Webots 搭建麦轮小车

18329015 郝裕玮

一、实验目标

参考 PPT 内容，搭建仿真的场景（地面、光照），搭建麦轮小车，然后写一个控制器，控制麦轮小车能够八向移动（前后左右斜）和自旋。

二、实验内容与步骤

1、修改世界坐标系：将 WorldInfo 下的 coordinateSystem 从 NUE 修改为 ENU，使得坐标系成为 z 轴指向天空的右手坐标系（即重力方向为 z 轴负方向，之前的 NUE 中重力方向是 y 轴负方向）。

2、设置光源和地面：

（1）设置光源：添加新节点 TexturedBackground 和 TexturedBackgroundLight。设置完后发现仍然一片漆黑，所以在 View->Change View 下修改为 Back View，修改后成功显示环境。

（2）设置地面：添加 CircleArena(Solid)节点，并将 rotation 修改为 1 0 0 1.57，使得整个圆盘垂直于重力方向。最后可修改半径 radius 和围墙高度 wallHeight 等参数(在这里不影响最终小车运行结果)。

3、车体

（1）添加 Robot 节点

（2）添加机器人刚体：在 children 节点下添加 Shape 节点并改名为 Body（这里的改名是指修改 DEF 的值）

（3）修改外观形状：在 geometry 节点添加几何属性，选择 Box，并设置车体大小为（x=0.3 y=0.2 z=0.08）m

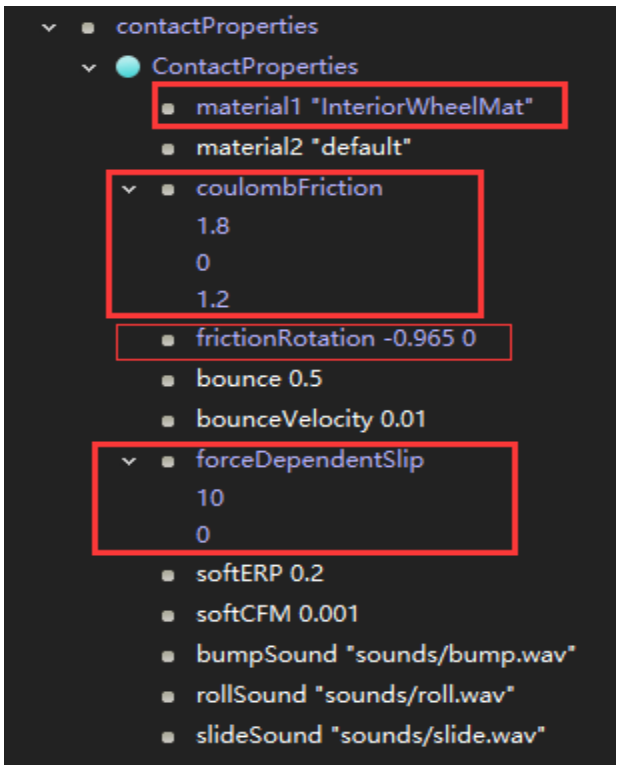
（4）修改碰撞边界形状和物理碰撞边界：boundingObject 添加属性，选择 USE 下刚刚重命名的 Shape 节点即 Body，修改为 bounding Object USE Body。同时给 physics 赋予物理属性 Physics，修改为 physics Physics。

（5）添加铰链，轮子和电机：这里复制助教给的麦轮模型，在助教的 world 中将这四个节点导出，并在我的 world 中导入。至此，我们完成了小车的硬件构造。

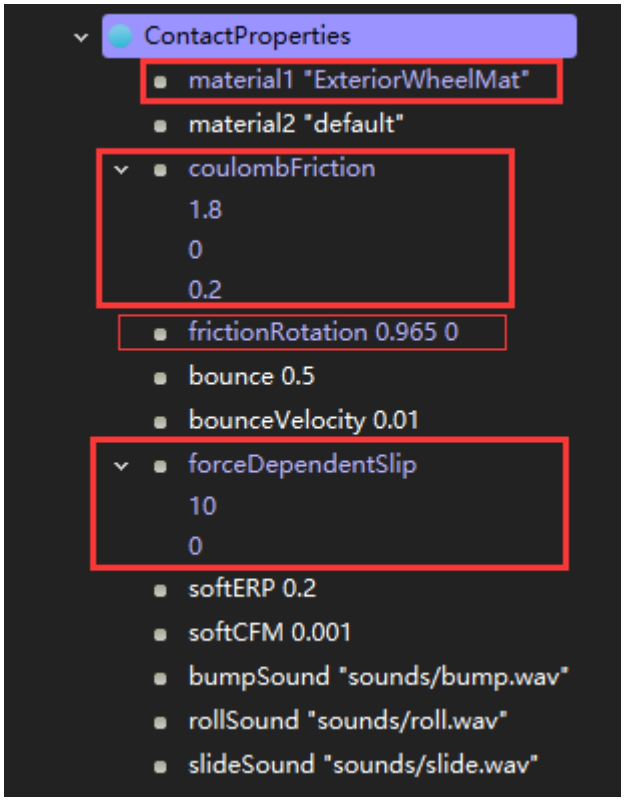
4、麦轮摩擦设置

在 WorldInfo 下的 contactProperties 添加 2 个新节点 ContactProperties，参数修改内容如下：

第 1 个 ContactProperties 节点（见下页，修改内容已框出）：



第 2 个 ContactProperties 节点（见下页，修改内容已框出）：



5、控制器代码

我参考了助教的代码为主体，并手动添加了斜向前进的四个方向键的代码（1：右前，2：右后，3：左前，4：左后）。接下来是对部分代码的解释：

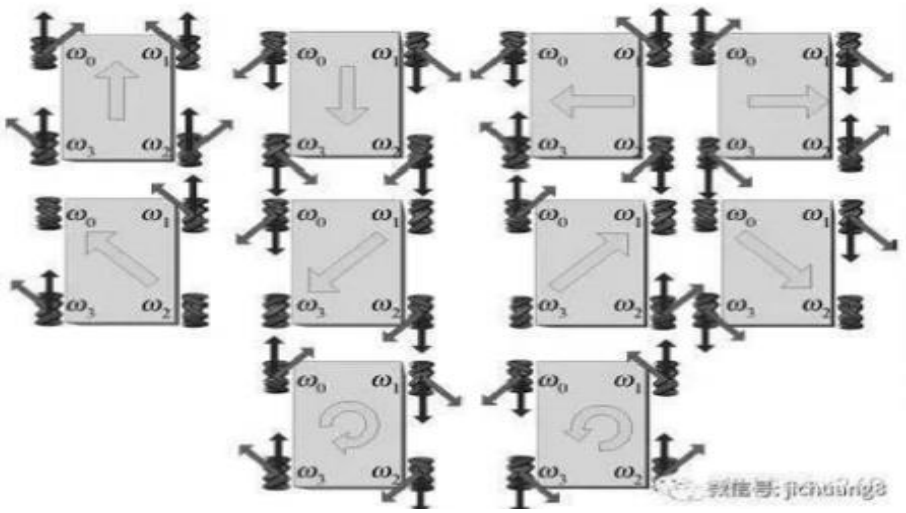
```
    double speed[4];
```

speed 数组用于存储 4 个电机 motors[i]的速度（motors[i]=speed[i]）。

```
    //W:向前  S:向后  A:向左  D:向右  1:右前  2:右后  3:左前  4:左后
    Q:左自旋  E:右自旋
    double speed_forward[4]={velocity,velocity,velocity,velocity};
    double speed_backward[4]={-velocity,-velocity,-velocity,-velocity};
    double speed_leftward[4]={velocity,-velocity,velocity,-velocity};
    double speed_rightward[4]={-velocity,velocity,-velocity,velocity};
    double speed_leftCircle[4]={velocity,-velocity,-velocity,velocity};
    double speed_rightCircle[4]={-velocity,velocity,velocity,-
velocity};
    double speed_rightForward[4]={0,velocity,0,velocity};
    double speed_rightBackward[4]={-velocity,0,-velocity,0};
```

```
    double speed_leftForward[4]={velocity,0,velocity,0};
    double speed_leftBackward[4]={0,-velocity,0,-velocity};
```

这里代表了小车前后左右，斜向和左右自旋时的四个车轮的速度方向，如下图所示：



```
        //根据按键决定电机怎么样转动
        if(keyValue1=='W'){
            for(i=0;i<=3;i++){
                speed[i]=speed_forward[i];
            }
        }
```

根据键盘得到信息将具体的方向数组中的电机速度赋值给 speed 数组。

```
        //让电机执行
        for(i=0;i<=3;i++){
            motors[i]->setVelocity(speed[i]);
        }
    }
```

将 speed 数组的值赋值给 motors 数组，得到 4 个电机 motors[i]的速度。（助教的代码是两个数组：speed1 和 speed2 两个速度分量进行叠加（因为在斜向前进时需要同时按 2 个键，如右前需要同时按 W 和 D）。不过我已经在代码中单独增加了 1,2,3,4 键来实现四个斜向前进，这样使得 8 个方向加左右自旋均可以只按一个键来实现。

所以我的代码可以去除 speed2 数组获取速度的代码部分，且 motors[i]最终只需获取 speed 的速度即可。）

整体代码如下所示：

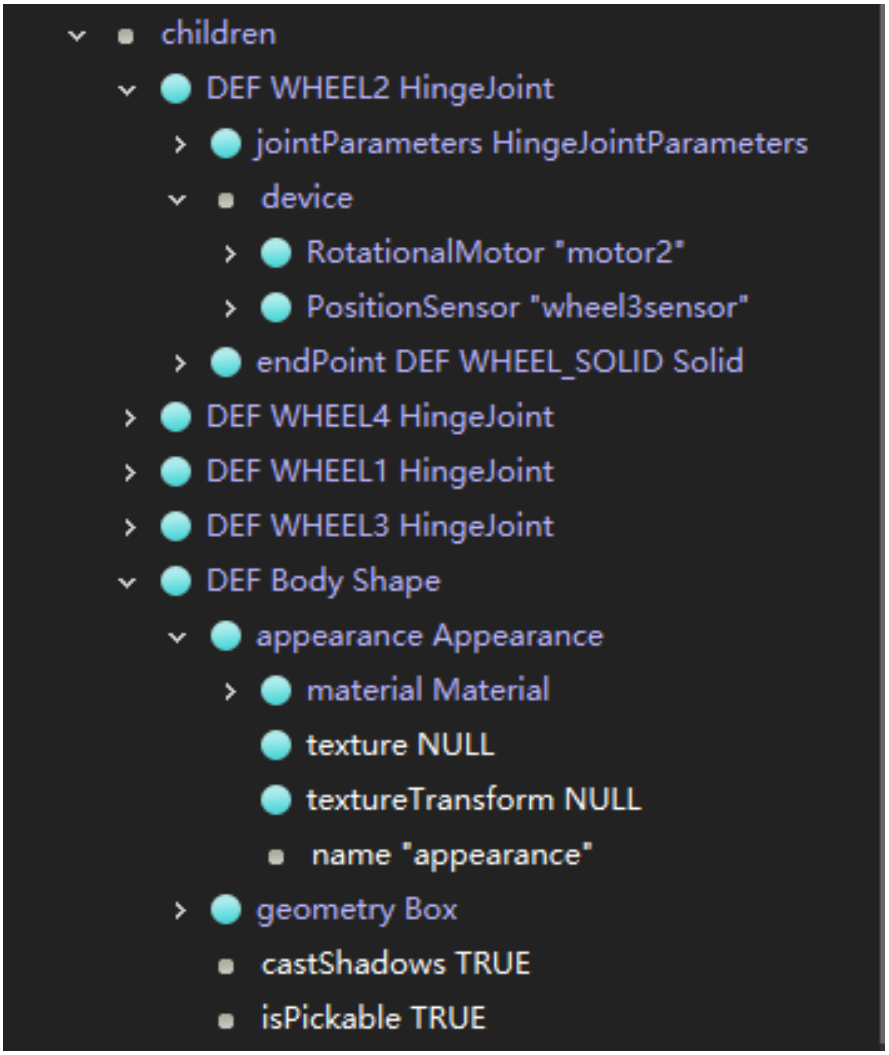
```
#include <webots/Robot.hpp>
#include <webots/Motor.hpp>
#include <webots/Keyboard.hpp>
#include <iostream>
#include <algorithm>
#include <iostream>
#include <limits>
#include <string>

using namespace std;
using namespace webots;

int main() {
    Motor *motors[4];//电机和键盘都要用 webots 给的类型
    webots::Keyboard keyboard;
    char wheels_names[4][8]={"motor1","motor2","motor3","motor4"};//对应
RotationMotor 里的句柄

    Robot *robot=new Robot();//使用 webots 的机器人主体
    keyboard.enable(1);//运行键盘输入设置频率是 1ms 读取一次

    double speed[4];
    double velocity=10;
    int i;

    //初始化
    for(i=0;i<=3;i++){
        motors[i]=robot->getMotor(wheels_names[i]);//按照你在仿真器里面设
置的名字获取句柄
        motors[i]->setPosition(std::numeric_limits<double>::infinity())
;
        motors[i]->setVelocity(0.0);//设置电机一开始处于停止状态
        speed[i]=0;
    }

    //W:向前  S:向后  A:向左  D:向右  1:右前  2:右后  3:左前  4:左后
    Q:左自旋  E:右自旋
    double speed_forward[4]={velocity,velocity,velocity,velocity};
```

```
    double speed_backward[4]={-velocity,-velocity,-velocity,-velocity};
    double speed_leftward[4]={velocity,-velocity,velocity,-velocity};
    double speed_rightward[4]={-velocity,velocity,-velocity,velocity};
    double speed_leftCircle[4]={velocity,-velocity,-velocity,velocity};
    double speed_rightCircle[4]={-velocity,velocity,velocity,-
velocity};
    double speed_rightForward[4]={0,velocity,0,velocity};
    double speed_rightBackward[4]={-velocity,0,-velocity,0};
    double speed_leftForward[4]={velocity,0,velocity,0};
    double speed_leftBackward[4]={0,-velocity,0,-velocity};

    int timeStep=(int)robot->getBasicTimeStep();//获取你在 webots 设置一帧
的时间
    cout<<timeStep<<endl;

    while(robot->step(timeStep)!=-1){//仿真运行一帧
        //获取键盘输入，这样写可以获得同时按下的按键（最多支持 7 个）
        int keyValue1=keyboard.getKey();
        int keyValue2=keyboard.getKey();
        cout<<keyValue1<<":"<<keyValue2<<endl;//用于检验是否正确接收到了按
键信息
        //根据按键决定电机怎么样转动
        if(keyValue1=='W'){
            for(i=0;i<=3;i++){
                speed[i]=speed_forward[i];
            }
        }
        else if(keyValue1=='S'){
            for(i=0;i<=3;i++){
                speed[i]=speed_backward[i];
            }
        }
        else if(keyValue1=='A'){
            for(i=0;i<=3;i++){
                speed[i]=speed_leftward[i];
            }
        }
        else if(keyValue1=='D'){
            for(i=0;i<=3;i++){
                speed[i]=speed_rightward[i];
            }
        }
        else if(keyValue1=='Q'){
            for (i=0;i<=3; i++){
```

```
                speed[i]=speed_leftCircle[i];
            }
        }
        else if(keyValue1=='E'){
            for (i=0;i<=3;i++){
                speed[i]=speed_rightCircle[i];
            }
        }
        else if(keyValue1=='1'){
            for (i=0;i<=3; i++){
                speed[i]=speed_rightForward[i];
            }
        }
        else if(keyValue1=='2'){
            for(i=0;i<=3;i++){
                speed[i]=speed_rightBackward[i];
            }
        }
        else if(keyValue1=='3'){
            for(i=0;i<=3;i++){
                speed[i]=speed_leftForward[i];
            }
        }
        else if(keyValue1=='4'){
            for(i=0;i<=3;i++){
                speed[i]=speed_leftBackward[i];
            }
        }
        else{
            for(i=0;i<=3;i++){
                speed[i]=0;
            }
        }

        //让电机执行
        for(i=0;i<=3;i++){
            motors[i]->setVelocity(speed[i]);
        }
    }
    return 0;
}
```

三、实验结果与分析

由于实验结果为动态，所以这里仅展示 children 节点的结构：



四、实验中的问题和解决方法

由于初期对软件操作不熟练以及看 PPT 不够细致，产生了如下问题：

1、控制器代码复制到窗口后，直接点了仿真按钮，发现小车不动且 Console 无键盘值输出，观看 b 站视频后才得知需要先点击代码上方的齿轮按钮进行编译并修改 controller“ ”引号内的执行程序名才可以正常运行。

2、修改摩擦系数和材质后发现小车仍然只能前进和后退，不能左右和斜向，反复对比后才发现在第一个 ContactProperties 节点下的 material1 材质写成了“InteriorwheelMat”（w 没大写），修改为“InteriorWheelMat”之后即可实现所有功能。

3、针对问题 2，我一开始以为是我的 Robot 和 CircleArena 节点有参数设置不对，于是决定复制导入助教的对应节点，在单独导入 CircleArena 节点时，我先删除了自己的 CircleArena，发现小车直接掉了下去，然后我重置了仿真时间也没有让小车出现。只好重新操作，即先导入 CircleArena，再删除原有的 CircleArena。不过后面发现和 Robot，CircleArena 无关，问题出在了第 2 点上。