Aprendizaje por Refuerzo en Robótica
Departamento de Ingeniería Informática
Instituto Tecnológico de Buenos Aires

Trabajo Práctico 1

Objetivo

Familiarizarse con el entorno de simulación de Webots mediante la construcción de modelos simples de ambiente y robot.

1. Introducción

El objetivo es simular un robot muy simple con forma de cilindro, compuesto de 2 ruedas y 2 sensores infrarrojos. El robot debe moverse en un ambiente sencillo con 3 obstáculos (un cubo, un cilindro y una pequeña pared) y delimitado por 4 paredes.

1. Para empezar, crear un proyecto llamado "arrtp1" en un directorio local seleccionando "New Project Directory..." en el menú "Wizard".

El proyecto cuenta con 2 directorios: worlds y controllers (además de plugins y protos). En el primero se guarda la representación de los "mundos virtuales" (escenas 3D) y en el segundo, los programas para controlar al robot simulado.

2. Piso y luces

1. Crear un nuevo "mundo" (File: New World) y guardarlo como "arrtp1.wbt" (File: Save World As...).

NOTA: Es conveniente guardar el "mundo" con frecuencia a medida que se edita, por si falla la conexión con el servidor de licencias u ocurre algún error con la simulación.

2. Abrir el árbol de la escena (Tools: Scene Tree...).

NOTA: Mientras se edita la escena 3D, se puede detener la simulación con el botón de "Stop".

3. Seleccionar el último nodo Solid, que representa el piso. Navegar por sus nodos descendientes hasta children/Shape/geometry/Color/color y editar los colores para que sea un grillado de dos grises claros.

4. Seleccionar el nodo PointLight y editar sus parámetros con:
   ambientIntensity: 0.4
   intensity: 0.6
   location: 0.4 0.5 0.4

NOTA: Para ver la posición de las luces, activar la visibilidad de los nodos PointLight, activando "Display lights" en las preferencias (Tools: Preferences...). (Webots v6.1.1)

5. Copiar el nodo PointLight (usar los botones en la parte superior del árbol) para crear 3 luces más con los siguientes parámetros:
   ambientIntensity: 0.3
   intensity: 0.4
   location: [0.4 0.5 -0.3] [-0.3 0.5 -0.3] [-0.3 0.5 0.4] (respectivamente)

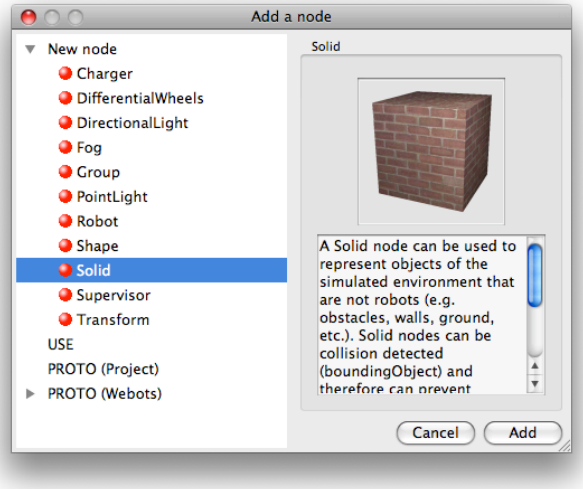
   La escena debería quedar mejor iluminada ahora.

NOTA: Para familiarizarse con el manejo del mouse en la navegación de la escena, consultar la ayuda (Help: How do I navigate in 3D?).

3. Pared

1. Seleccionar el último nodo Solid, que representa el piso, en el árbol de la escena.

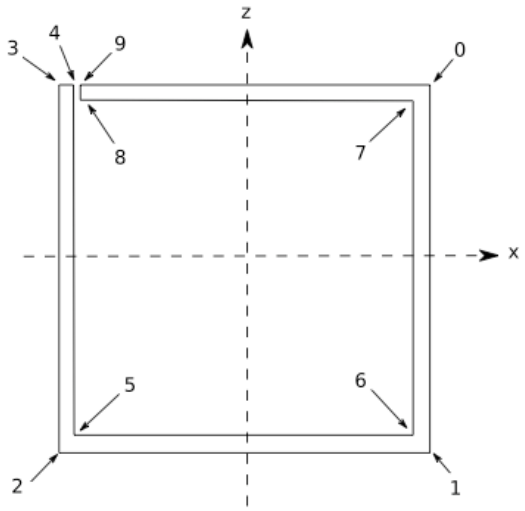
2. Usar el botón de "insert after" para crear un nuevo nodo hermano de tipo Solid. Nombrarlo como "wall" (editando su atributo "name").



3. Seleccionar su atributo "children" y usar "insert after" para insertar un nuevo nodo Shape.

4. En el atributo appearance de este nuevo Shape, crear un nuevo nodo de Appearance usando el botón de "new node". Idem para un nodo Material en el atributo material. Seleccione un color marrón claro como diffuseColor y uno más claro para specularColor en el nodo Material.

5. Crear un nodo Extrusion en el atributo geometry del Shape y editar los siguientes atributos:
   convex: FALSE
   spine: rango de Y de 0 a 0.1 (representa la altura de la pared)
   crossSection: ingresar los puntos del 0 al 10 de la siguiente matriz, repitiendo el punto 0 al final para cerrar la última cara: [0.5 0.5; 0.5 -0.5; -0.5 -0.5; -0.5 0.5; -0.49 0.5; -0.49 -0.49; 0.49 -0.49; 0.49 0.49; -0.4899 0.49; -0.4899 0.5]



6. Para prevenir que el robot traspase las paredes, es decir, para que sea posible detectar las colisiones contra la pared, es necesario definir el boundingObject de la pared. Los "bounding objects" sólo pueden ser formas simples por razones de eficiencia, tales como cajas, cilindros, esferas y primitivas de "indexed faceset". En este caso, conviene definir un grupo de cajas que se ajusten a las 4 paredes.

   Para ello, crear un nodo Group como boundingObject de la pared e insertar un Transform en su atributo children. Agregar un Shape como único hijo del Transform y crear un nodo Material en su Appearance con diffuseColor y specularColor blanco (esto es importante para la detección de colisiones, ya que se basa en el color).

   Añadir un nodo Box en su geometry con dimensiones [0.01 0.1 1] de manera que se ajusten al tamaño de una pared. Ajustar también la translation del nodo Transform a [0.495 0.05 0] para que se corresponda con la posición de la primera pared (deberían verse aristas blancas envolviéndola).

7. Cerrar el nodo Transform y copiarlo y pegarlo en la lista de hijos del boundingObject. Es conveniente reutilizar el Shape creado para la primera pared. Para ello, volver al nodo Shape original, y editar el campo DEF como "WALL_SHAPE" y luego, en el segundo, borrar el Shape para insertar un nuevo nodo seleccionando USE WALL_SHAPE. Editar su translation a [-0.495 0.05 0] para que se corresponda con la pared opuesta a la primera.

8. Repetir el último paso para crear la tercera y cuarta pared. En este caso, además de la traslation en X y Z, también se debe editar la rotation a [0 1 0 1.57] para cambiar la orientación perpendicularmente.

Al terminar, deberían verse aristas blancas en las 4 paredes al seleccionar cualquier nodo descendiente de este último Solid.

4. Obstáculos

1. Insertar un nuevo nodo Solid al final del árbol de escena. Nombrarlo como "green box".

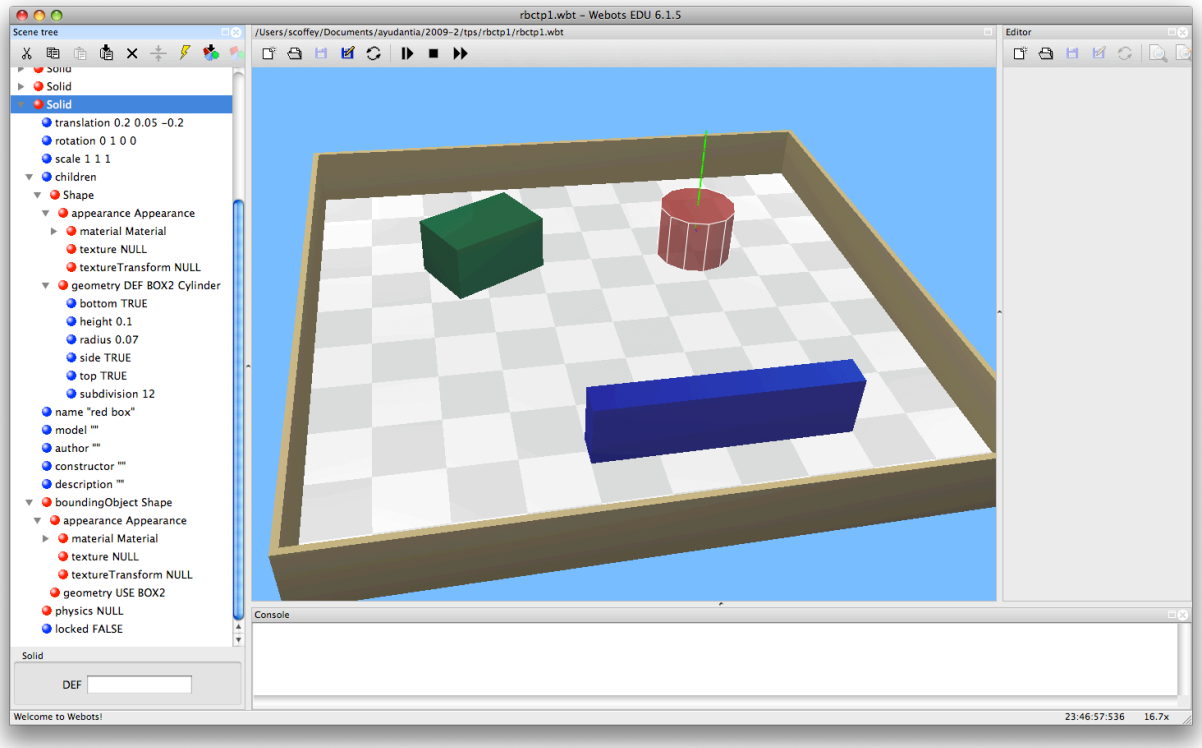
2. Como se hizo con la pared, añadir un nodo Shape como children, con Appearance y Material. Configurar su color como verde. (Siempre el specularColor más claro para la iluminación.)

3. Para el atributo de geometry del Shape, elegir un nodo Box con size [0.18 0.1 0.12] Editar el DEF del geometry para que sea BOX0.

4. Crear un nodo Shape para el boundingObject de este objeto, reutilizando el DEF anterior para su geometry. No olvidar poner nodos de Appearance y Material con color blanco para que funcionen las colisiones.

5. Editar la ubicación del obstáculo con traslation [-0.2 0.05 -0.2] y rotation [0 1 0 0.5].

6. Repetir los pasos anteriores para agregar otros 2 obstáculos más: una pequeña pared azul (no muy gruesa) y un cilindro rojo, ambos de alto 0.1. Reutilizar los formas definidas en geometry para los boundingObjects. Configurar traslation y rotation a criterio.



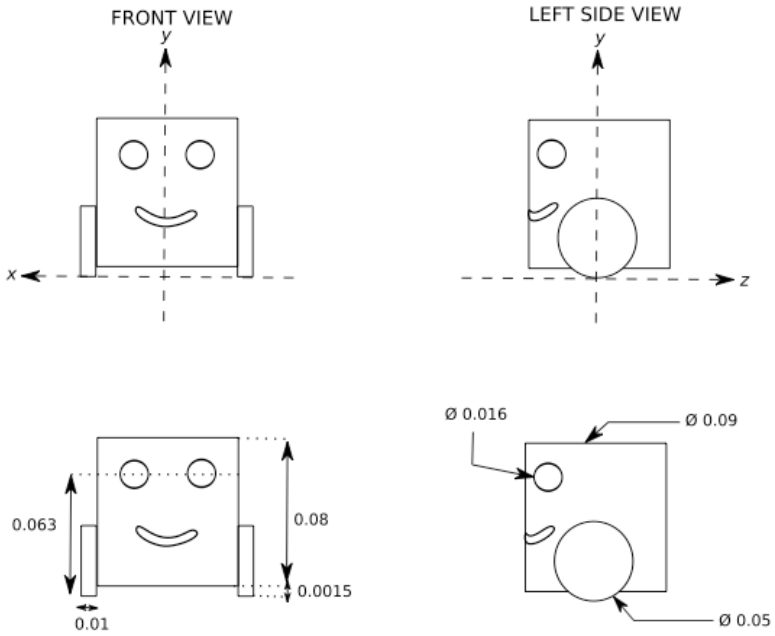
Con esto, el ambiente queda listo para la simulación.

5. Robot

Para modelar un robot simple utilizaremos un nodo DifferentialWheels con los siguiente nodos hijos: 1 nodo Transform (con Shape cilíndrico) para el cuerpo, 2 nodos Solid (con un nodo Transform con Shape cilíndrico cada uno) para las ruedas, 2 nodos DistanceSensor (con un nodo Transform con Shape cilíndrico cada uno) para los sensores infrarrojos y 1 nodo Shape para la textura de la "cara".

1. Insertar un nodo DifferentialWheels al final del árbol de escena y ponerle name: "mybot".

2. Cuerpo: Insertar un nodo Transform hijo de DifferentialWheels. Para crear un cuerpo con forma de cilindro, insertar un nodo Shape como children del Transform. En la geometry de este Shape insertar un nodo Cylinder, con height 0.08 y radius 0.045. Ingresar BODY en el DEF de la geometry para reutilizarla luego en el boundingObject. Ajustar la traslation en Y a 0.0415 en el nodo Transform para ubicar al robot por encima del piso. Darle color con un nodo Appearance y Material.



3. Ruedas

    1. Rueda izquierda: Insertar un nodo Solid hermano del Transform anterior y nombrarlo como "left wheel" (esto es importante para que luego pueda ser reconocido como rueda izquierda).

        Ahora hay que crearle una figura con forma de cilindro, aplicando una transformación que la rote alrededor del eje x. Por lo tanto, insertar primero un nodo Transform como children del Solid. Dentro de este nodo, crear un Shape con geometry: Cylinder. Establecer la apariencia como en las veces anteriores. Los parámetros del cilindro deberían ser: height 0.01 y radius 0.025.

        Ajustar la rotation del Transform de la rueda a [0 0 1 1.57] (rotación de π/2 alrededor del eje z). También ingresar WHEEL como DEF del Transform para reutilizarlo en la rueda derecha.

        Finalmente, ubicar el nodo Solid padre (no confundir con su Transform hijo) con traslation [-0.045 0.025 0] y establecer la rotation a [1 0 0 0] para que la rueda gire alrededor del eje X.

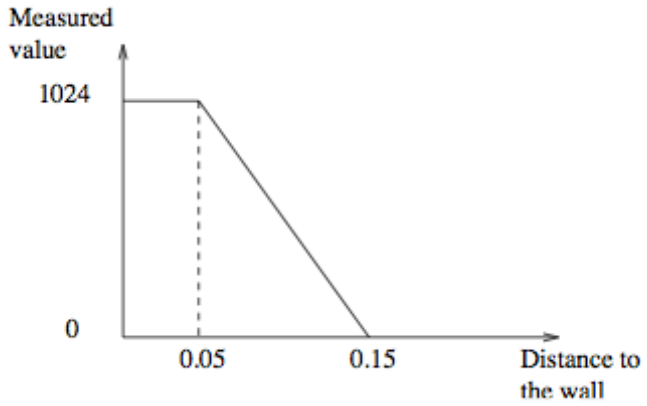
    2. Rueda derecha: Insertar un nodo Solid hermano del anterior y nombrarlo como "right wheel". Establecer su traslation a [0.045 0.025 0] y su rotation a [1 0 0 0]. En el nodo children, insertar un USE WHEEL.

4. Sensores infrarrojos

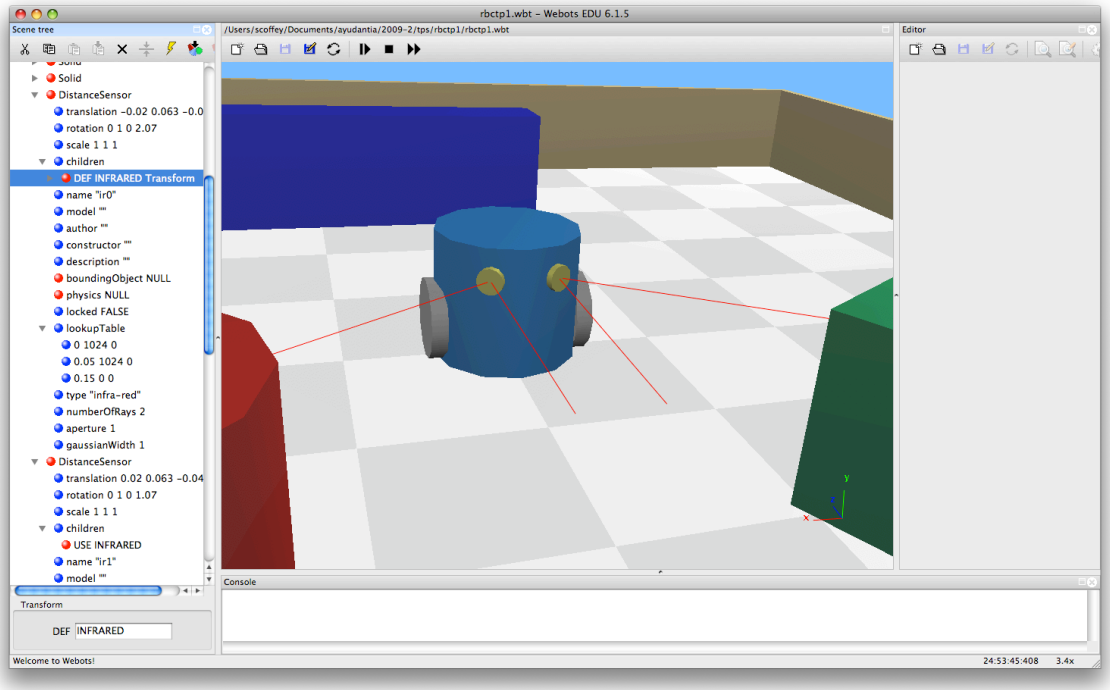
    1. Primer sensor IR: En los children del nodo DifferentialWheels, insertar un nodo DistanceSensor con nombre "ir0" (será utilizado en el controlador). Ahora hay que crearle una figura con forma de cilindro con una transformación, como en el caso de las ruedas. Insertar un nodo Transform como children del DistanceSensor. Ingresar INFRARED como su DEF, para reutilizarlo en el segundo sensor IR. Dentro del Transform, insertar un nodo Shape con geometry Cylinder, de height 0.004 y radius 0.008. Ajustar la rotation del Transform a [0 0 1 1.57].

        En el nodo DistanceSensor, ubicar el sensor y sus rayos con traslation [-0.02 0.063 -0.042]. En las Tools: Preferences: Rendering, activar Display sensor rays. Para tener un rayo dirigido hacia el frente del robot, ajustar la rotation del nodo DistanceSensor a [0 1 0 2.07].

        Por último, configurar el DistanceSensor con numberOfRay 2 y aperture 1 (para mejorar la detección de obstáculos) e ingresar esta matriz como lookupTable: [0 1024 0; 0.05 1024 0; 0.15 0 0]. Esta tabla determina una función decreciente que sirve para convertir las distancias a los valores de señal leídos de los sensores, como en el gráfico a continuación.



    2. Segundo sensor IR: Para modelar el segundo sensor infrarrojo, copiar un nodo DistanceSensor hermano y nombrarlo como "ir1". Ajustar su traslation a [0.02 0.063 -0.042] y su rotation a [0 1 0 1.07]. En sus children insertar USE INFRARED para reutilizar el Transform con Shape cilíndrica del primer sensor. La configuración de numberOfRay, aperture y lookupTable debe ser la misma.



5. Textura de la "cara"

    1. Copiar el directorio $WEBOTS_HOME/projects/samples/mybot/worlds/textures (contiene mybot.png) al directorio worlds del proyecto (donde WEBOTS_HOME corresponde al directorio donde fue instalado el Webots).

    2. Insertar un nodo Shape hijo del nodo DifferentialWheels. Darle una Appearance con un nodo ImageTexture en texture, con url: "textures/mybot.png" (relativa al directorio worlds del proyecto).

        NOTA: Las texturas sólo pueden ser mapeadas en una figura IndexedFaceSet con texCoord y texCoordIndex explícitos. Sólo se soportan imágenes JPG o PNG cuadradas, con ancho y alto en píxeles en potencias de 2 (por ejemplo: 128x128, 256x256, etc.). Consultar la documentación para más información.

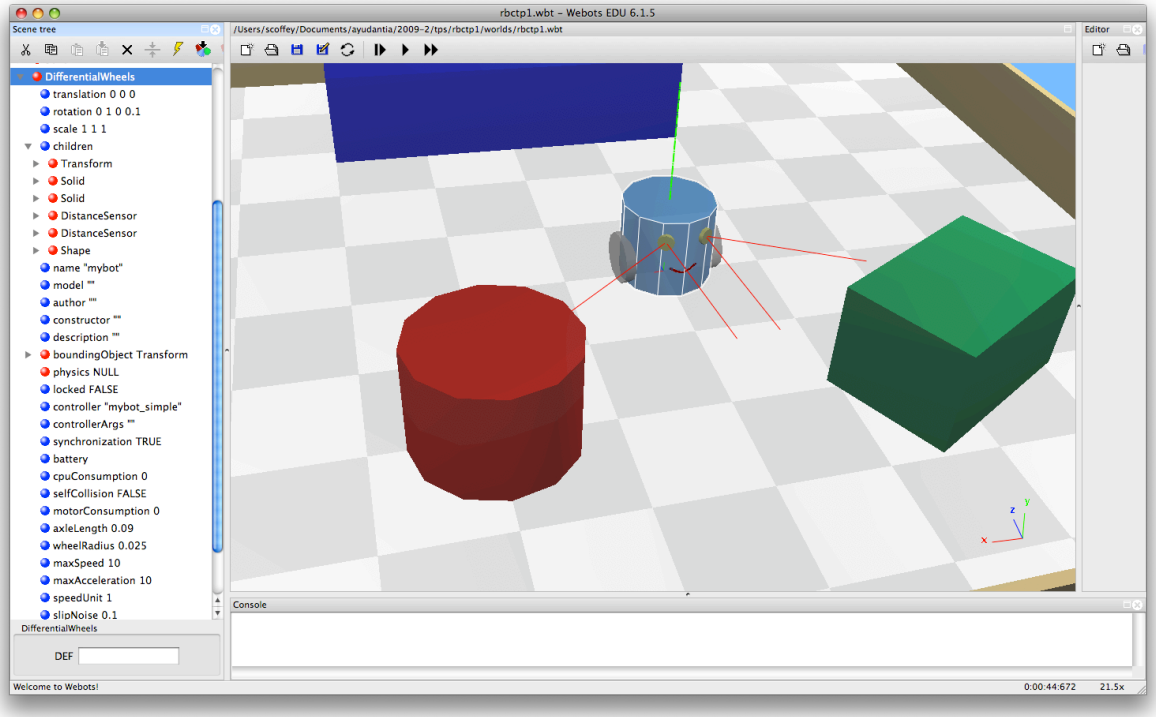
    3. Insertar un nodo IndexedFaceSet en su geometry con un nodo Coordinate en coord. Este tipo de nodo consiste en un conjunto de puntos utilizados para formar "caras" a través de un vector de índices que los agrupan, y a la vez mapear texturas en esas caras según el estándar VRML97. Ingresar la siguiente matriz de puntos en points: [0.015 0.038 -0.041; 0.015 0.023 -0.041; 0 0.023 -0.0455; -0.015 0.023 -0.041; -0.015 0.038 -0.041; 0 0.038 -0.0455]. Luego, ingresar en el atributo coordIndex del IndexedFaceSet los siguientes índices: 0, 1, 2, 5, -1, 5, 2, 3, 4, -1. (El número -1 indica el final de una cara dentro del conjunto.)

4. Insertar un nodo TextureCoordinate en texCoord e ingresar la siguiente matriz en point: [0 0; 0.5 0; 1 0; 1 1; 0.5 1; 0 1]. Ingresar en texCoordIndex: [5, 0, 1, 4, -1, 4, 1, 2, 3, -1]. Al terminar de ingresar todos los puntos y los índices, se debería ver la textura como si fuera la "boca" del robot.

5. Modificar el atributo creaseAngle del IndexedFaceSet a 0.9 para que la transición de iluminación entre las dos caras sea más suave.

6. Bounding object: Como aproximación para la detección de colisiones, se puede envolver al robot en un cilindro de las dimensiones de su cuerpo. Para ello, insertar un nodo Transform en el boundingObject del nodo DifferentialWheels y ubicarlo con traslation [0 0.0415 0]. Reutilizar el nodo BODY (definido al comienzo) para el atributo children de este nodo Transform.

7. Parámetros adicionales: Ingresar los siguientes atributos en el nodo DifferentialWheels del robot:
   axleLength: 0.09 (distancia entre ruedas)
   wheelRadius: 0.025
   speedUnit: 0.1



Adicionalmente, se pueden realizar ajustes en los parámetros de velocidad del nodo DifferentialWheels para mejorar los resultados de la simulación.

8. Controlador: Para utilizar de un programa de ejemplo, copiar el directorio $WEBOTS_HOME/projects/samples/mybot/controllers/mybot_simple al directorio controllers del proyecto (donde WEBOTS_HOME corresponde al directorio donde fue instalado el Webots). Seleccionar "mybot_simple" como atributo controller del nodo DifferentialWheels. Esto determina el controlador a utilizar para darle comportamiento al robot simulado.

   El robot está listo para la simulación.

6. Simulación

El controlador configurado para el robot es muy simple. Lee los valores de los sensores y ajusta las velocidades de las ruedas de modo que se eviten los obstáculos.

Comprobar el funcionamiento del controlador "mybot_simple" en la simulación.

```
/*
 * File:         mybot_simple.c
 * Date:         August 8th, 2006
 * Description:  A really simple controller which moves the MyBot robot
 *               and avoids the walls
 * Author:       Simon Blanchoud
 * Modifications:
 *
 * Copyright (c) 2008 Cyberbotics - www.cyberbotics.com
 */

#include <webots/robot.h>
#include <webots/differential_wheels.h>
#include <webots/distance_sensor.h>

#define SPEED 60
#define TIME_STEP 64

int main()
{
  wb_robot_init(); /* necessary to initialize webots stuff */

  /* Get and enable the distance sensors. */
  WbDeviceTag ir0 = wb_robot_get_device("ir0");
  WbDeviceTag ir1 = wb_robot_get_device("ir1");
  wb_distance_sensor_enable(ir0, TIME_STEP);
  wb_distance_sensor_enable(ir1, TIME_STEP);
```

```
  while(wb_robot_step(TIME_STEP)!=-1) {

    /* Get distance sensor values */
    double ir0_value = wb_distance_sensor_get_value(ir0);
    double ir1_value = wb_distance_sensor_get_value(ir1);

    /* Compute the motor speeds */
    double left_speed, right_speed;
    if (ir1_value > 500) {

        /*
         * If both distance sensors are detecting something, this means that
         * we are facing a wall. In this case we need to move backwards.
         */
        if (ir0_value > 500) {
            left_speed = -SPEED;
            right_speed = -SPEED / 2;
        } else {

            /*
             * We turn proportionnaly to the sensors value because the
             * closer we are from the wall, the more we need to turn.
             */
            left_speed = -ir1_value / 10;
            right_speed = (ir0_value / 10) + 5;
        }
    } else if (ir0_value > 500) {
        left_speed = (ir1_value / 10) + 5;
        right_speed = -ir0_value / 10;
    } else {

        /*
         * If nothing has been detected we can move forward at maximal speed.
         */
        left_speed = SPEED;
        right_speed = SPEED;
    }

    /* Set the motor speeds. */
    wb_differential_wheels_set_speed(left_speed, right_speed);
  }

  return 0;
}
```

Cuestionario

Sobre la construcción del ambiente:

- ¿Cuáles son los nodos mínimos que hace falta insertar para crear un obstáculo con forma de "caja" de color azul (sólido, pero omitiendo detección de colisiones)?
- ¿Para qué sirven los "bounding objects"? ¿Cuáles son sus requisitos en el árbol de escena?
- ¿Por qué la geometría de la pared que delimita el ambiente se define con puntos en 2 dimensiones y no 3?
- ¿Para qué sirve la "lookup table" de un sensor IR? (Explicar con ejemplos)
- ¿Por qué es necesario poner nodos de transformaciones (Transform) intermedios a los nodos de figuras (Shape) en el subárbol del nodo DifferentialWheels?
- ¿Para qué sirven los DEF en nodos como los de geometría?
- ¿Cómo se relacionan los atributos points y coordIndex en un nodo IndexedFaceSet?
- ¿Por qué los vectores de rotación tienen 4 componentes? ¿Cuál es la semántica de los primeros 3 componentes y del último?

Sobre el controlador y la simulación:

- Determinar qué funciones de la biblioteca de Webots están asociadas al "input" de los sensores y al "output" de los actuadores en este controlador.
- Modificar el código fuente del controlador para que imprima por salida estándar valores de debugging para los sensores y actuadores. NOTA: Cuando un programa se compila para cargarlo en el robot real, no pueden quedar llamadas a funciones que utilicen la entrada o salida estándar (stdin, stdout, stderr).
- Modificar la constante SPEED y explicar qué pasa a medida que aumenta o disminuye.
- Modificar la constante TIME_STEP y explicar qué pasa a medida que aumenta o disminuye.
- Modificar el código fuente para que el valor de umbral utilizado para los sensores infrarrojos esté definido en una constante simbólica THRESHOLD. Explicar qué pasa a medida que aumenta o disminuye este valor.
- Indicar valores de estas constantes para los cuales el robot no pueda evitar obstáculos, si es que existen.
- Explicar cómo influye la configuración de axleLength y wheelRadius en la simulación.
- Explicar en qué casos las velocidades de las ruedas tienen sentidos opuestos y por qué.
- Explicar si el robot puede o no detectar obstáculos al avanzar, girar y retroceder.