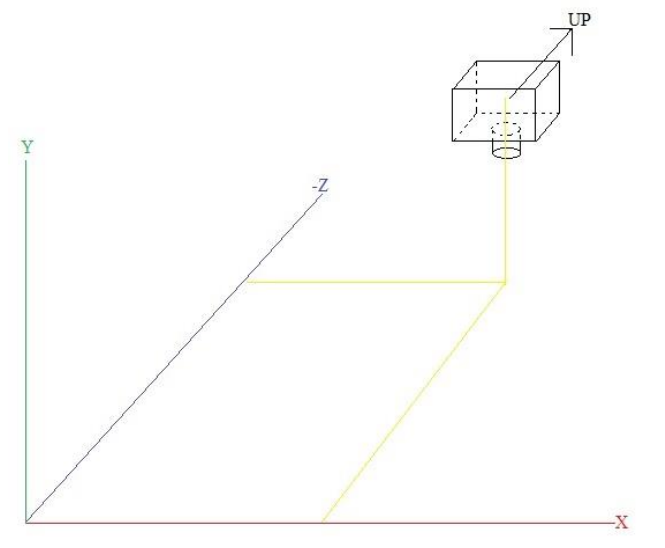


Figura 17. Posición y orientación de la cámara.

Una vez encontrados se crea un vector de 3 posiciones (zona de memoria donde caben 3 datos) en el cual guardaremos la posición donde más adelante situaremos la cámara. En este vector se guarda en la primera posición (x) el punto intermedio entre el punto situado más a la izquierda y el punto situado más a la derecha. A la hora de guardar la posición 3 (z) se realiza

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sangría: Izquierda: 2,52 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

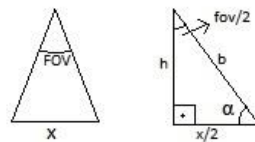
Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

exactamente lo mismo, pero con la parte superior e inferior, es decir se calcula y se guarda el punto medio entre el punto más superior y el punto más inferior. Para poder saber la altura a la que se debe situar la cámara para que pueda ver todo el circuito, primero se comprueba cual contiene una mayor distancia si el eje x (es decir el punto más a la derecha menos el punto más a la izquierda) o el eje z (es decir el punto más arriba menos el punto más abajo). Una vez obtenida la mayor distancia se calcula mediante trigonometría la altura necesaria, como se puede ver en la figura 18.



$$\alpha = 180 - \text{FOV}/2 - 90$$

$$\frac{h}{\sin(\alpha)} = \frac{b}{\sin(90^\circ)} = \frac{x/2}{\sin(\text{FOV}/2)}$$

$$h = \frac{x/2 * \sin(\alpha)}{\sin(\text{FOV}/2)}$$

Figura 18. Trigonometría utilizada para el cálculo de la altura de la cámara.

Con esto la cámara se situará automáticamente en el centro de cualquier circuito que se introduzca y a una distancia a la cual se pueda ver el circuito completamente.

Una vez tenemos la posición de la cámara, dependiendo de que opción haya elegido el usuario se hacen los cálculos para la proyección en perspectiva u ortográfica, esto implica que se cambia también la posición de la cámara obtenida anteriormente. En caso de que se opte por una visualización en ortográfica la cámara se situara justo encima del circuito, sin embargo, en caso de que se opte por una visualización en perspectiva se colocara la

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sangría: Izquierda: 2,52 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

cámara a 45° respecto a la posición inicial (justo encima del circuito) para que sea posible observar los modelos 3D y la perspectiva de la imagen. Como añadido, se ha implementado la funcionalidad de zoom en ambas proyecciones el cual se puede utilizar con la rueda del ratón.

- **(M) Model:** Esta matriz realiza una transformación de la posición en el modelo a la posición global. Normalmente es una combinación de tres posibles movimientos: trasladar, escalar y rotar. Cada uno de estos movimientos vienen dados por matrices, las cuales se multiplican unas sobre otras, comenzando por la matriz identidad, dando así una única matriz que será la matriz model. En esta aplicación se utiliza la matriz de translación y la de rotación en Y.

Las funciones de GLM son *translate(m,v)*, donde *m* es la matriz anterior y *v* es el vector de tres posiciones (x,y,z) que indica cuanto y hacia donde se mueve; y la función *rotate(m,angle,axis)* donde *m* es la matriz anterior, *angle* es el ángulo que se rota y *axis* es un vector de tres posiciones que indica en que eje rota.

Las matrices específicas que se utilizan en este proyecto se pueden observar en las figuras 19 y 20. En la figura 19 se puede observar la matriz de translación en X y Z que es donde se puede mover el robot.

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 19. Matriz de translación en X y Z.

En la figura 20 se puede ver la matriz de rotación en Y donde φ representa el ángulo que rota el robot.

$$\begin{bmatrix} \cos \varphi & 0 & \sin \varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \varphi & 0 & \cos \varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Código de campo cambiado

Código de campo cambiado

Código de campo cambiado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Figura 20. Matriz de rotación en Y con un ángulo φ .

En ambas matrices simplemente se debe sustituir las variables X,Y y φ con los valores correspondientes y se obtienen las matrices que se utilizan en la matriz model. A pesar del control que ofrece OpenGL sobre el cauce gráfico, en esta aplicación se ha necesitado llegar más en detalle sobre la matriz MVP, siendo exactos en la matriz Model (M), puesto que se necesita tener en todo momento las coordenadas de los sensores del robot para realizar los cálculos sobre las colisiones de los mismos con el circuito.

Se ha optado por la utilización de GLM como librería para el cálculo de la matriz MVP puesto que es la librería recomendada por OpenGL y por lo tanto es la más apta para este funcionamiento.

En este proyecto GLM se utiliza, como se ha dicho anteriormente, para el cálculo de la matriz MVP. Esta matriz se incluirá en la pila proporcionada por OpenGL y este realiza los cálculos pertinentes. Gracias a esto, y puesto que tenemos control total sobre la matriz Model, se puede realizar los cálculos para obtener las coordenadas de los sensores.

2. OpenGL



3. Qt

OpenGL no puede crear ventanas y no tiene una forma sencilla de interactuar con el usuario por lo que es necesario cubrir estas necesidades con otro software. Algunos de los diferentes softwares que pueden hacer esto son sdl y Qt. Se ha optado por Qt puesto que tiene una interacción con el usuario más sencilla y es fácil incluir en un proyecto OpenGL.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Negrita

Con formato: Centrado

Con formato: Fuente: Negrita

Código de campo cambiado

Con formato: Sangría: Izquierda: 1,25 cm

Código de campo cambiado

Con formato: Letra normal TFG, Sangría: Izquierda: 1,25 cm, Sin viñetas ni numeración

Con formato: Sangría: Izquierda: 1,25 cm

Con formato: Justificado, Sangría: Izquierda: 1,25 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Qt es un framework de desarrollo de aplicaciones multiplataforma para PC que se suele utilizar en gran parte para programas que utilicen interfaz gráfica.

Internamente se utiliza C++ con alguna extensión para funciones como Signals y Slots, por lo tanto, se utiliza orientación a objetos. Puesto que se utiliza C++, en los proyectos se encontrarán archivos de 4 tipos:

- .pro: Solo habrá un archivo .pro en el proyecto. Este archivo contiene toda la información necesaria para realizar la build de la aplicación.
- .h: Estos archivos incluyen la declaración de variables y las cabeceras de las funciones de la clase correspondiente, tanto pública como privada.
- .cpp: Son los archivos en los cuales se sitúa el código fuente de la clase.
- .ui: Este archivo es el correspondiente a la interfaz gráfica.

Para el desarrollo de la interfaz gráfica se puede escribir en C++ utilizando el módulo Widget, además de esto, Qt tiene una herramienta gráfica llamada Qt Designer que es un generador de código basado en Widgets. En la figura 21 se puede observar Qt Designer con widgets colocados en una aplicación.

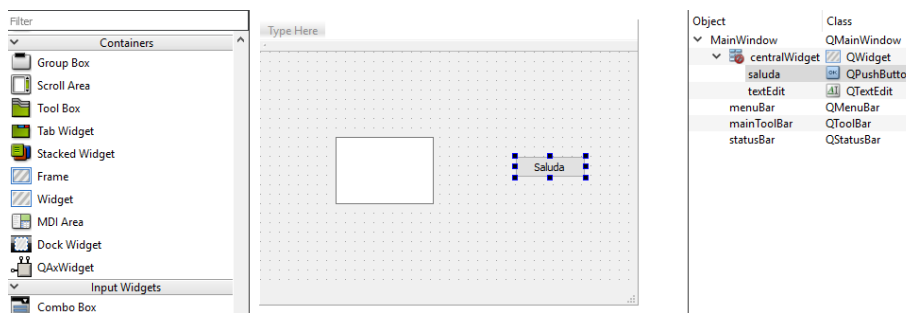


Figura 21. Qt Designer.

La integración de estas herramientas (OpenGL y Qt) se realiza de forma muy sencilla puesto que Qt tiene la API de OpenGL y por lo tanto solo hace falta decirle al IDE que estás utilizando que vas a utilizar esas librerías incluyendo lo siguiente "#include <QOpenGLWidget>". Además, en caso de que se esté utilizando como IDE Qt Creator se debe ir al fichero .ui y en el widget en el que se muestra la simulación, promoverle a la clase que tenga el código de la simulación. En caso de utilizar librerías

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

externas se debe incluir en el fichero .pro la palabra reservada LIBS seguido del path donde se encuentre la librería: LIBS += pathDeMiLibreria y en la clase en la que se utiliza realizar el include correspondiente.

Las funciones básicas y de mayor importancia que nos proporciona OpenGL con Qt son las siguientes:

- **initializeGL():** Esta función se ejecuta una sola vez y antes que las otras dos funciones. Por lo tanto, se utiliza para inicializar y configurar todo lo necesario para la utilización de OpenGL u otros.
- **paintGL():** Esta función es la que renderiza la escena de OpenGL y se llama siempre que el widget necesite ser actualizado. Además, este método será en el cual se modificará la posición de la cámara y la posición del robot, es decir, las matrices view y model. Gracias a los cambios en esta última matriz se realiza la animación de movimiento del robot.
- **resizeGL(int w, int h):** En este método se debe configurar el viewport (los parámetros de entrada w y h son el ancho y el alto respectivamente de la zona donde se podrá visualizar el código desarrollado en OpenGL), el tipo de vista (perspectiva u ortográfica), etc. Es llamado por primera vez cuando el widget se crea (siempre después de initializeGL) y siempre que el widget sea reescalado. Este método es el responsable de crear la matriz projection.

En cuanto a las funciones relacionadas con la GUI se trata de las funciones típicas que podrías encontrarte en otros software que incluyen interacción con el usuario, por ejemplo la función que se ejecuta al pulsar un botón es `onNombreDeBoton clicked()`. Además cada tipo de widget tiene métodos específicos, por ejemplo un campo de texto tiene el método `value()` que devuelve el texto introducido en ese campo, otro ejemplo se trata del widget checkbox el cual tiene el método `isChecked()` que devuelve si el checkbox está con un tick. ~~Para poder implementar el cauce gráfico existen diferentes APIs que se pueden utilizar como pueden ser Direct3D (para Windows), Mantle (para tarjetas AMD), Vulkan (basado en Mantle y multiplataforma) o OpenGL. Se decidió utilizar OpenGL puesto que es la librería más conocida, pública y multiplataforma, además es fácil integrarlo en la mayoría de los proyectos.~~

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

OpenGL es una API multilenguaje y multiplataforma que se utiliza para el desarrollo de aplicaciones en las que se utilicen gráficos 2D y 3D. Esto te proporciona funciones con las cuales podrás realizar imágenes, animaciones, juegos, simulaciones, etc. OpenGL te permite realizar entre otras muchas cosas:

- Construir formas geométricas a partir de las primitivas que este te proporciona.
- Ubicar los objetos en la escena.
- Ubicar el punto desde el que se visualiza la escena.
- Poner color o texturas.
- Crear luces.
- Realizar la rasterización.

Las funciones básicas y de mayor importancia que nos proporciona OpenGL con Qt son las siguientes:

initializeGL(): Esta función se ejecuta una sola vez y antes que las otras dos funciones. Por lo tanto, se utiliza para inicializar y configurar todo lo necesario para la utilización de OpenGL u otros.

paintGL(): Esta función es la que renderiza la escena de OpenGL y se llama siempre que el widget necesite ser actualizado. Además, este método será en el cual se modificará la posición de la cámara y la posición del robot, es decir, las matrices view y model. Gracias a los cambios en esta última matriz se realiza la animación de movimiento del robot.

• **resizeGL(int w, int h):** En este método se debe configurar el viewport (los parámetros de entrada w y h son el ancho y el alto respectivamente de la zona donde se podrá visualizar el código desarrollado en OpenGL), el tipo de vista (perspectiva u ortográfica), etc. Es llamado por primera vez cuando el widget se crea (siempre después de initializeGL) y siempre que el widget sea reescalado. Este método es el responsable de crear la matriz projection.

OpenGL tiene diferentes versiones que siguen pudiendo ser utilizadas hoy en día, a continuación, realizaré una muy breve explicación de las más relevantes:

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Sin viñetas ni numeración

Con formato: Letra normal TFG

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Sin viñetas ni numeración

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

• OpenGL 1.X: en las diferentes actualizaciones fueron haciendo extensiones al núcleo de la API.

• OpenGL 2.X: se incorporó GLSL (OpenGL Shading Language), con el cual se podía programar las etapas de transformación y rasterizado del cauce gráfico.

• OpenGL 3.X: en la primera etapa (OpenGL 3.0) se nombra ciertas funciones como obsoletas, que serán marcadas para ser eliminadas en futuras versiones (la mayoría de ellas en la versión 3.1).

• OpenGL 4.X: actualmente la última versión de OpenGL (4.6) lanzada este mismo año 2017. Se añaden una gran cantidad de funcionalidades.

A pesar de esto OpenGL tiene un problema, no es fácil crear una interfaz con la que un usuario pueda interactuar, por ello y para solucionar este problema se utiliza Qt en este proyecto.

La integración de estas herramientas (OpenGL y Qt) se realiza de forma muy sencilla puesto que Qt tiene la API de OpenGL y por lo tanto solo hace falta decirle al IDE que estás utilizando que vas a utilizar esas librerías incluyendo lo siguiente "#include <QOpenGLWidget>". Además en caso de que se esté utilizando como IDE Qt Creator se debe ir al fichero .ui y en el widget en el que se muestra la simulación, promoverle a la clase que tenga el código de la simulación. En caso de utilizar librerías externas se debe incluir en el fichero .pro la palabra reservada LIBS seguido del path donde se encuentre la librería: LIBS += pathDeMiLibreria y en la clase en la que se utiliza realizar el include correspondiente.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Negrita

Con formato: Sin viñetas ni numeración

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Sangría: Izquierda: 0 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

3. GLM



A pesar del control que ofrece OpenGL sobre el cauce gráfico, en esta aplicación se ha necesitado llegar más en detalle sobre la matriz MVP, siendo exactos en la matriz Model (M), puesto que se necesita tener en todo momento las coordenadas de los sensores del robot para realizar los cálculos sobre las colisiones de los mismos con el circuito.

Se ha optado por la utilización de GLM como librería para el cálculo de la matriz MVP puesto que es la librería recomendada por OpenGL y por lo tanto es la más apta para este funcionamiento.

En este proyecto GLM se utiliza, como se ha dicho anteriormente, para el cálculo de la matriz MVP. Esta matriz se incluirá en la pila proporcionada por OpenGL y este realiza los cálculos pertinentes. Gracias a esto, y puesto que tenemos control total sobre la matriz Model, se puede realizar los cálculos para obtener las coordenadas de los sensores.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Letra normal TFG, Sin viñetas ni numeración

Con formato: Justificado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

4. Metodología ágil

4. Metodología ágil

Durante el desarrollo de la aplicación se han utilizado herramientas utilizadas habitualmente en proyectos en los que se aplica metodología ágil. Estas herramientas han sido:

- **Trello** como tablero de tareas, este se divide en las columnas habituales (Product backlog, To Do, Doing, Done). A su vez cada tarea tiene asignada una dificultad representada mediante colores. Las tareas no tienen asignadas personas puesto que hay una sola persona encargada de este tablero. A pesar de no ser relevante para la organización de un equipo y la división de tareas, puesto que solo se trata de una persona, ha resultado muy útil para no perder la visión de proyecto. Todo esto se puede ver en la figura 1322.

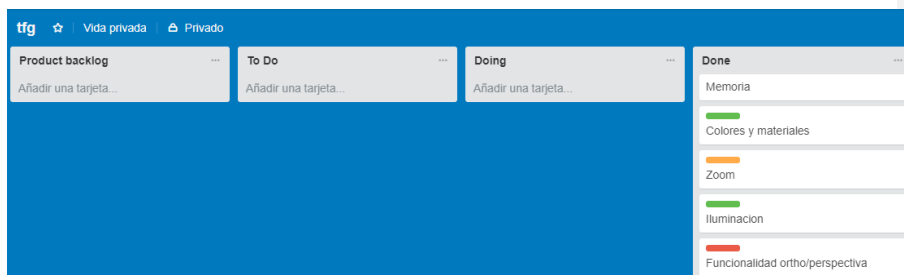


Figura 1322: Trello del proyecto Simulación de un robot sigue líneas.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Letra normal TFG, Sin viñetas ni numeración

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

- **Git** como repositorio y control de versiones (utilizado concretamente Github). Sobre este repositorio se ha ido subiendo los diferentes los diferentes incrementos de funcionalidad de la aplicación de forma periódica y que gracias a él se ha podido realizar un control de versiones. Se puede observar el repositorio desde la aplicación de windowsWindows en la figura 1423.

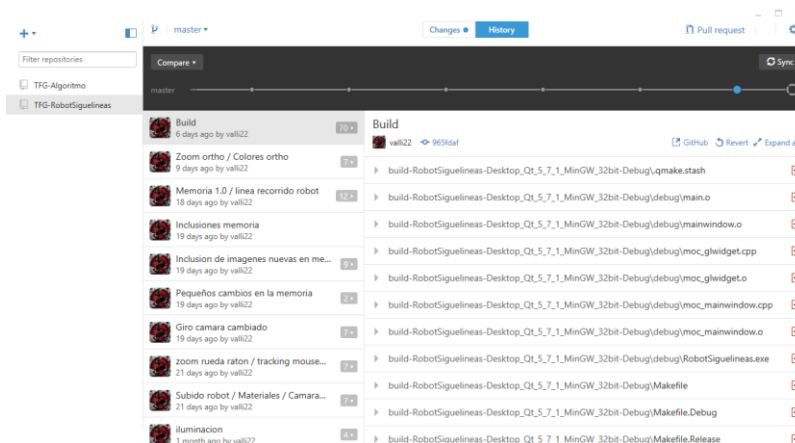


Figura 1423: Repositorio en Github desde la aplicación de escritorio de Windows.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Capítulo 4 Descripción de la aplicación

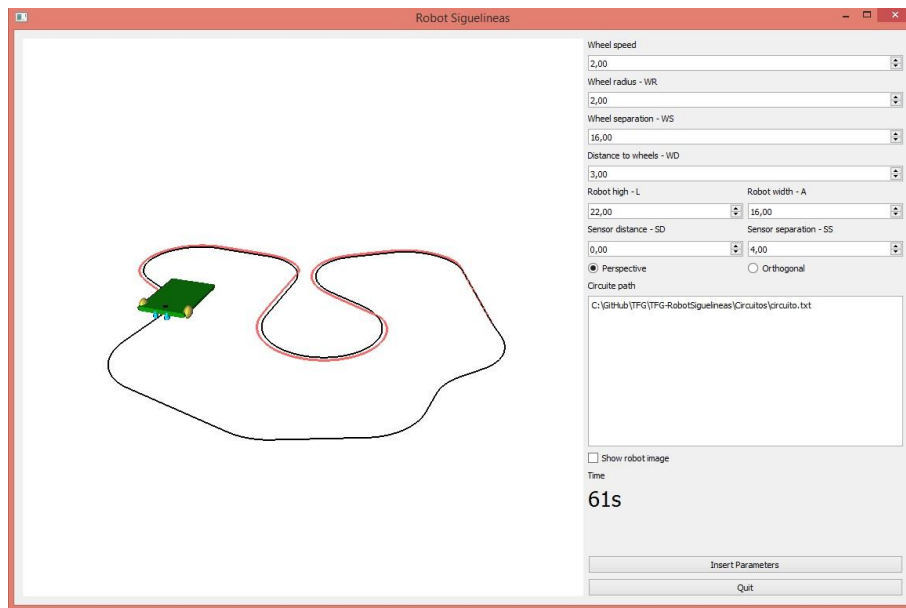


Figura 24. Vista completa de la aplicación.

Como se puede observar en la figura 24, la aplicación se puede dividir en dos zonas: izquierda y derecha. En la zona de la izquierda se muestra la simulación del robot en 3D, mientras que en la zona de la derecha se sitúan los controles de usuario.

1. Zona izquierda: Viewport

En la parte izquierda de la aplicación se puede observar una pantalla en blanco, el viewport, sobre esta se muestra la simulación del robot una vez ingresados los datos del mismo de este.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Letra normal TFG

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva



En esta zona se puede observar la simulación de la aplicación, en función de lo escogido en la zona derecha, tanto en perspectiva, ver figura 25, como en ortográfica, ver figura 26. Al iniciarse se dibuja el circuito y se hace un cálculo para posicionar automáticamente la cámara sobre el mismo. Sabiendo que la cámara se sitúa mirando hacia $y=0$ y la parte de arriba de la cámara está situada hacia $-z$, como se muestra en la figura 15, se buscan los puntos del circuito más externos, es decir, el punto situado más a la izquierda (x menor), más a la derecha (x mayor), más arriba (z mayor) y más abajo (z menor).

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

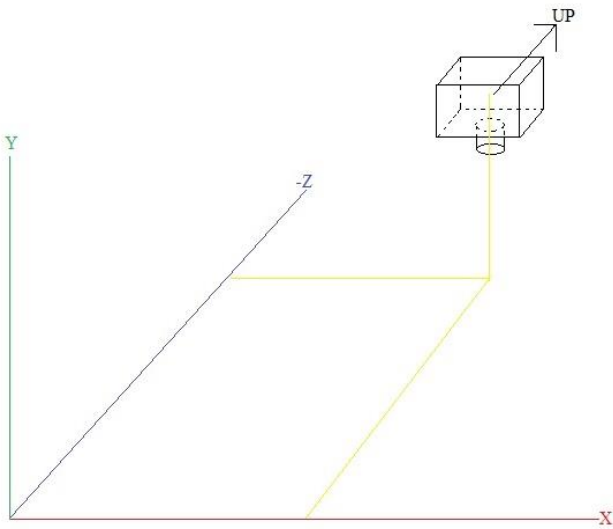
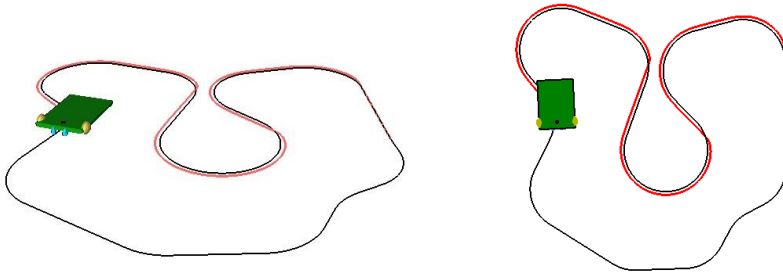


Figura 25. Vista en perspectiva.

Figura 26. Vista en ortográfica.

Figura 27. Posición y orientación de la cámara.

Una vez encontrados se crea un vector de 3 posiciones en el cual guardaremos la posición donde más adelante situaremos la cámara. En este vector se guarda en la primera posición (x) el punto intermedio entre el punto situado más a la izquierda y el punto situado más a la derecha, a la hora de guardar la posición 3 (z) se realiza exactamente lo mismo, pero con la parte superior e inferior, es decir se calcula y se guarda el punto medio entre el punto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Justificado, Sangría: Izquierda: 0 cm

Con formato: Justificado, Sangría: Izquierda: 1,25 cm

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Sangría: Izquierda: 1,25 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

más superior y el punto más inferior. Para poder saber la altura a la que se debe situar la cámara para que pueda ver todo el circuito, primero se comprueba cual contiene una mayor distancia si el eje x (es decir el punto más a la derecha menos el punto más a la izquierda) o el eje z (es decir el punto más arriba menos el punto más abajo). Una vez obtenida la mayor distancia se calcula mediante trigonometría la altura necesaria, como se puede ver en la figura 16.

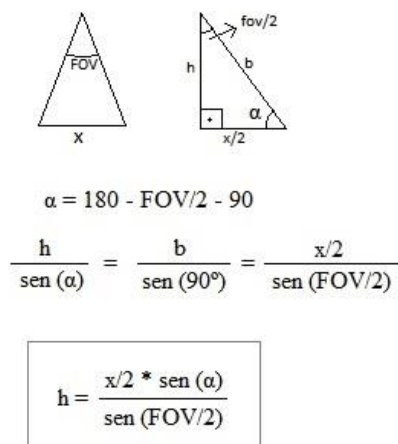


Figura 16. Trigonometría utilizada para el cálculo de la altura de la cámara.

Con esto la cámara se situará automáticamente en el centro de cualquier circuito que se introduzca y a una distancia a la cual se pueda ver el circuito completamente.

Una vez tenemos la posición de la cámara, dependiendo de que opción haya elegido el usuario se hacen los cálculos para la proyección en perspectiva u ortográfica, esto implica que se cambia también la posición de la cámara obtenida anteriormente. En caso de que se opte por una visualización en ortográfica la cámara se situará justo encima del circuito, sin embargo, en caso de que se opte por una visualización en perspectiva se colocará la cámara a 45° respecto a la posición inicial (justo encima del circuito) para que sea posible observar los modelos 3D y la perspectiva de la imagen. Como

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Justificado, Sangría: Izquierda: 1,25 cm

Con formato: Sangría: Izquierda: 1,25 cm

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

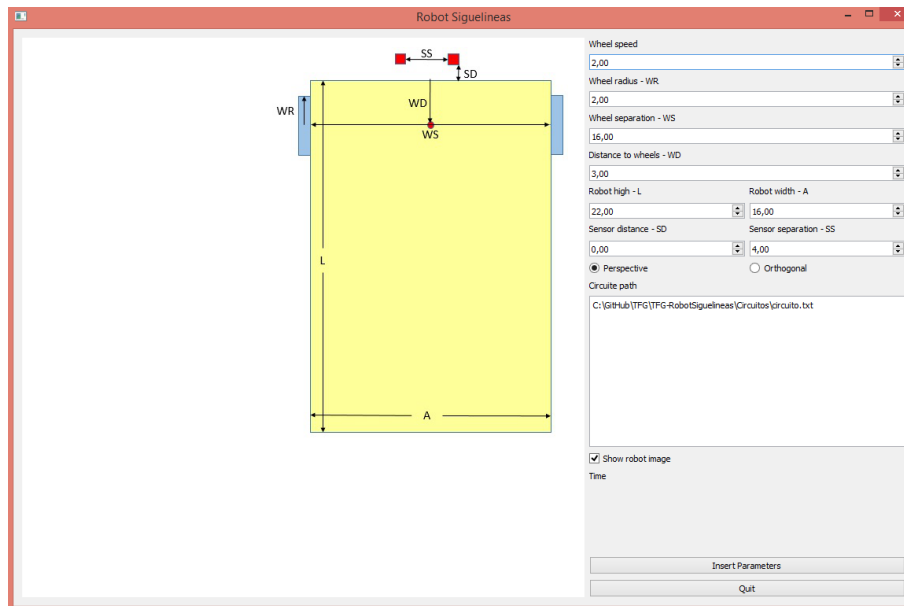
Con formato: Fuente: Century Gothic, 10 pto, Cursiva

añadido, se ha implementado la funcionalidad de zoom en ambas proyecciones el cual se puede utilizar con la rueda del ratón.

Una vez realizado esto se pasa a la En el fase en bucle de la aplicación se realizan de los cálculos de la posición y el dibujo del robot. Dependiendo de los parámetros introducidos en la parte derecha de la aplicación el robot dibujado será acorde a las mismas. Los cálculos necesarios son, el cálculo de la posición del robot, el cálculo de las coordenadas de los sensores y el algoritmo que determina si los sensores del robot están tocando alguna parte del circuito.

Además, se muestra en todo momento el recorrido que el robot ha llevado durante esa ejecución mediante una línea roja, con lo que se da un mejor feedback al usuario.

Esta zona también se aprovecha para mostrar todos los parámetros geométricos del robot sobre una imagen del mismo de este como se muestra en la figura 27. Esta imagen se puede activar y desactivar en todo momento desde la zona derecha de la aplicación. Se explica con mayor detenimiento en el siguiente apartado.



Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Negrita

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Figura 27. Aplicación con imagen de datos geométricos activa.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

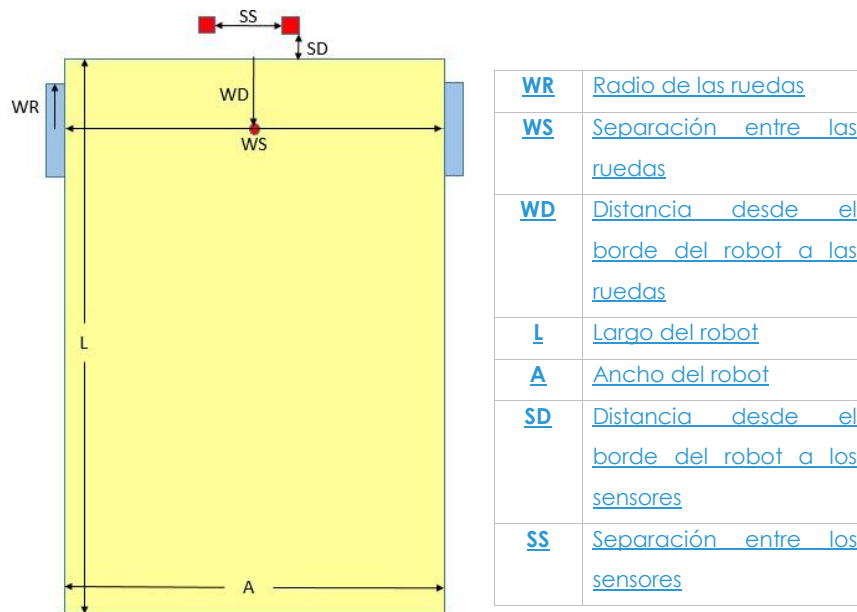
1. 2. Zona derecha: Campos de entrada de datos

Con formato: Sin viñetas ni numeración

Con formato: TFG titulo 2

La zona derecha de la aplicación se centra en la recogida de datos del usuario, ~~que a continuación se pasa a explicar y el porqué de las decisiones tomadas.~~

Para un mejor entendimiento de los widgets de interacción con el usuario, se ha añadido abreviaturas a cada uno de ellos y la posibilidad de superponer una imagen con las referencias visuales sobre el robot, más tarde se explica en más detalle. En la figura 28, se muestra esta imagen y una tabla de abreviaturas que explican el significado de cada una de ellas.



Con formato: Izquierda

Figura 28. Imagen de referencia sobre datos geométricos del robot y tabla de abreviaturas.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

El archivo que contiene toda la información sobre la interfaz y sus widgets es "MainWindow.ui". Para comenzar se decidió que la parte donde se realiza la simulación ocupara cerca del 75% de la aplicación puesto que es la parte más importante, el lugar donde más tiempo se ~~este~~estará observando y donde más detalles hay que fijarse, la comentada en el punto anterior.

Una vez entrada en la parte de interacción con el usuario se ~~coloca~~decidió que ~~estuviera~~ a la derecha por similitud a la mayor parte de aplicaciones encontradas y por lo tanto que le ~~resultara~~resulte más natural al usuario el acceso a esta. Además, todas las labels ~~estarán~~están en inglés puesto que este es el idioma más hablado y por tanto dota a la aplicación de una mayor usabilidad.

A continuación, se ~~decidió~~opta por incluir los inputs en grupos de conceptos similares, como el grupo de las ruedas, el de los sensores y el del tamaño del robot. En el caso de las ruedas se ~~decidió~~per utilizar tres inputs con sus respectivos labels en diferentes niveles, como se observa en la figura ~~29~~27, puesto que la unión de los 3 parámetros de las ruedas en un mismo nivel no resultaba fácil de visualizar.



Figura ~~17~~29. Paquete de ruedas en tres niveles.

Después de las ruedas viene un parámetro que comparten las ruedas y el robot, que es la distancia a la que se sitúan las ruedas desde el extremo superior del robot. Por lo tanto, este parámetro deb~~er~~ía estar situado entre los parámetros de las ruedas y los del robot.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Al contrario que para las ruedas, para el tamaño del robot y los parámetros de los sensores, se agrupan [en 2](#) en un mismo nivel puesto que así se asimila a simple vista que esos parámetros están relacionados, como se ve en la figura [1830](#).

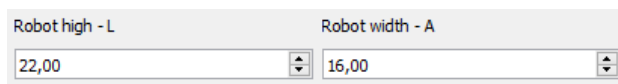


Figura [1830](#). Paquete de dimensiones del robot en un nivel

Hasta aquí están los parámetros de entrada necesarios para la simulación del robot (a excepción del circuito). Por tanto, ahora deberán entrar las opciones de la aplicación que no son directamente relevantes para la simulación de la aplicación, en primer lugar, [se puede](#) encontrar dos radio buttons que deciden si el usuario desea la vista perspectiva u ortográfica, [como se puede observar en la figura 31](#).

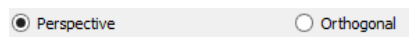


Figura 31. Radio buttons para elegir tipo de proyección.

-Esta opción está colocada aquí puesto que a pesar de que todavía queda introducir el circuito, si se pusiera detrás del input para el circuito esta opción sería poco visible y podría ignorarse, mientras que viniendo de opciones de tamaño reducido se ve claramente. [En la figura 25 se puede observar cómo se visualiza con vista en perspectiva, la vista ortográfica se puede observar en la figura 26](#).

El último input se trata de un cuadro de texto editable en el que se debe introducir el path del circuito que se desea utilizar, [ver figura 32](#).

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Centrado

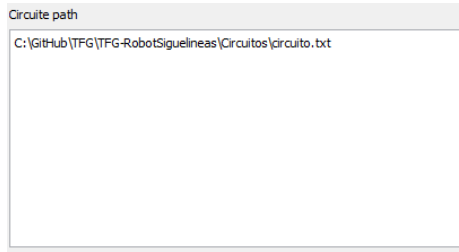
Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

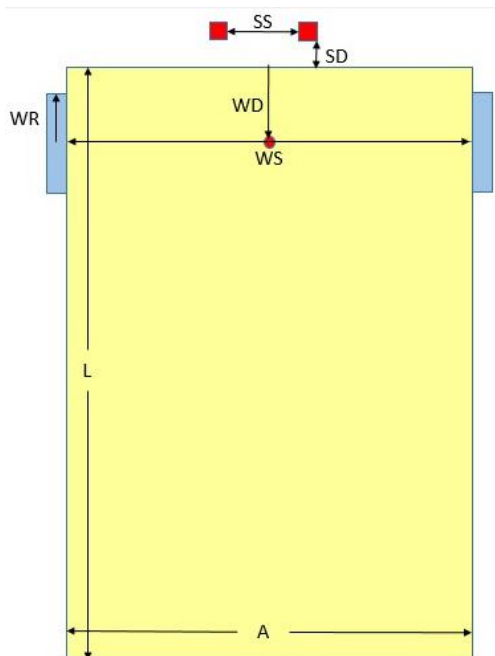
Con formato: Fuente: Century Gothic, 10 pto, Cursiva



- Con formato: Fuente: Century Gothic, 10 pto, Cursiva
- Con formato: Fuente: 10 pto, Cursiva
- Con formato: Fuente: Century Gothic, 10 pto, Cursiva
- Con formato: Fuente: Century Gothic, 10 pto, Cursiva
- Con formato: Centrado

Figura 32. Campo de texto donde se debe introducir el path del circuito.

A continuación, se sitúa un checkbox que activa y desactiva una la imagen de referencia que tiene las indicaciones y las medidas del robot para que el usuario sepa exactamente qué es lo que está modificando en todo momento. Además, en cada input de los anteriores se ha añadido la abreviatura al final para mayor entendimiento del usuario. Se puede observar la imagen que se da como referencia al usuario en la figura 1928.



- Con formato: Fuente: Century Gothic, 10 pto, Cursiva
- Con formato: Fuente: Century Gothic, 10 pto, Cursiva
- Con formato: Fuente: Century Gothic, 10 pto, Cursiva
- Con formato: Fuente: Century Gothic, 10 pto
- Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Figura 19. Imagen de referencia para el usuario para saber qué es lo que modifica en todo momento.

Por último Después, hay una zona en la que se muestran los segundos pasados desde que empezó la simulación como se muestra en la figura 3320, con un tamaño de letra que hace que sobresalga sobre el resto de la interfaz haciendo así que una vez la simulación este comenzada se vea claramente la simulación y los segundos, pudiendo ignorar el resto de la interfaz.



Figura 3320. Tiempo de simulación.

Por último, hay dos botones, el primero cuyo texto pone "Insert Parameters", el cual cuando pulses se introducen todos los parámetros a la aplicación y se inicia la simulación. Y un último botón que contiene el texto "Quit" que sirve para cerrar la aplicación. Estos dos últimos botones se pueden observar en la figura 34.

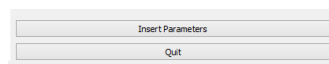


Figura 34. Botones de inicio de simulación y cerrar aplicación.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Justificado

Con formato: Centrado

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

2. Casos de uso

3. Casos de uso

A ~~continuación~~continuación, se muestran diferentes resultados de diferentes robots sobre un mismo circuito:

WR	Radio de las ruedas
WS	Separación entre las ruedas
WD	Distancia desde el borde del robot a las ruedas
L	Largo del robot
A	Ancho del robot
SD	Distancia desde el borde del robot a los sensores
SS	Separación entre los sensores

Figura 21. Leyenda de las abreviaturas usadas

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Letra normal TFG, Centrado, Sin viñetas ni numeración

Con formato: Centrado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

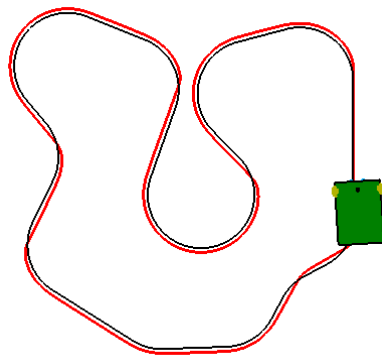
Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Parámetros	Speed	WR	WS	WD	L	A	SD	SS
Valor	2	2	16	3	22	16	0	4

Tiempo: 90s.



Este robot es un robot con parámetros geométricos medios lo que hace que se realice el circuito en un tiempo normal.

Con formato: Justificado

Parámetros	Speed	WR	WS	WD	L	A	SD	SS
Valor	2	1.75	10	7	30	15	2	7

Tiempo: 92s.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

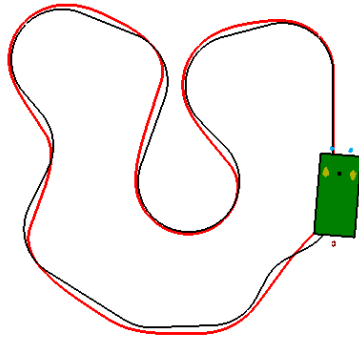
Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Simulación de un robot sigue líneas

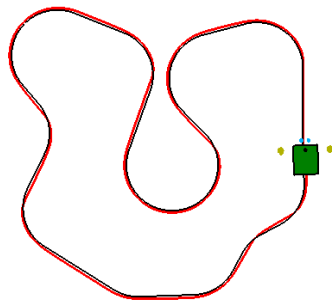


Este robot tiene un menor radio de ruedas, una mayor distancia entre sensores y ruedas y una mayor distancia entre sensores, además tiene las ruedas más juntas, todo esto hace que se realicen menos giros pero de mayor magnitud.

Parámetros	Speed	WR	WS	WD	L	A	SD	SS
Valor	2	1.5	20	2	12	10	2	3

Parámetros	Speed	WR	WS	WD	L	A	SD	SS
Valor	2	1.5	20	2	12	10	2	3

Tiempo: 125s.



Este robot tiene una separación entre ruedas muy grande pero una separación entre sensores reducida. Esto hace que se detecten muy rápido las colisiones, sin embargo, realiza giros de mayor amplitud, lo que hace que en un circuito como este con una gran cantidad de giros sea más lento.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Justificado

Con formato: Justificado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

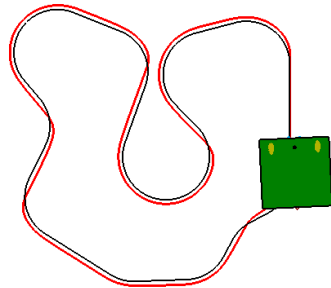
Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Justificado

Parámetros	Speed	WR	WS	WD	L	A	SD	SS
Valor	2	2.5	20	4	30	30	0	5

Tiempo: 76s.



Este robot posee la misma distancia entre sensores y ruedas que el anterior, misma distancia entre ruedas sin embargo los sensores entre si tienen más distancia y tiene un mayor radio de rueda lo que hace que en este circuito sea más rápido.

Con formato: Justificado

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Conclusiones

Gracias a este proyecto he obtenido bastante conocimiento sobre C++, además de sobre una tecnología relativamente reciente como es Qt. Para poder realizarlo además ha sido necesario indagar en el funcionamiento más profundo de OpenGL, dotándome así de un mayor conocimiento sobre cómo funciona a bajo nivel.

De los objetivos principales propuestos a la hora de realizar la aplicación, han sido todos logrados.

Una de las posibles mejoras a realizar sobre este proyecto podría ser el de incluir una forma de realizar carreras varios robots a la vez, o realizar estas carreras de manera online, una persona realizando de host y el resto poniendo sus robots y viendo la carrera. Además se podría mejorar la forma y figuras del robot y el circuito.

Otra posible mejora sería el de desarrollar un algoritmo que dado un circuito encontrara los mejores parámetros posibles del robot para realizar el robot en el menor tiempo posible.

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Título TFG

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Bibliografía

Simulación de un robot sigue líneas Grado en Ingeniería Informática

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto, Cursiva

Con formato: Fuente: Century Gothic, 10 pto

Con formato: Fuente: Century Gothic, 10 pto, Cursiva