目录

1. 仿真使用
环境安装
解决打开时赛道环境不显示的问题
解决箭头、二维码不显示的问题
rgb摄像头相关仿真使用
赛道仿真运行

# 1. 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！！！*

## 1.1. 相关环境配置

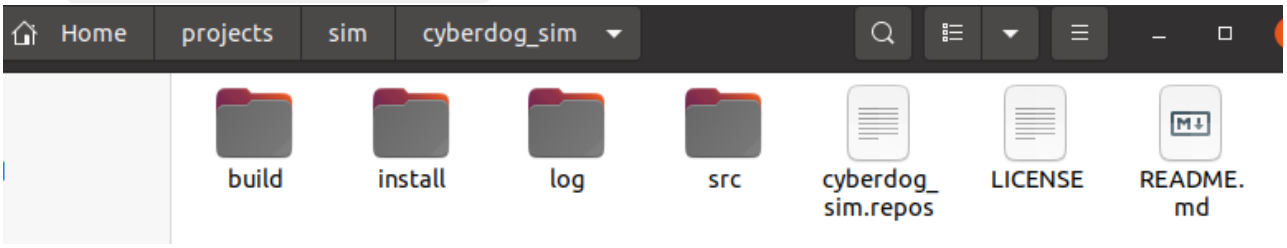
- 安装docker,ros galactic
  https://docs.docker.com/engine/install/ubuntu/
  https://docs.ros.org/en/galactic/Installation/Ubuntu-Install-Debians.html
- 下载比赛压缩包
  https://pan.educg.net/s/L5myHM
- 本地导入 Docker 镜像（注意这只加载了镜像，但没有运行容器）
  sudo docker load -i cyberdog_raceV2.tar
- 运行 Docker 镜像来创建一个容器
  sudo docker run -it --privileged=true -e DISPLAY=$DISPLAY -v/tmp/.X11-unix:/tmp/.X11-unix cyberdog_sim:v2
- 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
```

- 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。

- 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

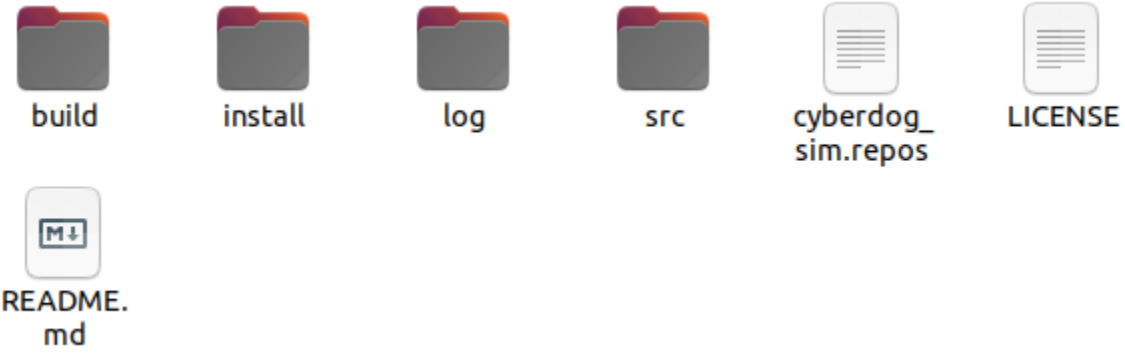
```
luke@akubi:~/Desktop/lcm/build$ cmake -DLCM_ENABLE_JAVA=ON ..
-- Building LCM 1.5.1 (ABI v1)
-- Could NOT find Python3 (missing: Development.Module) (found version "3.8.10")
-- CPack: Packages will be placed under /home/luke/Desktop/lcm/build/packages
-- Configuring done
-- Generating done
-- Build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

可以不用管，直接make就行

- 仿真使用

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

注意编译的时候要把build,intall,log这三个文件夹删掉再编译。

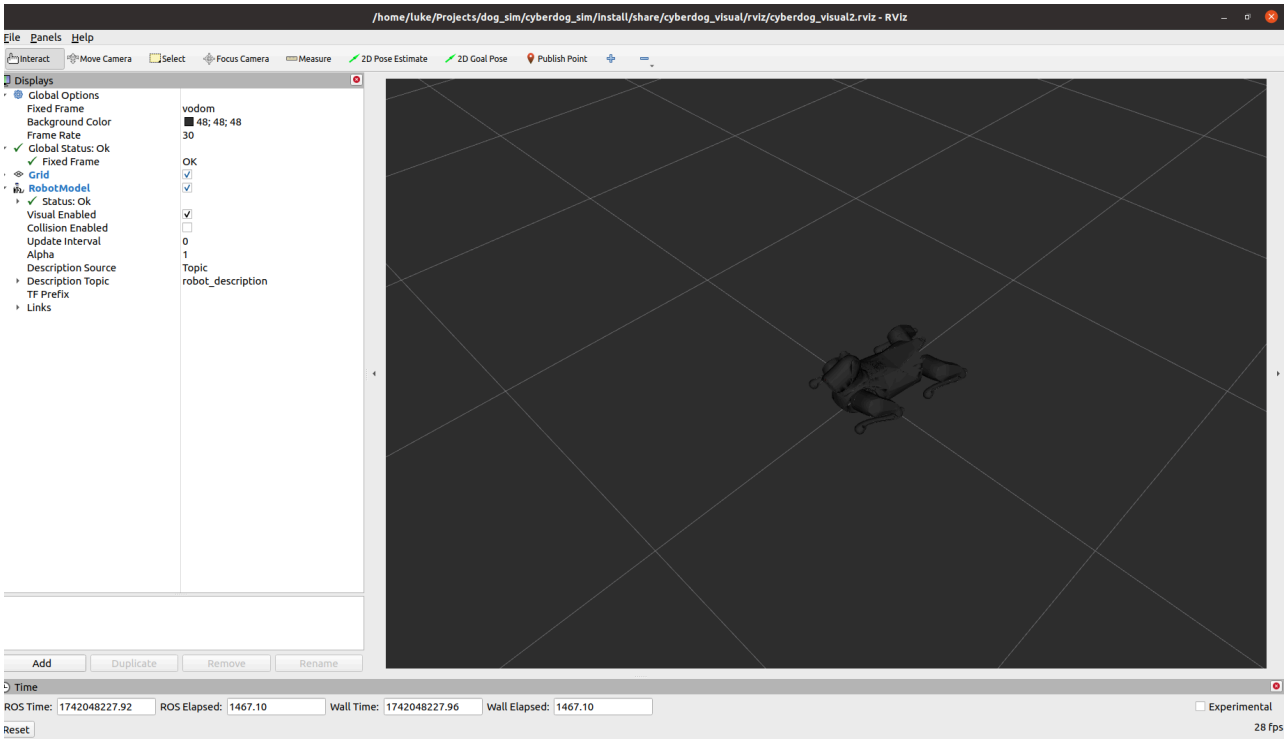


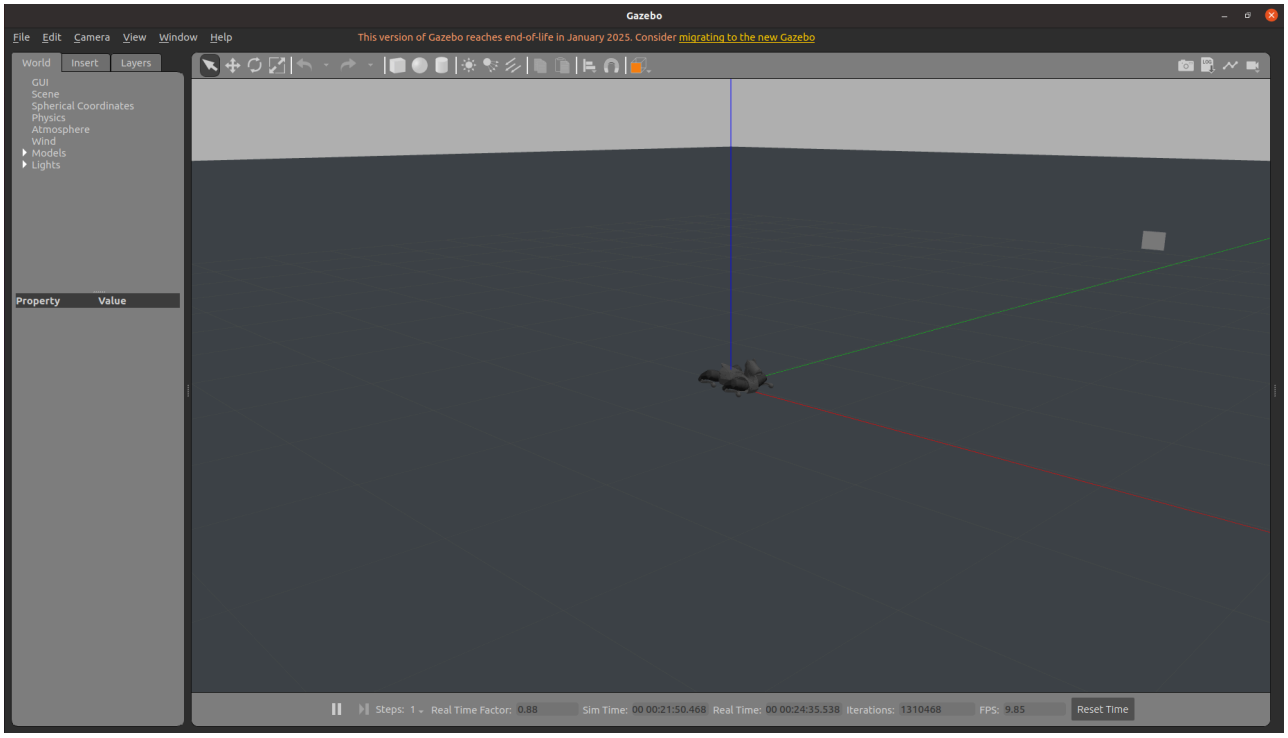
```
## 下载
$ git clone https://github.com/MiRoboticsLab/cyberdog_sim.git
因为我们是从比赛压缩包下载的，里面有这个cyberdog_sim，应该就不用再git了，直接vcs展开即可
$ cd cyberdog_sim
$ vcs import < cyberdog_sim.repos

## 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

## 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

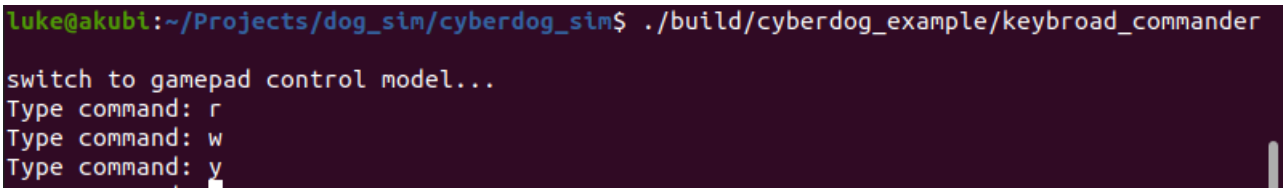
在仿真程序中提供了cyberdog_example的仿真例程包，该包提供了keybroad_commander和cyberdogmsg_sender两个例程。 keybroad_commander演示了如何使用gampad_lcmt向控制发送基本控制指令 该程序运行方法如下： 需要在cyberdog_sim文件夹下运行

```
$ 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。

- 其他例程
  https://miroboticslab.github.io/blogs/#/cn/cyberdog_loco_cn?id=_24-接口示例
  - 基本动作:

在`cyberdog_sim/src/loco_hl_example/basic_motion`中运行`main.py`,控制机 器人依次完成站立，握手，作揖，抬头，低头，原地踏步旋转，趴下等动作。
注意：依赖lcm数据类型文件`robot_control_cmd_lcmt.py`和`robot_control_response_lcmt.py`。

```
luke@akubi:~/Projects/dog_sim/cyberdog_sim/src/loco_hl_example/basic_motion$ python3 main.py
```

- 序列动作:

在`cyberdog_sim/src/loco_hl_example/sequential_motion`内
实现控制机器人依次站立，调整高度，抬起右后腿，原地踏步旋转，趴下等动作
注意：运行该Python脚本，依赖lcm数据类型文件`robot_control_cmd_lcmt.py`和序列动作文件`cyberdog2_ctrl.toml`。

```
luke@akubi:~/Projects/dog_sim/cyberdog_sim/src/loco_hl_example/sequential_motion
$ python3 main.py
0,main.py
1,cyberdog2_ctrl.toml
2,robot_control_cmd_lcmt.py
3,robot_control_cmd_lcmt.cpython-38.pyc
Input a toml ctrl file number:
1
Load file=./cyberdog2_ctrl.toml

robot_control_cmd lcm publish mode : 12 gait_id : 0 msg.duration= 5000
robot_control_cmd lcm publish mode : 21 gait_id : 5 msg.duration= 300
robot_control_cmd lcm publish mode : 21 gait_id : 5 msg.duration= 400
robot_control_cmd lcm publish mode : 21 gait_id : 0 msg.duration= 500
robot_control_cmd lcm publish mode : 21 gait_id : 0 msg.duration= 500
robot_control_cmd lcm publish mode : 21 gait_id : 0 msg.duration= 500
robot_control_cmd lcm publish mode : 21 gait_id : 5 msg.duration= 500
robot_control_cmd lcm publish mode : 11 gait_id : 3 msg.duration= 3000
robot_control_cmd lcm publish mode : 7 gait_id : 0 msg.duration= 4000
!
```

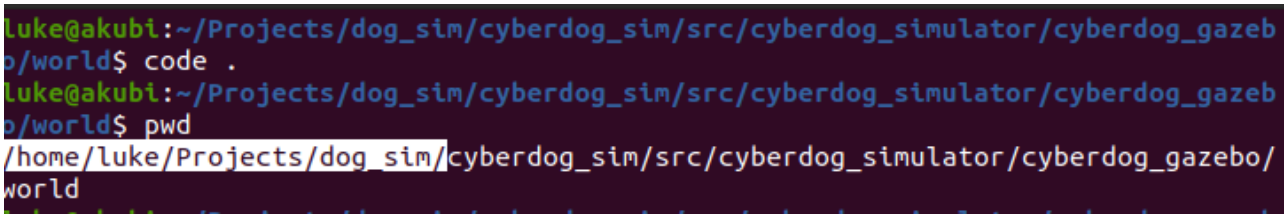
- 自定义步态:

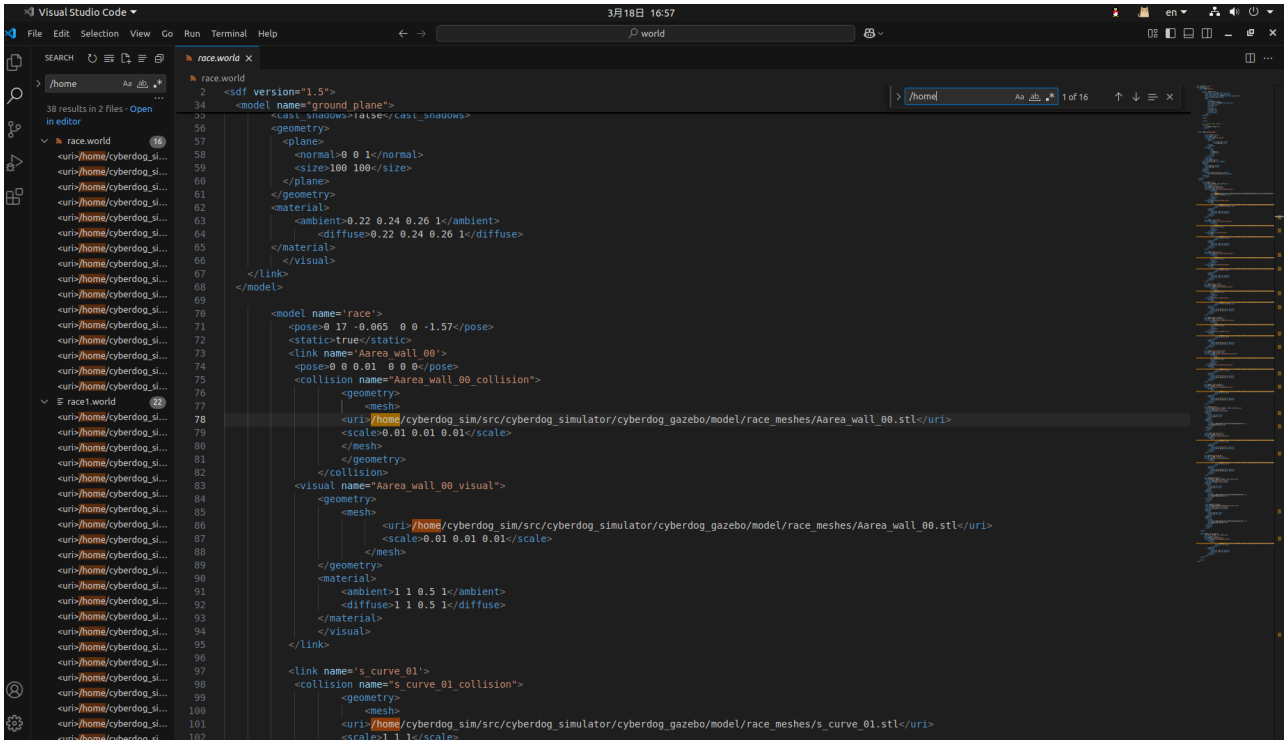
在cyberdog_sim/src/loco_hl_example/customized_gait文件夹中运行main.py.
本例程是Python脚本，通过读取自定义步态文件和序列动作文件，实现控制机器人依次站立，太空步和趴下等动作。示例中Gait_Params_moonwalk.toml 文件包含2.2.2自定义步态相关参数介绍，脚本首先按一定映射关系将其编码为基本robot_control_cmd_lcmt 结构体序列(Gait_Params_moonwalk_full.toml)再下发。
注意：运行该Python脚本，依赖lcm数据类型文件`robot_control_cmd_lcmt.py`和`file_send_lcmt.py`，自定义步态文件`Gait_Def_moonwalk.toml`和`Gait_Params_moonwalk.toml`，以及序列动作文件`Usergait_List.toml`。

# 1.2. 解决打开时赛道环境不显示的问题

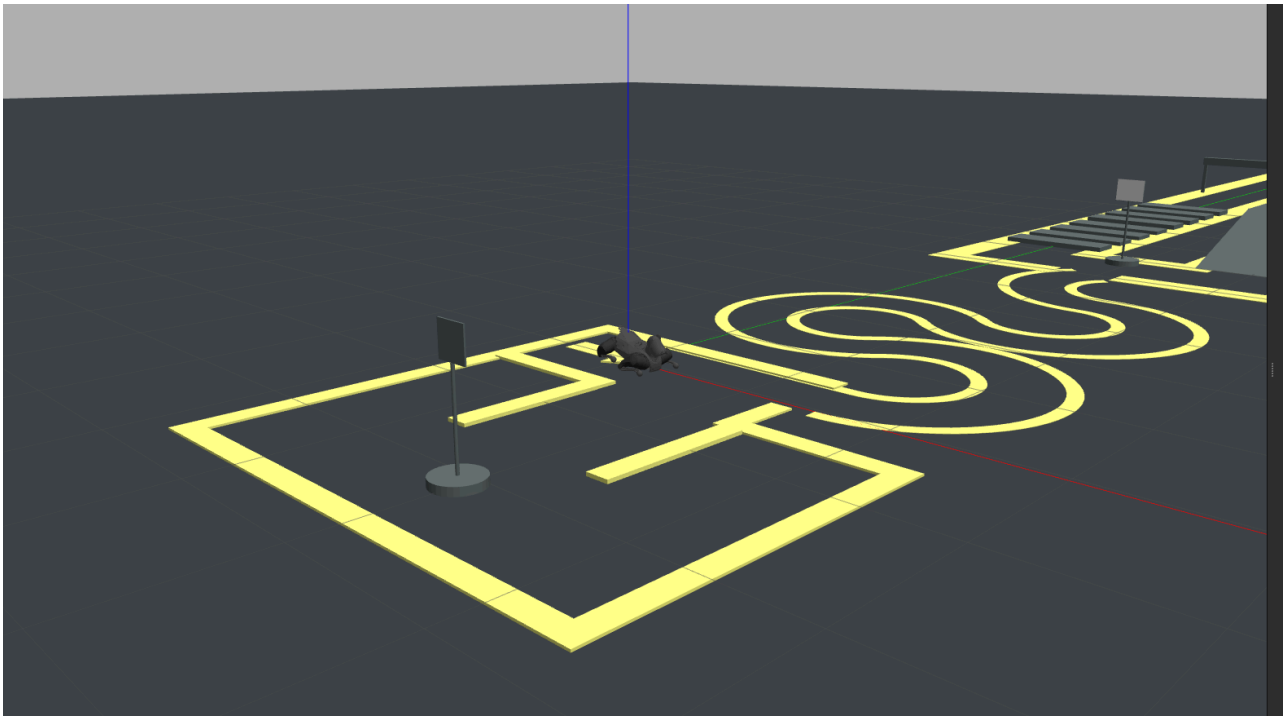
在`cyberdog_sim/src/cyberdog_simulator/cyberdog_gazebo/world`文件夹中，用vscode打开

- `code .`
  然后`ctrl+F`搜索`/home`
  因为`/home`是docker容器里面的路径，这里我们要将他改为自己电脑里面的路径。
  在`cyberdog_sim/src/cyberdog_simulator/cyberdog_gazebo/world`文件夹中，输入pwd显示当前路径，复制cyberdog_sim前面的路径然后替换掉`/home`即可(一定要是绝对路径，不能是相对路径)




再打开仿真，赛道就出现了（此时箭头，二维码仍没有）。



## 1.3. 解决箭头、二维码不显示的问题

压缩包里面有箭头、二维码的png图片，以及一个`gazebo.material`。
因为这些文件不在cyberdogsim文件夹里，所以提取出来没有。

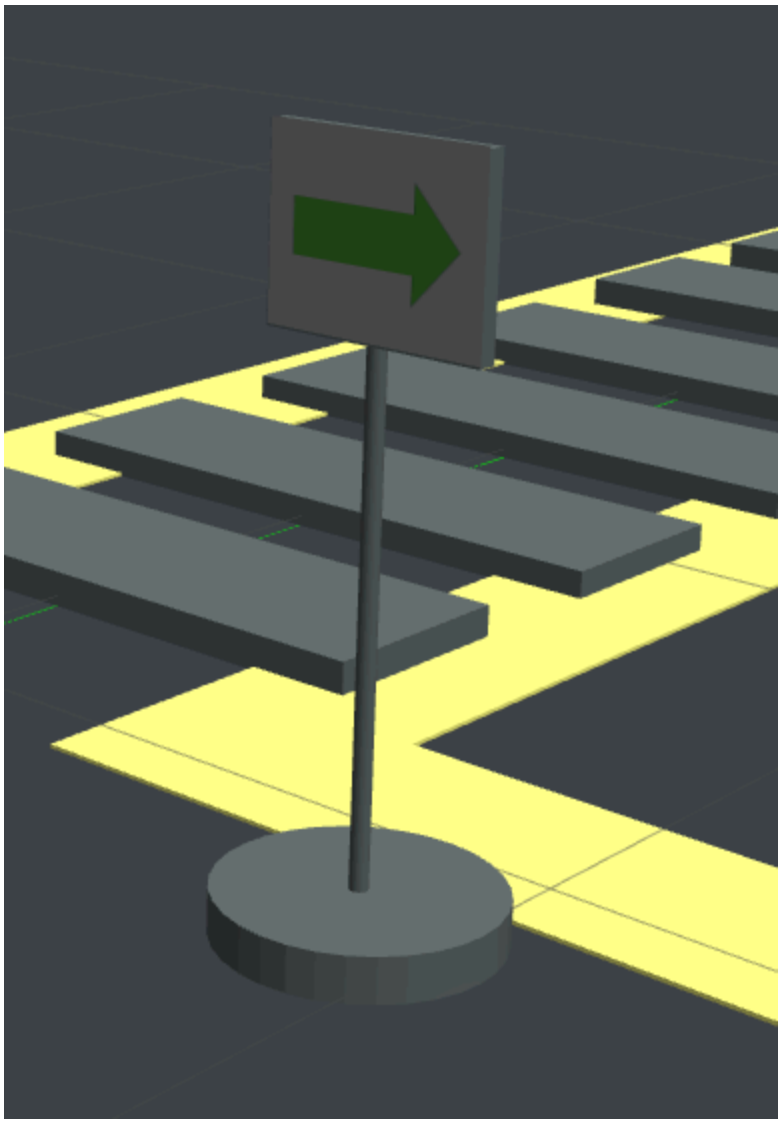
gazebo安装路径一般是：`/usr/share/gazebo-11`

在`/usr/share/gazebo-11/media/materials/scripts`里面替换掉`gazebo.material`同名文件，二维码，箭头图片在放进`/usr/share/gazebo-11/media/materials/textures`里。

让`/usr/share/gazebo-11`获得写权限

- `sudo chmod 777 /usr/share/gazebo-11`

然后打开仿真，发现有箭头，二维码了



注意：石板路有一点问题，想要更改此错误请去比赛官网找对应的解决方案。

# 1.4. rgb摄像头相关仿真使用

## 1.4.1. 在仿真平台中添加image订阅

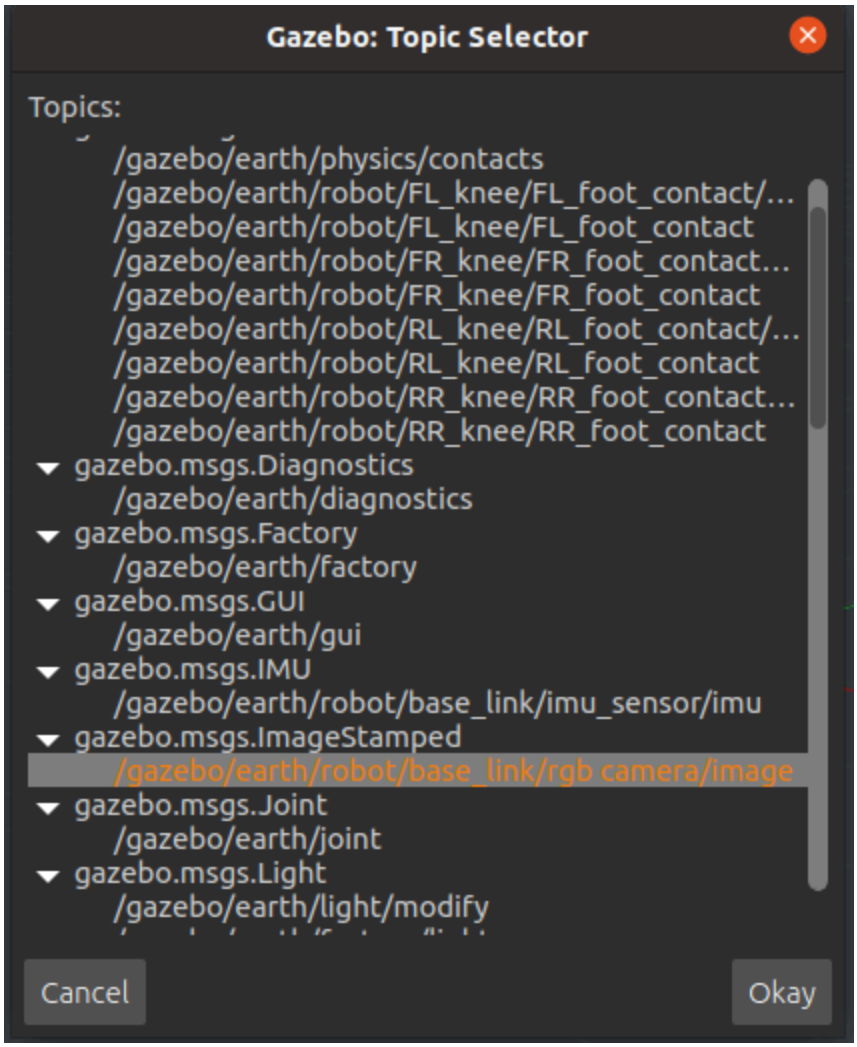
打开ros

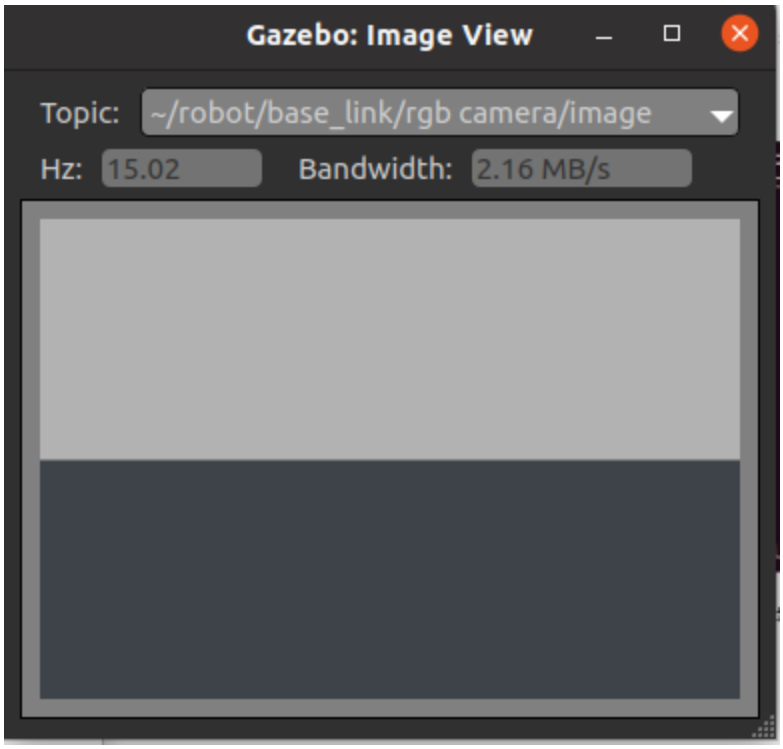
`source /opt/ros/galactic/setup.bash`

在`cyberdog_sim/src/cyberdog_simulator/cyberdog_robot/cyberdog_description/xacro`文件夹里修改`gazebo.xacro`，添加

```
<gazebo reference="RGB_camera_link">
    <sensor type="camera" name="rgb camera">
        <always_on>true</always_on>
        <update_rate>15.0</update_rate>
        <camera name="rgb_camera">
            <horizontal_fov>1.46608</horizontal_fov>
            <image>
                <width>320</width>
                <height>180</height>
                <format>R8G8B8</format>
            </image>
            <distortion>
                <k1>0.0</k1>
                <k2>0.0</k2>
                <k3>0.0</k3>
                <p1>0.0</p1>
                <p2>0.0</p2>
                <center>0.5 0.5</center>
            </distortion>
        </camera>
        <plugin name="rgb_camera_plugin"
        filename="libgazebo_ros_camera.so">
            <ros>
                <!-- <namespace>stereo</namespace> -->
                <remapping>~/image_raw:=image_raw</remapping>
                <remapping>~/camera_info:=camera_info</remapping>
            </ros>
            <!-- Set camera name. If empty, defaults to sensor
            name (i.e. "sensor_name") -->
            <camera_name>rgb_camera</camera_name>
            <!-- Set TF frame name. If empty, defaults to link
            name (i.e. "link_name") -->
            <frame_name>RGB_camera_link</frame_name><hack_baseline>0.2</hack_baseline>
        </plugin>
    </sensor>
</gazebo>
```

然后运行仿真程序，运行仿真程序后可通过 window->Topic Visualization 中找到对应 topic 并打开,可确认rgb 相机正常运行


Gazebo: Topic Selector
Topics:
/gazebo/earth/physics/contacts
/gazebo/earth/robot/FL_knee/FL_foot_contact/...
/gazebo/earth/robot/FL_knee/FL_foot_contact
/gazebo/earth/robot/FR_knee/FR_foot_contact...
/gazebo/earth/robot/FR_knee/FR_foot_contact
/gazebo/earth/robot/RL_knee/RL_foot_contact/...
/gazebo/earth/robot/RL_knee/RL_foot_contact
/gazebo/earth/robot/RR_knee/RR_foot_contact...
/gazebo/earth/robot/RR_knee/RR_foot_contact
gazebo.msgs.Diagnostics
/gazebo/earth/diagnostics
gazebo.msgs.Factory
/gazebo/earth/factory
gazebo.msgs.GUI
/gazebo/earth/gui
gazebo.msgs.IMU
/gazebo/earth/robot/base_link/imu_sensor/imu
gazebo.msgs.ImageStamped
/gazebo/earth/robot/base_link/rgb camera/image
gazebo.msgs.Joint
/gazebo/earth/joint
gazebo.msgs.Light
/gazebo/earth/light/modify
Cancel
Okay

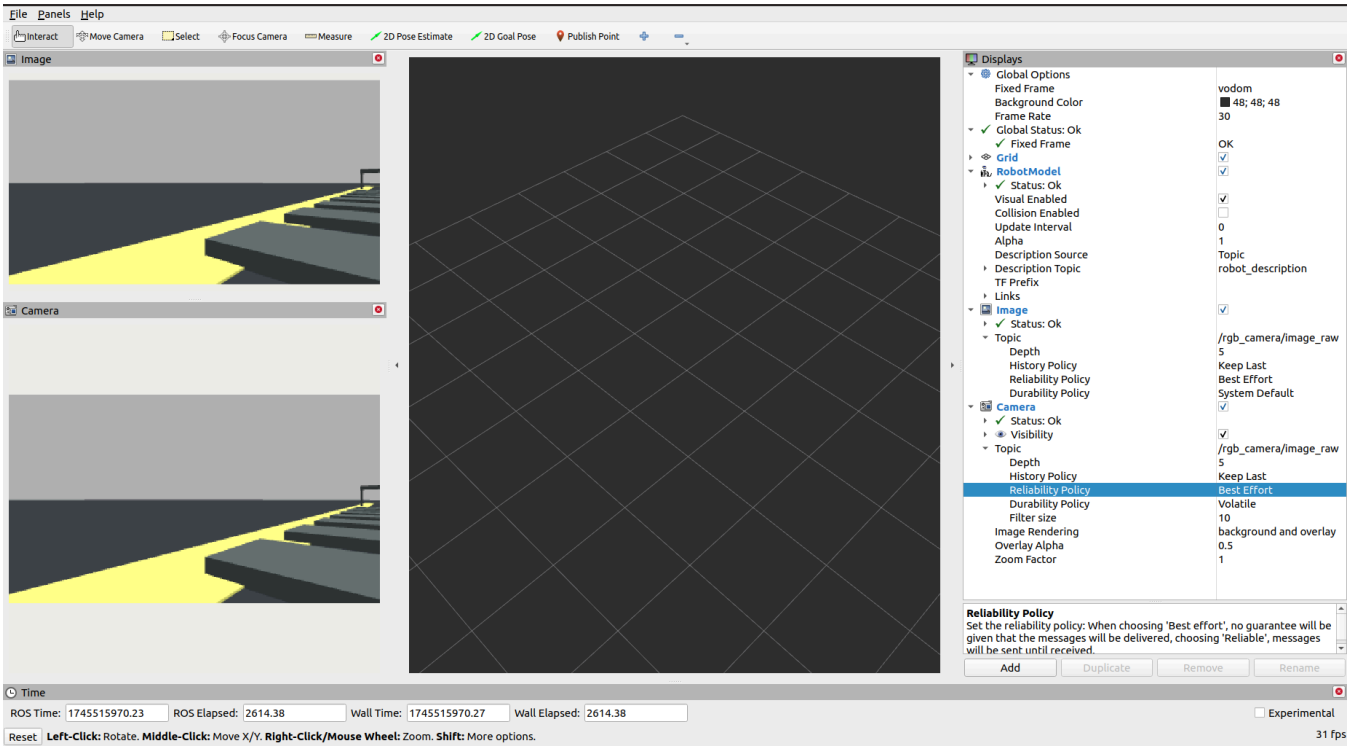



通过 `ros2 topic list` 可确认 topic 正常发送

```
luke@akubi:~/Projects/dog_sim/cyberdog_sim$ ros2 topic list
/opt/ros/galactic/bin/ros2:6: DeprecationWarning: pkg_resources is deprecated as
 an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import load_entry_point
/apply_force
/clicked_point
/clock
/goal_pose
/imu
/initialpose
/joint_states
/parameter_events
/performance_metrics
/rgb_camera/camera_info
/rgb_camera/image_raw
/robot_description
/rosout
/tf
/tf_static
/yaml_parameter
```

## 1.4.2. 在rivz中可视化

在 Add 中 By topic 点击 `/image_raw/Image` 和 `/image_raw/Camera`
在Displays的image中，把topic的Reliability Policy改为Best effort，同理，在Camera中，topic的Reliability Policy改为Best effort。




# 1.5. 赛道仿真运行

相关文件见压缩包。

`main1.py` 依赖于这两个文件 `robot_control_cmd_lcmt.py` 和 `robot_control_response_lcmt.py`，因为是通过lcm传信息的。然后是自定义文件 `CtrlBase.py`， `NodeBase.py`，`TaskRunner.py`。

把 `cyberdog2_user80_ballet.toml` （限高杆），`cyberdog2_user81_ballet.toml` （上坡），`cyberdog2_user82_ballet.toml` （下坡）替换 `cyberdog_sim/src/cyberdog_locomotion/control/motion_list/cyberdog2/user/parameter` 的对应文件。
把 `user_gait_00.toml` （限高杆），`user_gait_01.toml` （上坡），`user_gait_02.toml` （下坡）替换 `cyberdog_sim/src/cyberdog_locomotion/control/motion_list/cyberdog2/user/define` 的对应文件。

注意：执行 `main1.py` 之前不要忘记了激活ros环境

```
source /opt/ros/galactic/setup.bash
```

运行 `main1.py`

```
python3 main1.py
```

然后狗开始运动