仿真使用

注意：请先关闭 **conda**，退出 **conda** 环境，不然可能会有意想不到的错误 !!!

1. 安装 docker,ros galactic

   https://docs.docker.com/engine/install/ubuntu/

   https://docs.ros.org/en/galactic/Installation/Ubuntu-Install-Debians.html

2. 下载比赛压缩包

   https://pan.educg.net/s/L5myHM

3. 本地导入 Docker 镜像（注意这只加载了镜像，但没有运行容器）

   sudo docker load -i cyberdog_raceV2.tar

4. 运行 Docker 镜像来创建一个容器

   sudo docker run -it –privileged=true -e DISPLAY=$DISPLAY -v/tmp/.X11-unix:/tmp/.X11-unix cyberdog_sim:v2

5. 首先我们要把这个比赛 docker 展开，方便修改代码
   - sudo docker ps -a 用来显示所有容器的信息

```
luke@Akubi:~/projects/dog_sim$ sudo docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED      STATUS                      PORTS     NAMES
2096cacd4316   cyberdog_sim:v2   "/ros_entrypoint.sh …"   2 days ago   Exited (254) 2 days ago               friendly_clarke
```

6. 使用 docker cp 将目录复制到本地
   - docker cp 2096cacd4316:/home/cyberdog_sim /projects/sim

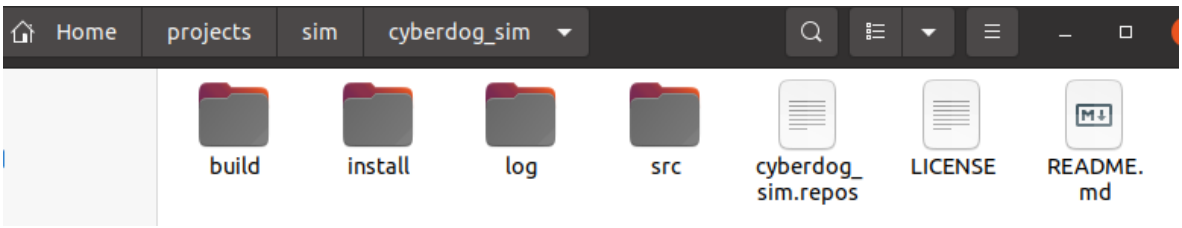
   /home/cyberdog_sim 是容器的源目录， /projects/sim 是本地目录 但是文件会有锁，无法更改文件，需要更改文件权限方便后续操作

   - sudo chown -R username filename

   username 就是你 Ubuntu 系统安装的时候取得名字，filename 就是被锁文件夹名字。

   例如 sudo chown -R coco libbpf 这个时候 libbpf 文件夹就被解锁而且里面的内容都不会被锁，也就是相当于是全部解锁。

7. 进入 /projects/sim/cyberdog_sim 文件夹，阅读 REDAME.md,安装依赖



   注意安装 lcm 的时候可能报错

```
luke@akubi:~/Desktop/lcm/build$ cmake -DLCM_ENABLE_JAVA=ON ..
-- Building LCM 1.5.1 (ABI v1)
-- Could NOT find Python3 (missing: Development.Module) (found version "3.8.10")
-- CPack: Packages will be placed under /home/luke/Desktop/lcm/build/packages
-- Configuring done
-- Generating done
-- Build files have been written to: /home/luke/Desktop/lcm/build
```

可以不用管，直接 make 就行

8. 仿真使用

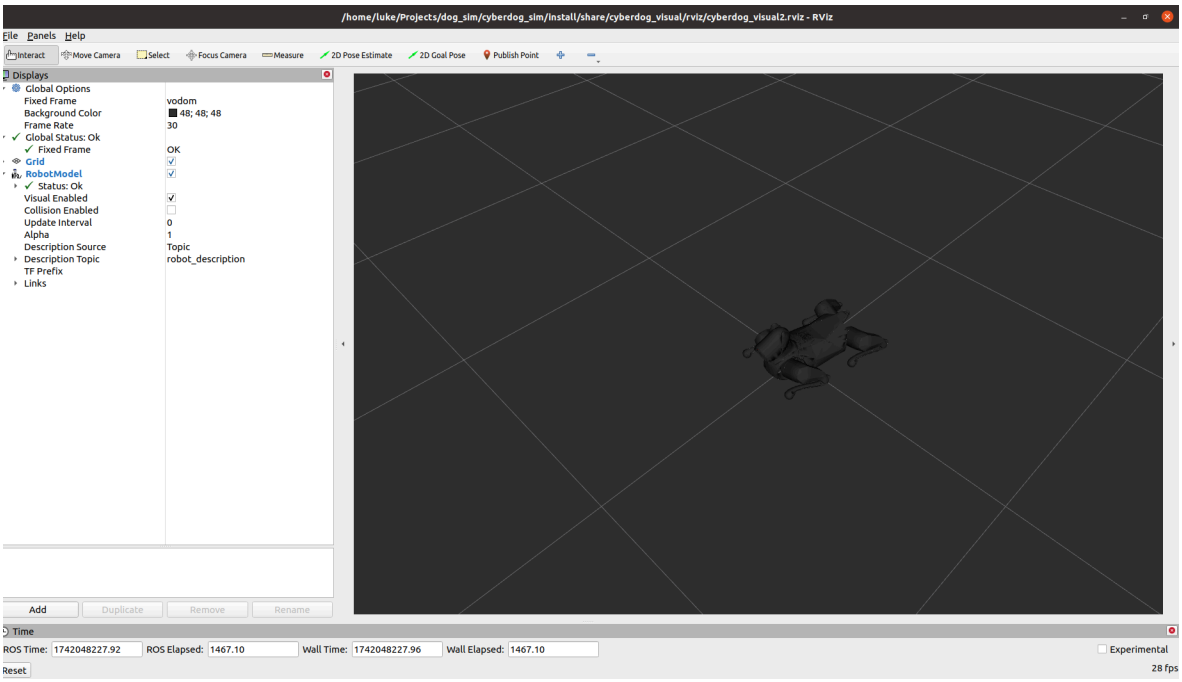
详见 https://miroboticslab.github.io/blogs/#/cn/cyberdog_gazebo_cn

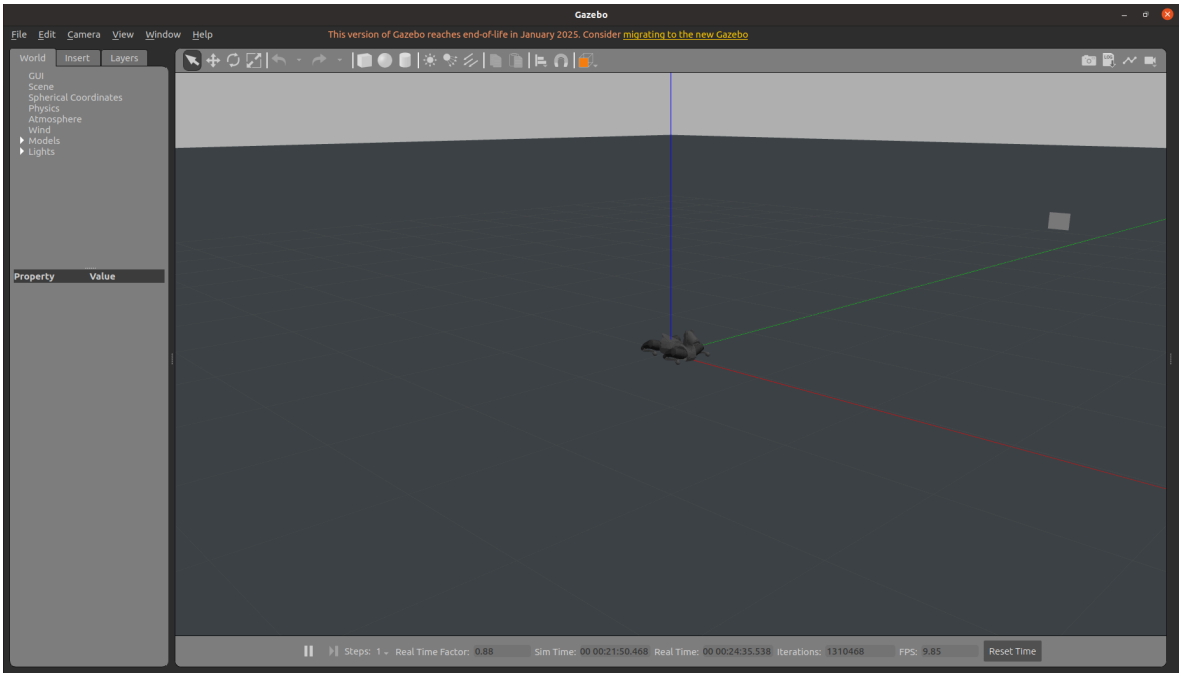
```
## 下载
$ git clone https://github.com/MiRoboticsLab/cyberdog_sim.git
因为我们是从比赛压缩包下载的，里面有这个 cyberdog_sim，应该就不用再 git 了，直接 vcs 展开即可
$ cd cyberdog_sim
$ vcs import < cyberdog_sim.repos

## 编译
需要将 src/cyberdog_locomotion/CMakeLists.txt 中的 BUILD_ROS 置为 ON
需要在 cyberdog_sim 文件夹下进行编译

$ source /opt/ros/galactic/setup.bash
$ colcon build --merge-install --symlink-install --packages-up-to
cyberdog_locomotion cyberdog_simulator


## 使用
需要在 cyberdog_sim 文件夹下运行

$ python3 src/cyberdog_simulator/cyberdog_gazebo/script/launchsim.py
```

编译的时候可能会出现这个 stderr，不用管

```
/home/luke/Projects/dog_sim/cyberdog_sim/src/cyberdog_simulator/cyberdog_gazebo/incl
   57 |     DECLARE_PARAMETER( Vec3< double >, se_ori_cali_offset );
      |                                                           ^
      |                                                           -
---
Finished <<< cyberdog_gazebo [16.2s]
Starting >>> cyberdog_simulator
Finished <<< cyberdog_simulator [0.47s]
[Processing: cyberdog_locomotion]locomotion:build 41% - 51.5s]
[Processing: cyberdog_locomotion]
[Processing: cyberdog_locomotion]
--- stderr: cyberdog_locomotion
**** Onboard Build disabled ****
---
Finished <<< cyberdog_locomotion [2min 2s]

Summary: 7 packages finished [2min 2s]
  3 packages had stderr output: cyberdog_gazebo cyberdog_locomotion cyberdog_visual
```

此时应该会弹出来几个控制台和 GUI 界面，但是此时比赛场景加载不出来





接着见官方文档的 2.2.4 仿真例程

在仿真程序中提供了 cyberdog_example 的仿真例程包，该包提供了 keybroad_commander 和 cyberdogmsg_sender 两个例程。 keybroad_commander 演示了如何使用 gampad_lcmt 向控制发送基本控制指令 该程序运行方法如下： 需要在 cyberdog_sim 文件夹下运行

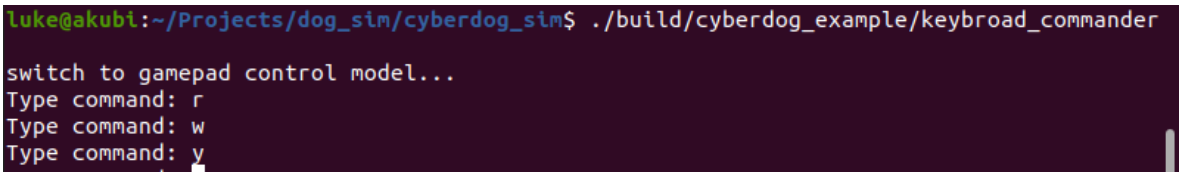
```
$ source /opt/ros/galactic/setup.bash
$ source install/setup.bash
$ ./build/cyberdog_example/keybroad_commander
```

运行后，可在终端输入对应的指令来控制机器人

键位 指令 键位 指令

w	x 方向速度增加最大速度的 0.1 倍	i	pitch 方向速度增加最大速度的 0.1 倍
s	x 方向速度减少最大速度的 0.1 倍	k	pitch 方向速度减少最大速度的 0.1 倍
d	y 方向速度增加最大速度的 0.1 倍	l	yaw 方向速度增加最大速度的 0.1 倍
a	y 方向速度减少最大速度的 0.1 倍	j	yaw 方向速度减少最大速度的 0.1 倍
e	切换为 QP 站立模式(kp kd 较小)	t	切换为缓慢趴下模式
r	切换为 locomotion 模式	y	切换为恢复站立模式

输入 r，进入键盘控制，输入 w，狗向前移动，输入 y，狗恢复站立。



cyberdogmsg_sender 演示了使用/yaml_parameter 来对 yaml 文件中的控制参数进行实时修改，以及使用/apply_force 来仿真中的机器人施加外力。 该程序的运行方法如下： 需要在 cyberdog_sim 文件夹下运行

```
$ source /opt/ros/galactic/setup.bash
$ source install/setup.bash
$ ./build/cyberdog_example/cyberdogmsg_sender
```

该例程先把参数 use_rc 置为 0(该参数为 1 时为遥控模式，置为 0 后才能够通过仿真程序进行控制)； 然后通过设置 control_mode 参数使机器人站立起来，并进入 locomotion 模式，即原地踏步(control_mode 的参数可参阅控制程序的 control_flag.h 文件)； 接着对机器人的左前小腿施加侧向的外力； 最后通过修改 des_roll_pitch_height 参数使机器人在踏步时 roll 角变为 0.2 弧度。