1.实验名称及目的

1.1 实验名称

使用行为树控制无人机实验

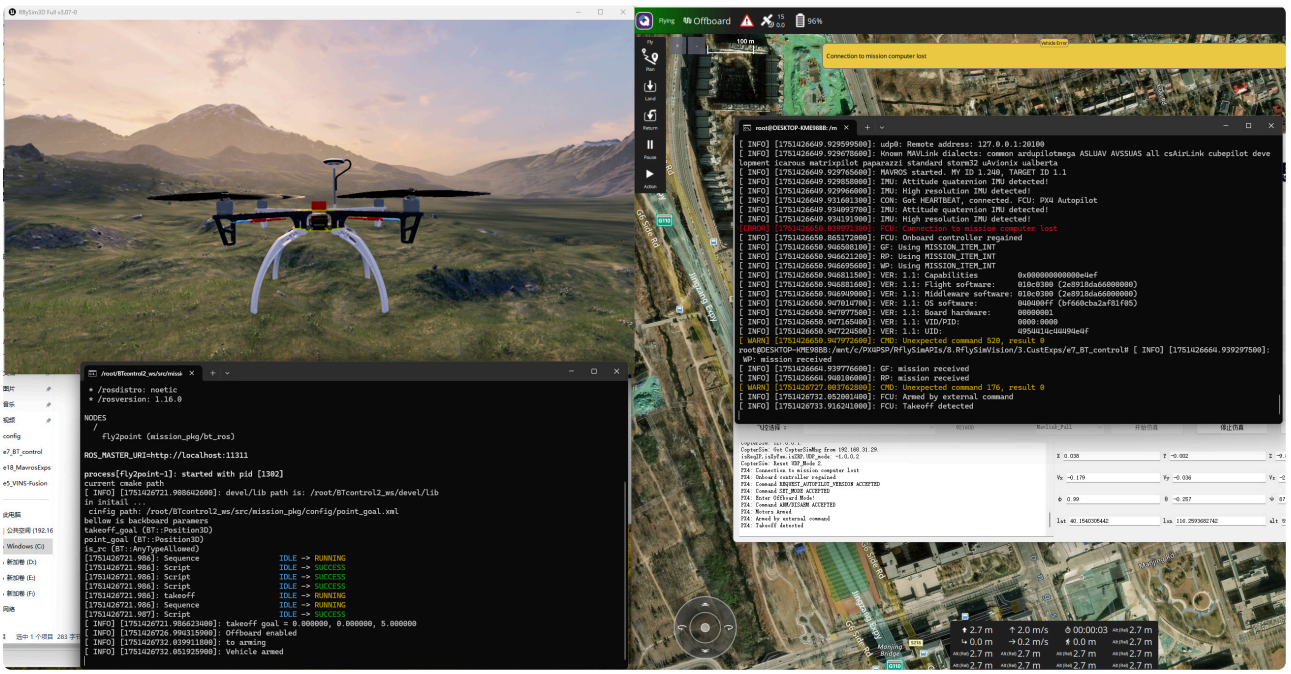
1.2 实验目的

使用行为树实现无人机的起飞、指点飞行和降落功能；自定义行为树节点并使用。

1.3 关键知识点

2.实验效果

实现了无人机按照预设行为树逻辑完成起飞、指点飞行及平稳降落的全过程，自定义的悬停节点使无人机在目标点稳定悬停
10 秒。



3.文件目录

例程目录：[安装目录]\RflySimAPIs\8.RflySimVision\3.CustExps\e7_BT_control

文件夹/文件名称	说明
mission_pkg.zip	已经配置好的mission_pkg包
MavrosTest.py	Mavrosc 测试程序，用于打开Mavros

4.运行环境

序号	软件要求	硬件要求	
		名称	数量
1	Windows 10及以上版本	笔记本/台式电脑	1
2	Visual Studio Code		
3	RflySim 工具链		

5.实验步骤

Step 1: 创建工作空间

双击 WinWSL.bat 进入 WinWSL 的 Ubuntu 环境。在其中输“./Build.sh”，自动拷贝 mission_pkg.zip 到 BTcontrol_ws 并等待编译成功。

Step 2: 打开仿真并启动mavros

在 windows 系统下运行 SITLRunMAVLink.bat，输入飞机数量 1，等待 fixed。然后打开WinWSL.bat运行MavrosTest.py启动mavros。

```
CopterSim: Got 20100 message from 127.0.0.1
CopterSim: MAVLink MSG Received from 20100
sysid,compid,msgid: 1,240,111
CopterSim: TCP port received new connection.
CopterSim: TCP port 4560 connected successfully with SITL
CopterSim: Receive Mavlink heartbeat
PX4: Init MAVLink
PX4: Awaiting GPS/EKF fixed for Position control...
PX4: Found firmware version: 1.12.3dev
PX4: Command ID: 512 ACCEPTED
PX4: [logger] ./log/2025-07-04/01_48_26.ulg
PX4: Command ID: 512 ACCEPTED
PX4: Command ID: 512 DENIED
PX4: Command ID: 512 ACCEPTED
```

Step 3: 行为树实现起飞，指点飞行，降落

再次打开WinWSL.bat，输入下面指令运行send_goal.launch文件，可以观察到无人机先是起飞到了目标点(0,
0, 5)，然后飞行到了目标点(0, 1.5, 1)，最后降落。

```
cd /root/BTcontrol_ws/ source devel/setup.bash roslaunch mission_pkg send_goal.launch
```

```
[ INFO] [1751350841.295435200]: devel/lib path is: /root/BTcontrol_ws/devel/lib
in initail ...
 cinfig path: /root/BTcontrol_ws/src/mission_pkg/config/point_goal.xml
bellow is backboard paramers
takeoff_goal (BT::Position3D)
point_goal (BT::Position3D)
is_rc (BT::AnyTypeAllowed)
[1751350841.401]: Sequence                  IDLE -> RUNNING
[1751350841.401]: Script                    IDLE -> SUCCESS
[1751350841.401]: Script                    IDLE -> SUCCESS
[1751350841.401]: Script                    IDLE -> SUCCESS
[1751350841.401]: takeoff                   IDLE -> RUNNING
[1751350841.401]: Sequence                  IDLE -> RUNNING
[1751350841.401]: Script                    IDLE -> SUCCESS
[ INFO] [1751350841.401171600]: takeoff goal = 0.000000, 0.000000, 5.000000
[ INFO] [1751350846.419090000]: Offboard enabled
[ INFO] [1751350851.464890100]: to arming
[ INFO] [1751350851.484568700]: Vehicle armed
[ INFO] [1751350859.664999700]:  takeoff finished .....
[1751350859.665]: Takeoff                   IDLE -> SUCCESS
[1751350859.666]: Script                    SUCCESS -> IDLE
[1751350859.666]: Takeoff                   SUCCESS -> IDLE
[1751350859.666]: Sequence                  RUNNING -> SUCCESS
[1751350859.666]: Sequence                  SUCCESS -> IDLE
[1751350859.666]: takeoff                   RUNNING -> SUCCESS
[ INFO] [1751350859.666188700]: ========================goal: 0.000000,1.500000,1.000000
[1751350864.317]: ArriveGoal                IDLE -> SUCCESS
[1751350864.320]: land                      IDLE -> RUNNING
[1751350864.320]: Sequence                  IDLE -> RUNNING
[1751350864.321]: SetBlackboard             IDLE -> SUCCESS
[1751350864.321]: SetBlackboard             IDLE -> SUCCESS
[ INFO] [1751350864.321102600]: to land ...
[ INFO] [1751350872.378085000]: change mode (AUTO.LAND)
[1751350872.378]: Land                      IDLE -> SUCCESS
[1751350872.391]: SetBlackboard             SUCCESS -> IDLE
[1751350872.391]: SetBlackboard             SUCCESS -> IDLE
[1751350872.391]: Land                      SUCCESS -> IDLE
[1751350872.391]: Sequence                  RUNNING -> SUCCESS
[1751350872.391]: Sequence                  SUCCESS -> IDLE
[1751350872.391]: land                      RUNNING -> SUCCESS
[1751350872.391]: Script                    SUCCESS -> IDLE
[1751350872.391]: Script                    SUCCESS -> IDLE
[1751350872.391]: Script                    SUCCESS -> IDLE
[1751350872.392]: takeoff                   SUCCESS -> IDLE
[1751350872.392]: ArriveGoal                SUCCESS -> IDLE
[1751350872.392]: land                      SUCCESS -> IDLE
[1751350872.392]: Sequence                  RUNNING -> SUCCESS
[1751350872.392]: Sequence                  SUCCESS -> IDLE
```

起飞的目标点，指点飞行的目标点可以在/BTcontrol_ws/src/mission_pkg/config/point_goal.xml中修改：

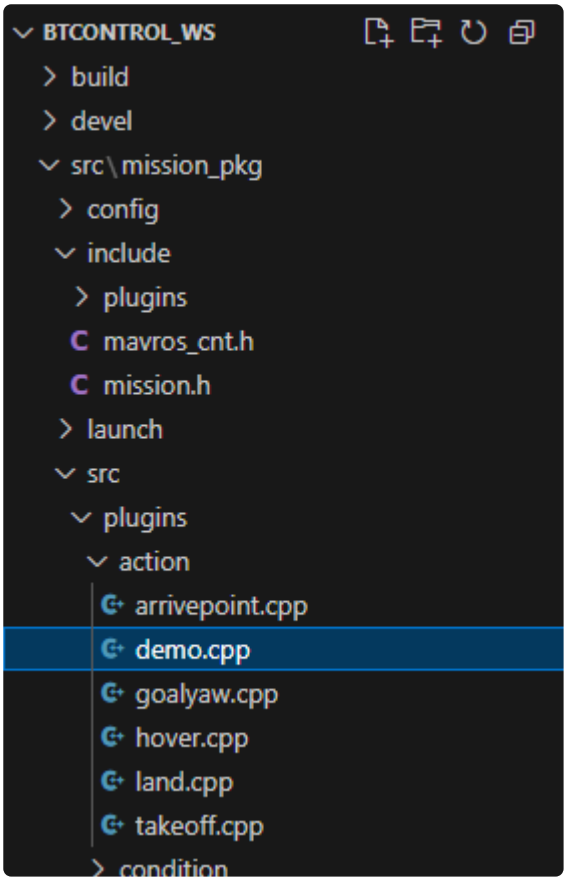
```
<BehaviorTree ID="PointGoal"> <Sequence _description="make drone fly to special point"> <!-- 定义起飞目标位置 --> <Script code=" takeoff_goal:='0.0;0.0;5.0' "/> <!-- 定义指点飞行的目标位置 --> <Script code=" point_goal:='0.0;1.5;1.0' "/> <Script code=" is_rc:=0 " /> <SubTree ID="takeoff" is_rc="{is_rc}" goal="{takeoff_goal}"/> <Action ID="ArriveGoal" tolerance_d="0.2" velocity="0;0;0" acc="0;0;0" ctrl_type="0" goal="{point_goal}"/> <SubTree ID="land" /> </Sequence> </BehaviorTree>
```

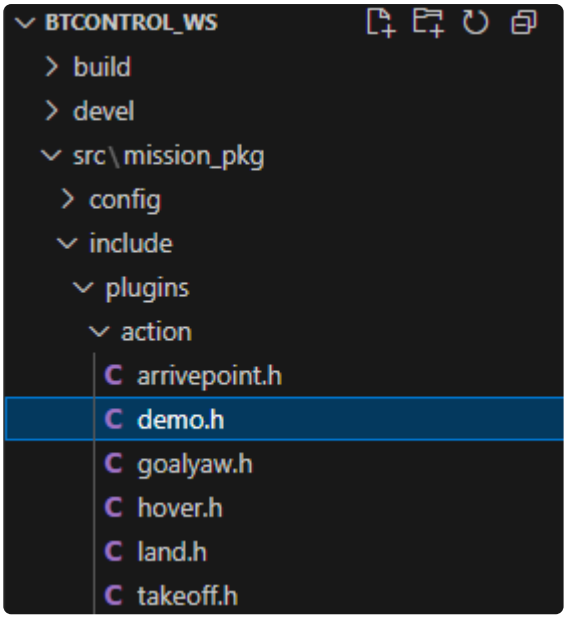
Step 4: 自定义行为树节点

创建一个demo行为树（这里沿用hover行为树的代码演示行为树的创建过程，它主要包括节点开始时、节点运行时和节点结束时三个动作，
实现无人机在 Offboard
模式下的悬停功能）并将demo作为子树添加进PointGoal行为树为例：

1. **定义节点类**

在BTcontrol_ws\src\mission_pkg\include\plugins\action下放置C++头文件demo.h；在BTcontrol_ws\src\mission_pkg\src\plugins\action下创建demo.cpp定义悬停节点的具体行为和逻辑，注册的节点名称为
"DEMO"：

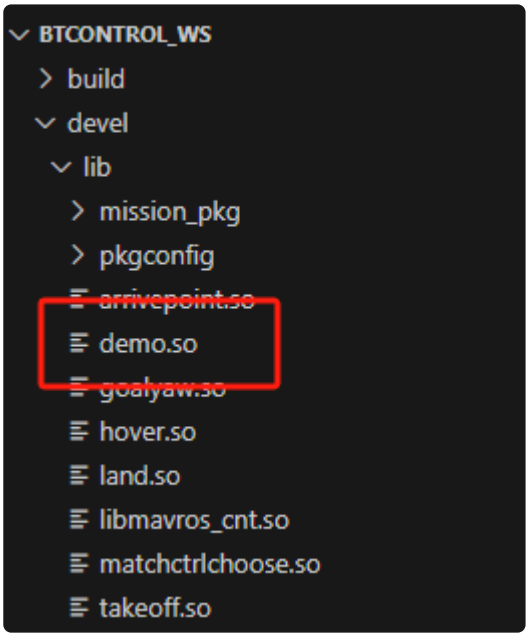





```
BT_REGISTER_NODES(factory) { factory.registerNodeType<Hover>("DEMO"); }
```

2. **将自定义节点加入BT工厂**

使用catkin_make编译通过后，自定义的行为树插件.so文件会保存在devel/lib下：



运行bt_ros.cpp时会通过下面的代码将自定义的行为树插件.so文件加入到BT工厂中：

```
116: for(size_t i = 0; i < lib_so_files.size(); ++i) 117: { 118: factory.registerFromPlugin(lib_so_files[i]); 119: }
```

3. **在XML中引用该节点**

在/BTcontrol_ws/src/mission_pkg/config/中创建demo行为树的XML配置文件demo.xml，定义行为树的ID为demo，动作节点的ID为DEMO：

```
<?xml version="1.0" encoding="UTF-8"?> <root BTCPP_format="4" main_tree_to_execute="demo"> <BehaviorTree ID="demo"> <Sequence _description="make drone fly to special piont"> <Action ID="DEMO" hover_is_end="{hover_is_end}" stop_hover="{stop_hover}" is_time_ctrl="{is_time_ctrl}"/> </Sequence> </BehaviorTree> </root>
```

如果将demo节点添加进PointGoal行为树，让无人机在到达目标点后（ArriveGoal节点后）悬停10s，需要先在/BTcontrol_ws/src/mission_pkg/config/point_goal.xml的基础上将demo.xml 文件中的行为树定义包含到当前的行为树文件中：

```
<include path="demo.xml" />
```

然后在<BehaviorTree ID="PointGoal">中ArriveGoal节点前添加悬停行为树子模块并设置变量的初始值：

```
<Script code=" hover_is_end:=true "/> <!--如果被设置为 true，悬停动作会提前终止。 --> <Script code=" stop_hover:=false "/> <!--控制悬停动作的时间行为 --> <Script code=" is_time_ctrl:=10.0 " />
```

```
<SubTree ID="demo" hover_is_end="{hover_is_end}" gstop_hover="{stop_hover}" is_time_ctrl="{is_time_ctrl}"/>
```

运行结果可以看到在指点飞行前多了10s的悬停：

```
[ INFO] [1751537652.635968700]: flag_time: 1751537642.770109
[ INFO] [1751537652.711910600]: recv stop hover command
[ INFO] [1751537652.712073400]: keep_time:10.000000
[ INFO] [1751537652.712231900]: current_time: 1751537652.712216
[ INFO] [1751537652.712395000]: flag_time: 1751537642.770109
[ INFO] [1751537652.789207100]: recv stop hover command
[ INFO] [1751537652.789363700]: keep_time:10.000000
[ INFO] [1751537652.789485400]: current_time: 1751537652.789468
[ INFO] [1751537652.789648500]: flag_time: 1751537642.770109
[ INFO] [1751537652.789766200]: use keep time is end
[1751537652.792]: DEMO                          RUNNING -> SUCCESS
[1751537652.792]: DEMO                          SUCCESS -> IDLE
[1751537652.792]: Sequence                      RUNNING -> SUCCESS
[1751537652.792]: Sequence                      SUCCESS -> IDLE
[1751537652.792]: demo                          RUNNING -> SUCCESS
[ INFO] [1751537652.792499400]: ========================goal: 0.000000,1.500000,1.000000
[1751537657.443]: ArriveGoal                    IDLE -> SUCCESS
[1751537657.443]: land                          IDLE -> RUNNING
[1751537657.443]: Sequence                      IDLE -> RUNNING
[1751537657.443]: SetBlackboard                 IDLE -> SUCCESS
[1751537657.443]: SetBlackboard                 IDLE -> SUCCESS
[ INFO] [1751537657.443515000]: to land ...
[ INFO] [1751537665.498915100]: change mode (AUTO.LAND)
```

Step 5: 行为树节点可视化（补充内容）

用下面的指令运行gen_xml.launch文件，在bt_ros.cpp文件中会通过 BehaviorTree.CPP 库
的 writeTreeNodesModelXML()
函数生成描述已注册行为树节点（XML节点）的元数据模型并保存在
BTcontrol_ws\src\mission_pkg\config\all_node.xml文件中。

roslaunch mission_pkg send_goal.launch

```
<root BTCPP_format="4">
    <TreeNodesModel>
        <Action ID="ArriveGoal">
            <input_port name="tolerance_d" type="double">less than the value,it arrvied</input_port>
            <input_port name="velocity" type="BT::Position3D">This is expect speed</input_port>
            <input_port name="acc" type="BT::Position3D">This is expect accelerate</input_port>
            <input_port name="ctrl_type" type="int">0:PosCtrl,1:Velocity,2:Acc,3:Planner</input_port>
            <input_port name="goal" type="BT::Position3D">drone fly to goal</input_port>
        </Action>
        <Action ID="DEMO">
            <input_port name="is_time_ctrl" type="double">keep hover time</input_port>
            <input_port name="hover_is_end" type="bool">if exit current node,be set ture</input_port>
            <input_port name="stop_hover" type="bool">emit hover event</input_port>
        </Action>
        <Action ID="GoalYaw">
            <input_port name="yaw_rate" type="BT::AnyTypeAllowed">use we want yaw_rate</input_port>
            <input_port name="is_set_point" type="BT::AnyTypeAllowed">if we want set point</input_port>
            <input_port name="point" type="BT::AnyTypeAllowed">set we need point,but the is_set_point is true</input_port>
            <input_port name="goal_yaw" type="BT::AnyTypeAllowed">we need to control the fcu yaw</input_port>
        </Action>
        <Action ID="Hover">
            <input_port name="is_time_ctrl" type="double">keep hover time</input_port>
            <input_port name="hover_is_end" type="bool">if exit current node,be set ture</input_port>
            <input_port name="stop_hover" type="bool">emit hover event</input_port>
        </Action>
        <Action ID="Land">
            <input_port name="speed_z" type="BT::AnyTypeAllowed">land speed</input_port>
            <input_port name="use_speed" type="BT::AnyTypeAllowed">Land ... You can use the speed to control, or use AUTO.LAND
        </Action>
        <Condition ID="MatchCtrlChoose">
            <input_port name="expected_value" type="int"/>
            <input_port name="config_param" type="int"/>
        </Condition>
        <Action ID="Takeoff">
            <input_port name="is_rc" type="BT::AnyTypeAllowed">is remote controller operator</input_port>
            <input_port name="goal" type="BT::Position3D">Takeoff position ... </input_port>
        </Action>
    </TreeNodesModel>
</root>
```

生成的 XML 可用于 Groot2（BehaviorTree
官方编辑器，下载地址https://www.behaviortree.dev/groot）加载节点元数据。

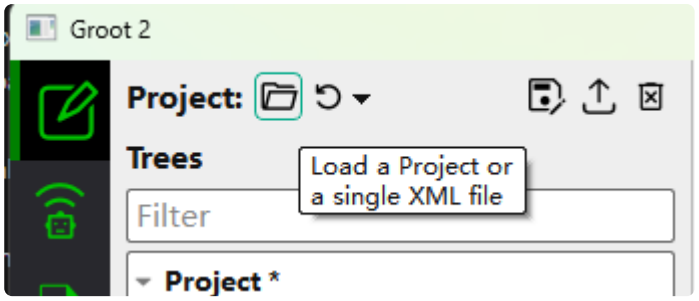
如果要可视化PointGoal行为树，需要把每个行为节点以及调用的子树的行为节点在文件的all_node.xml中对应的节点模型复制过来放在对应的位置。例如，takeoff子树的行为节点的节点模型放置在子树的XML文件下：

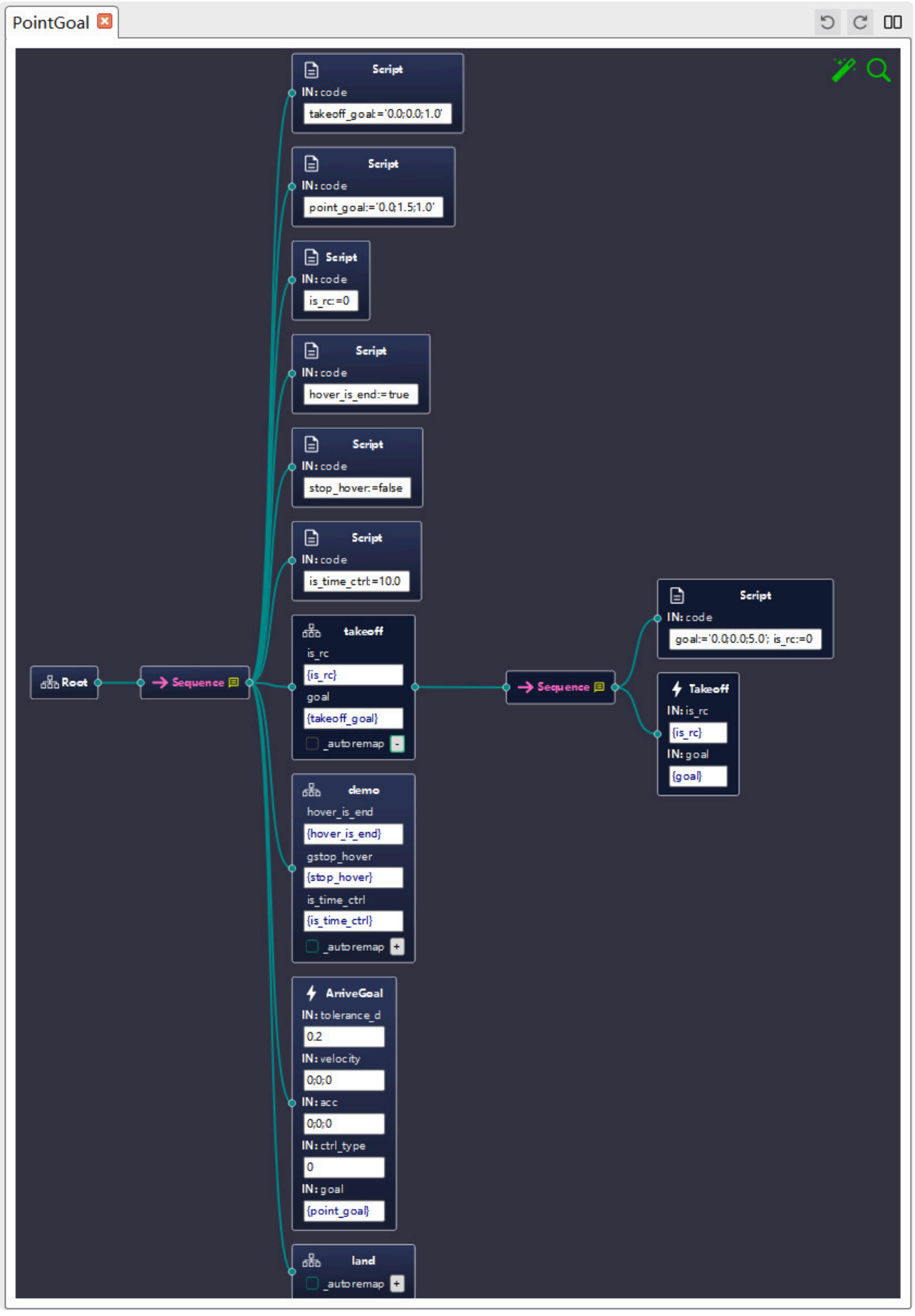
```xml
<?xml version="1.0" encoding="UTF-8"?>
<root BTCPP_format="4">
  <BehaviorTree ID="takeoff">
    <Sequence _description="make drone takeoff">
      <!-- 定义变量 -->
      <Script code=" goal:='0.0;0.0;5.0'; is_rc:=0 " />
      <!-- 调用 Takeoff 节点 -->
      <Action ID="Takeoff" is_rc="{is_rc}" goal="{goal}"/>
    </Sequence>

  </BehaviorTree>

  <!-- 定义节点模型 -->
  <TreeNodesModel>
    <Action ID="Takeoff">
    <input_port name="is_rc" type="int">is remote controller operator</input_port>
    <input_port name="goal" type="BT::Position3D">Takeoff position ...</input_port>
    </Action>
  </TreeNodesModel>
</root>
```

在PointGoal行为树XML文件下面放置子树节点模型以及行为节点的节点模型：

```xml
                hover_is_end="{hover_is_end}"
                gstop_hover="{stop_hover}"
                is_time_ctrl="{is_time_ctrl}"/>

      <Action ID="ArriveGoal" tolerance_d="0.2"
                velocity="0;0;0"
                acc="0;0;0"
                ctrl_type="0"
                goal="{point_goal}"/>
      <SubTree ID="land" />
    </Sequence>
  </BehaviorTree>

  <!-- Description of Node Models (used by Groot) -->
  <TreeNodesModel>
    <SubTree ID="takeoff" />
    <SubTree ID="land" />
    <Action ID="ArriveGoal">
      <input_port name="tolerance_d"
                  type="double">less than the value,it arrvied</input_port>
      <input_port name="velocity"
                  type="BT::Position3D">This is expect speed</input_port>
      <input_port name="acc"
                  type="BT::Position3D">This is expect accelerate</input_port>
      <input_port name="ctrl_type"
                  type="int">0:PosCtrl,1:Velocity,2:Acc,3:Planner</input_port>
      <input_port name="goal"
                  type="BT::Position3D">drone fly to goal</input_port>
    </Action>
    <SubTree ID="demo" />
  </TreeNodesModel>

</root>
```

然后用Groot2打开PointGoal行为树XML文件point_goal.xml，可以看到每个节点的执行顺序和输入：


Groot 2
Project:
Trees
Load a Project or
a single XML file
Filter
Project *


PointGoal
Root
Sequence
Script
IN: code
takeoff_goal:='0.0;0.0;1.0'
Script
IN: code
point_goal:='0.0;1.5;1.0'
Script
IN: code
is_rc:=0
Script
IN: code
hover_is_end:=true
Script
IN: code
stop_hover:=false
Script
IN: code
is_time_ctrl:=10.0
takeoff
is_rc
{is_rc}
goal
{takeoff_goal}
_autoremap
demo
hover_is_end
{hover_is_end}
gstop_hover
{stop_hover}
is_time_ctrl
{is_time_ctrl}
_autoremap
ArriveGoal
IN: tolerance_d
0.2
IN: velocity
0;0;0
IN: acc
0;0;0
IN: ctrl_type
0
IN: goal
{point_goal}
land
_autoremap
Sequence
Script
IN: code
goal:='0.0;0.0;5.0'; is_rc:=0
Takeoff
IN: is_rc
{is_rc}
IN: goal
{goal}

6.参考资料

无

7.常见问题

Q：Could NOT find behaviortree_cpp (missing: behaviortree_cpp_DIR)

A：sudo apt-get install ros-noetic-behaviortree-cpp