El entorno de simulación Webots

Robótica - Grado en Ingeniería de Computadores

Departamento de Sistemas Informáticos

E.T.S.I. de Sistemas Informáticos - Universidad Politécnica de Madrid

22 de octubre de 2023

Introducción

Importancia de la simulación en robótica

Permite la prueba de algoritmos **antes** de implementarlos en hardware real

- Menor tiempo y coste de desarrollo
- **Entorno seguro para experimentar y aprender**

Facilita la **reproducción** y el análisis de situaciones específicas

- Es prácticamente imposible reproducir escenarios en el mundo real

Campos de aplicación:

- **Industria**: Desarrollo y prueba de algoritmos (p.ej. vehículos autónomos)
- **Academia**: Enseñanza e investigación, p.ej. en robótica o inteligencia artificial
- **Competiciones**: Preparación y entrenamiento en competiciones, p.ej. RoboCup

¿Qué es Webots[1]?

Plataforma *Open Software* (Apache License 2.0) para la simulación de robótica:

- Creación y uso de robots y entornos en un espacio tridimensional (o mundo)
- Biblioteca extensa de modelos predefinidos de robots, sensores y actuadores
- Simulación precisa de físicas y renderizado realistas

Entorno completo de modelado, programación y simulación para prototipado:

- Soporte para múltiples lenguajes de programación incluyendo C, C++ y Python
- Desarrollo de controladores de robot utilizando una API intuitiva
- Posibilidad de importar y exportar código para y desde otras plataformas
- Integración con robots y hardware real para pruebas en el mundo real

1 Sitio web oficial: https://www.cyberbotics.com/.
2 Concretamente desde diciembre de 2018, desde la publicación de la versión R2019a

Algunos términos comunes

Mundo: Fichero que contiene las descripciones de los robots y su entorno

Controlador: Programa con el código que controla cualquier robot del mundo

Controlador supervisor: Aquel que permite funciones de administración

Nodo: Cada objeto existente en el escenario o mundo

Campo, propiedad o característica: campo variable en el nodo

Instalación del simulador

En Windows, basta con descargar el instalador de la última versión y lanzarlo.

En macOS, dos opciones:

1. Descargar el fichero de instalación `.dmg` de la aplicación e instalar
2. Instalar a través de *homebrew*

En GNU/Linux, varias opciones:

1. **Añadiendo el repositorio como fuente adicional del APT (recomendado).**
2. Desde un *tarball* (`.tar.bz2`) o un paquete `.deb` en el caso de Debian
3. Instalando el paquete disponible en `snap`
4. Contenedores docker (generalmente para simulaciones *headless*)

Existen también contenedores de docker y servidor para, por ejemplo, el lanzamiento de simulaciones *headless*.

Un vistazo a la interfaz

/usr/local/webots/projects/samples/robotbenchmark/maze_runner/worlds/maze_runner.wbt (maze_runner) - Webots R2022b

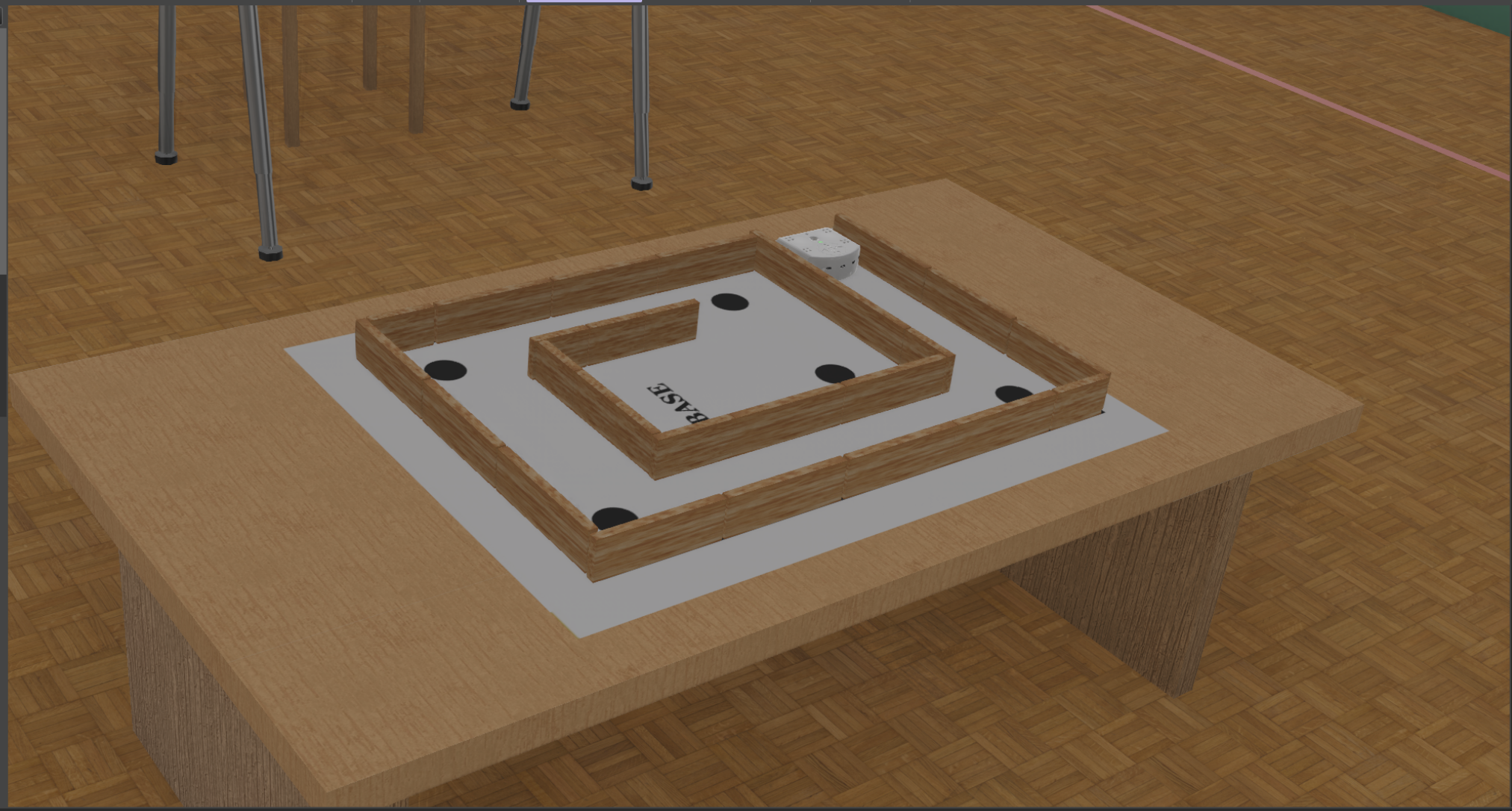
le Edit View Simulation Build Overlays Tools Help

aze Runner

0:00:00:000 - 0.00x

IMPORTABLE EXTERNPROTO

- WorldInfo
- Viewpoint
- TexturedBackground
- CeilingLight "ceiling light(1)"
- SpotLight
- Table "table"
- DEF MAZE_WALLS Group
- DEF THYMIO2 Thymio2
- MazeRunnerBenchmark "robot"
- Floor "floor"
- Table "table(2)"
- WoodenChair "wooden chair"
- WoodenChair "wooden chair(1)"
- WoodenChair "wooden chair(2)"
- WoodenChair "wooden chair(3)"
- WoodenChair "wooden chair(4)"
- WoodenChair "wooden chair(5)"
- WoodenChair "wooden chair(6)"
- WoodenChair "wooden chair(7)"
- Door "door"
- DEF CEILING Solid
- DEF DOOR_WALL Solid
- DEF BACK_WALL Solid
- DEF SHORT_WINDOW_WALL Solid
- DEF SHORT_WINDOW_WALL Solid
- DEF LONG_WINDOW_WALL Solid
- DEF LONG_WINDOW_WALL Solid



...rs/move_in_maze/move_in_maze.py

move_in_maze.py

```
"""Naive maze runner controller."""

from controller import Robot

# Get reference to the robot.
robot = Robot()

# Get simulation step length.
timeStep = int(robot.getBasicTimeSt

# Constants of the Thymio II motors
maxMotorVelocity = 6

# Get left and right wheel motors.
leftMotor = robot.getDevice("motor.
rightMotor = robot.getDevice("motor

# Frontal distance sensors that can
outerLeftSensor = robot.getDevice("
centralLeftSensor = robot.getDevice
centralSensor = robot.getDevice("pr
centralRightSensor = robot.getDevic
outerRightSensor = robot.getDevice(

# Enable sensors.
outerLeftSensor.enable(timeStep)
centralLeftSensor.enable(timeStep)
centralSensor.enable(timeStep)
centralRightSensor.enable(timeStep)
outerRightSensor.enable(timeStep)

# Get and enable ground sensors to
# groundLeftSensor = robot.getDevic
# groundRightSensor = robot.getDevi
# groundLeftSensor.enable(timeStep)
# groundRightSensor.enable(timeStep

# Disable motor PID control mode.
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf')

# Set ideal motor velocity.
velocity = 0.7 * maxMotorVelocity

isRotating = False
while robot.step(timeStep) != -1:
    # Read values from four distanc
    if not isRotating and centralSe
        # Black circle detected.
        isRotating = True
    elif isRotating and outerLeftSe
        isRotating = False

    leftMotor.setVelocity(velocity)
    if isRotating:
        rightMotor.setVelocity(-vel
    else:
        rightMotor.setVelocity(velo

```

nsole - All

```
O: move_in_maze: Starting controller: python3 -u move_in_maze.py
O: supervisor: Starting controller: python3 -u supervisor.py
```

Interfaz de Webots

Barra de menú: Accesos a todos los aspectos de la aplicación.

Barra de herramientas principal: Trabajo sobre la simulación

Árbol de escena: Información jerárquica acerca del mundo, objetos y robots.

- `Worldinfo`: Parámetros como el paso de simulación y la gravedad.
- `Viewpoint`: Parámetros relacionados con la perspectiva de visualización.
- **Editor de dominio**: Modificación de caracteristicas del nodo seleccionado.

Pantalla de simulación: Ventana para la visualización de la simulación.

Editor de texto: Sirve para editar los controladores de los robots.

Consola: Salida estándar para los controladores que estén funcionando

Jerarquía de archivos en un proyecto

Estructura base de directorios

The standard file hierarchy of a project

Un proyecto es un directorio con, al menos, un directorio denominado `world/`

- Contiene ficheros de descripción de mundo (`.wbt`) y archivos del proyecto
- **Deberá incluir al menos** un fichero con extensión `.wbt`
- Puede incluir un directorio `textures\` con las texturas a utilizar

Ahora bien, normalmente son necesarios más directorios; estos son:

- `controllers/`: Fuentes para el control de robots.
- `libraries/`: Posibles bibliotecas externas en el proyecto.
- `plugins/`: Plugins para alterar el comportamiento típico de la simulación
- `protos/`: Prototipos disponibles para todos los ficheros del proyecto.

Ficheros asociados a un mundo

Cada mundo (e.g. `world.wbt`) lleva asociados los siguientes ficheros ocultos:

- `.world.wbproj`: Información sobre la UI del usuario (e.g. perspectiva).
- `.world.jpg`: Imagen de carga de 768x432 en simulaciones o animaciones.

Si no existen o se eliminan, se crean al guardar correctamente el mundo.

El directorio `controllers/`

Contiene un **directorio por cada posible controlador** de la simulación:

- El `.wbt` contiene el nombre del controlador a iniciarse para cada robot.
- Ese nombre hace referencia al **directorio del controlador**
- Es un campo independiente de plataforma y lenguaje (sólo es una cadena)

Cuando Webots intenta inicializar un controlador sigue el siguiente proceso:

1. Busca en `controllers/` un directorio que coincida con el nombre indicado
2. Busca en el subdirectorio un fichero que coincida con el nombre indicado
3. Si hay varios, selecciona uno de ellos siguiendo el siguiente orden:

   ```
   [.exe] > .class > .jar > .bsg > .py > .m
   ```

4. Si no encuentra ninguno, lanzará un error y iniciará un controlador vacío

Prácticas de webots

Flujo de trabajo

A la hora de editar el mundo de webots, se deben seguir los siguientes pasos:

1. Reiniciar la simulación
2. Editar el mundo
3. Guardar los cambios
4. Lanzar la simulación

¡Cuidado con reinitar la simulación sin guardar los cambios! Se perderán

Tutoriales de Webots

¿Para qué repetir lo que ya está bien explicado?. Hagámoslo práctico:

1. Tutorial 1: Introducción y primera simulación
2. Tutorial 2: Modificando el entorno
3. Tutorial 3: Cambios de apariencia
4. Tutorial 4: Sobre los controladores de un robot
5. Tutorial 5: Sólidos y físicas
6. Tutorial 6: Creando un robot de cuatro ruedas
7. Tutorial 7: Creación de PROTO para la reutilización

El tutorial es interesante pero no lo necesitamos. El 9 no sirve para nada

¡GRACIAS!