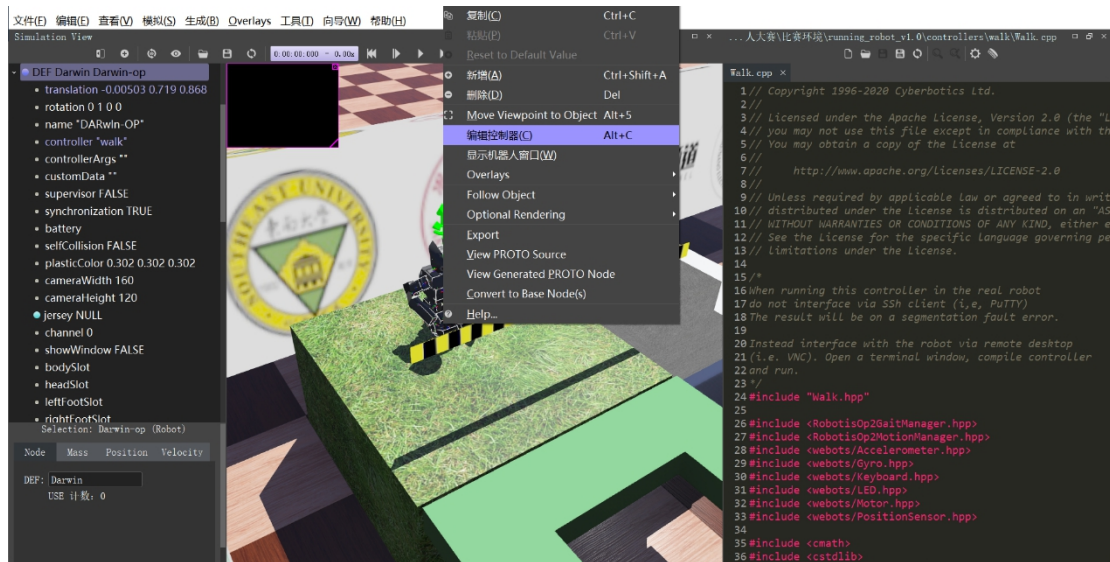


用 C 代码控制机器人的运动

1 对控制器代码的分析及说明

打开 RunningRobotEnv_v1.0.wbt 文件，选中机器人，并单击右键，选择“编辑控制器”，在右侧窗口打开 Walk.cpp 文件。



在 Walk.cpp 中，首先定义一些关键常量，代表了机器人的一些运动的关节：

```
44 static const char *motorNames[NMOTORS] = {
45     "ShoulderR" /*ID1*/, "ShoulderL" /*ID2*/, "ArmUpperR" /*ID3*/, "ArmUpperL" /*ID4*/, "ArmLowerR" /*ID5*/,
46     "ArmLowerL" /*ID6*/, "PelvYR" /*ID7*/, "PelvYL" /*ID8*/, "PelvR" /*ID9*/, "PelvL" /*ID10*/,
47     "LegUpperR" /*ID11*/, "LegUpperL" /*ID12*/, "LegLowerR" /*ID13*/, "LegLowerL" /*ID14*/, "AnkleR" /*ID15*/,
48     "AnkleL" /*ID16*/, "FootR" /*ID17*/, "FootL" /*ID18*/, "Neck" /*ID19*/, "Head" /*ID20*/
49 };
```

在 Robot 函数中设置机器人的初始状态，如头顶和眼睛 LED 的颜色等初始参数，并初始化键盘输入和运动管理器 motion manager 等：

```
1. Walk::Walk() : Robot() {
2.     mTimeStep = getBasicTimeStep();
3.
4.     getLED("HeadLed")->set(0xFF0000);
5.     getLED("EyeLed")->set(0x00FF00);
6.     mAccelerometer = getAccelerometer("Accelerometer");
7.     mAccelerometer->enable(mTimeStep);
8.
9.     getGyro("Gyro")->enable(mTimeStep);
10.
```

```

11. for (int i = 0; i < NMOTORS; i++) {
12.     mMotors[i] = getMotor(motorNames[i]);
13.     string sensorName = motorNames[i];
14.     sensorName.push_back('S');
15.     mPositionSensors[i] = getPositionSensor(sensorName);
16.     mPositionSensors[i]->enable(mTimeStep);
17. }
18.
19. mKeyboard = getKeyboard();
20. mKeyboard->enable(mTimeStep);
21.
22. mMotionManager = new RobotisOp2MotionManager(this);
23. mGaitManager = new RobotisOp2GaitManager(this, "config.ini");
24. }

```

myStep 函数可以使机器人运动一步：

```

1. void Walk::myStep() {
2.     int ret = step(mTimeStep);
3.     if (ret == -1)
4.         exit(EXIT_SUCCESS);
5. }

```

wait 函数用于使机器人等待一段时间：

```

1. void Walk::wait(int ms) {
2.     double startTime = getTime();
3.     double s = (double)ms / 1000.0;
4.     while (s + startTime >= getTime())
5.         myStep();
6. }

```

run 函数用于控制机器人的连续运动：

```

1. // function containing the main feedback loop
2. void Walk::run() {
3.     cout << "The robot will automatically take a few steps" << endl;
4.
5.     // First step to update sensors values

```

```

6.   myStep();
7.
8.   // play the hello motion
9.   mMotionManager->playPage(9); // init position
10.  wait(200);
11.
12.  // main loop
13.  while (true) {
14.      checkIfFallen();
15.
16.      mGaitManager->setXAmplitude(0.0);
17.      mGaitManager->setAAmplitude(0.0);
18.
19.      mGaitManager->start();
20.      mGaitManager->setXAmplitude(1.0);
21.
22.      mGaitManager->step(mTimeStep);
23.
24.      // step
25.      myStep();
26.  }
27. }

```

在仿真器开始运行时，首先运行与 Walk.cpp 文件在同一个文件夹中的 main.cpp 文件，其中调用了 run 函数。在 run 函数中，motion_manager 用于使机器人站立，然后控制器进入无限 while 循环。循环中要做的第一件事是检查机器人是否没有掉落，即调用 checkIfFallen 函数，它是通过使用加速度计来实现的。以上是经过修改的 run 函数，使机器人不用进行键盘控制直接运动起来，后续会介绍如何进行修改。

checkIfFallen 函数用于检测机器人是否摔倒：

```

1. void Walk::checkIfFallen() {
2.     static int fup = 0;
3.     static int fdown = 0;
4.     static const double acc_tolerance = 80.0;
5.     static const double acc_step = 100;
6.
7.     // count how many steps the accelerometer
8.     // says that the robot is down
9.     const double *acc = mAccelerometer->getValues();
10.    if (acc[1] < 512.0 - acc_tolerance)

```

```

11.     fup++;
12. else
13.     fup = 0;
14.
15. if (acc[1] > 512.0 + acc_tolerance)
16.     fdown++;
17. else
18.     fdown = 0;
19.
20. // the robot face is down
21. if (fup > acc_step) {
22.     mMotionManager->playPage(10); // f_up
23.     mMotionManager->playPage(9);  // init position
24.     fup = 0;
25. }
26. // the back face is down
27. else if (fdown > acc_step) {
28.     mMotionManager->playPage(11); // b_up
29.     mMotionManager->playPage(9);  // init position
30.     fdown = 0;
31. }
32. }

```

其中设置了机器人面部朝下摔倒和背部朝下摔倒两种情况，并根据摔倒情况让机器人重新站立起来，重新初始化运动状态。

2 对代码进行修改来控制机器人的运动

我们尝试对代码样例 Walk.cpp 进行修改，让机器人可以自动运动起来，而不用键盘对其进行控制。将函数 void Walk::run() 中的 bool isWalking = false; 语句删去，并主循环 while 的内容修改为如下：

```


1. while (true) {
2.     checkIfFallen();
3.
4.     mGaitManager->setXAmplitude(0.0);
5.     mGaitManager->setAAmplitude(0.0);
6.
7.     mGaitManager->start();
8.     mGaitManager->setXAmplitude(1.0);
9.
10.    mGaitManager->step(mTimeStep);

```

```

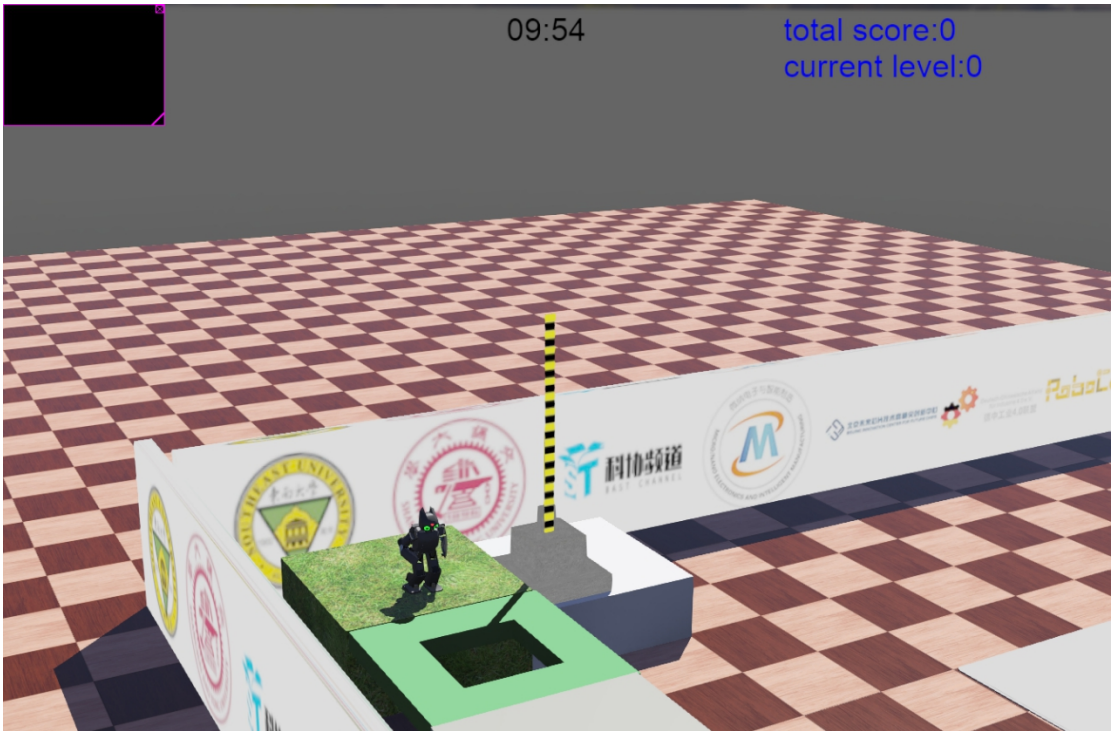
11.
12.     // step
13.     myStep();
14. }

```

然后按住 ctrl+s 进行保存，并点击右上角的 进行编译，如下图。



然后重新运行仿真，即可看到机器人开始自动前进，无需进行键盘控制，如下图。



代码中， `mGaitManager->start();`让机器人开始运动，`mGaitManager->setXAmplitude(1.0);`让机器人向前方运动。由于 `while` 循环会一直运行，因此机器人会一直向前运动下去。若想让机器人在适当位置停止，可以修改 `while` 循环的条件。

3 常用函数列表

函数名称	用途	备注
<code>mGaitManager->start ()</code>	启动	
<code>mGaitManager->stop ()</code>	停止	
<code>mGaitManager->step (int t)</code>	运行一段时间后 停止	t 的单位为毫秒

mGaitManager->setXAmplitude (double X)	前进/后退	X 影响脚步向前的长度，它可以取-1 到 1 之间的任何值
mGaitManager->setYAmplitude (double Y)	左移/右移	Y 影响脚步在侧面方向上的长度，它可以取-1 到 1 之间的任何值
mGaitManager->setAAmplitude (double A)	左转/右转	A 影响步态的角度并允许机器人在行走过程中旋转，它可以取 0 到 1 之间的任何值
wait (int t)	等待	t 的单位为毫秒
mMotionManager->playPage(9)	准备好了	playPage 为一系列封装好的机器人动作文件
mMotionManager->playPage(10)	站起来	初始状态为机器人面部朝下，让机器人站起来
mMotionManager->playPage(11)	站起来	初始状态为机器人背部朝下，让机器人站起来