基于强化学习的水面WRT对抗系统实验报告

# 验证目标

根据设定的训练场景，产生并收集规范化的数据集。利用数据集进行算法训练生成决策模型，然后对决策模型进行测试和评估。

# 验证过程

## 2.1 验证过程流程图

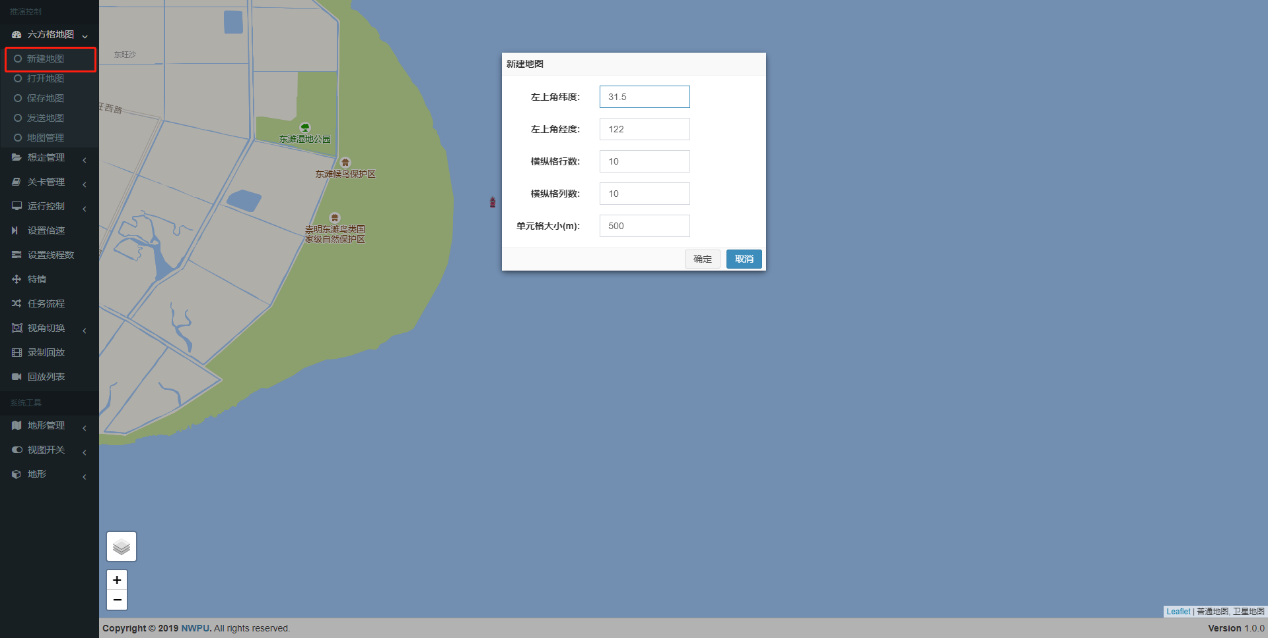


## 2.2 训练和验证中的关卡设置

### 2.2.1 绘制地图

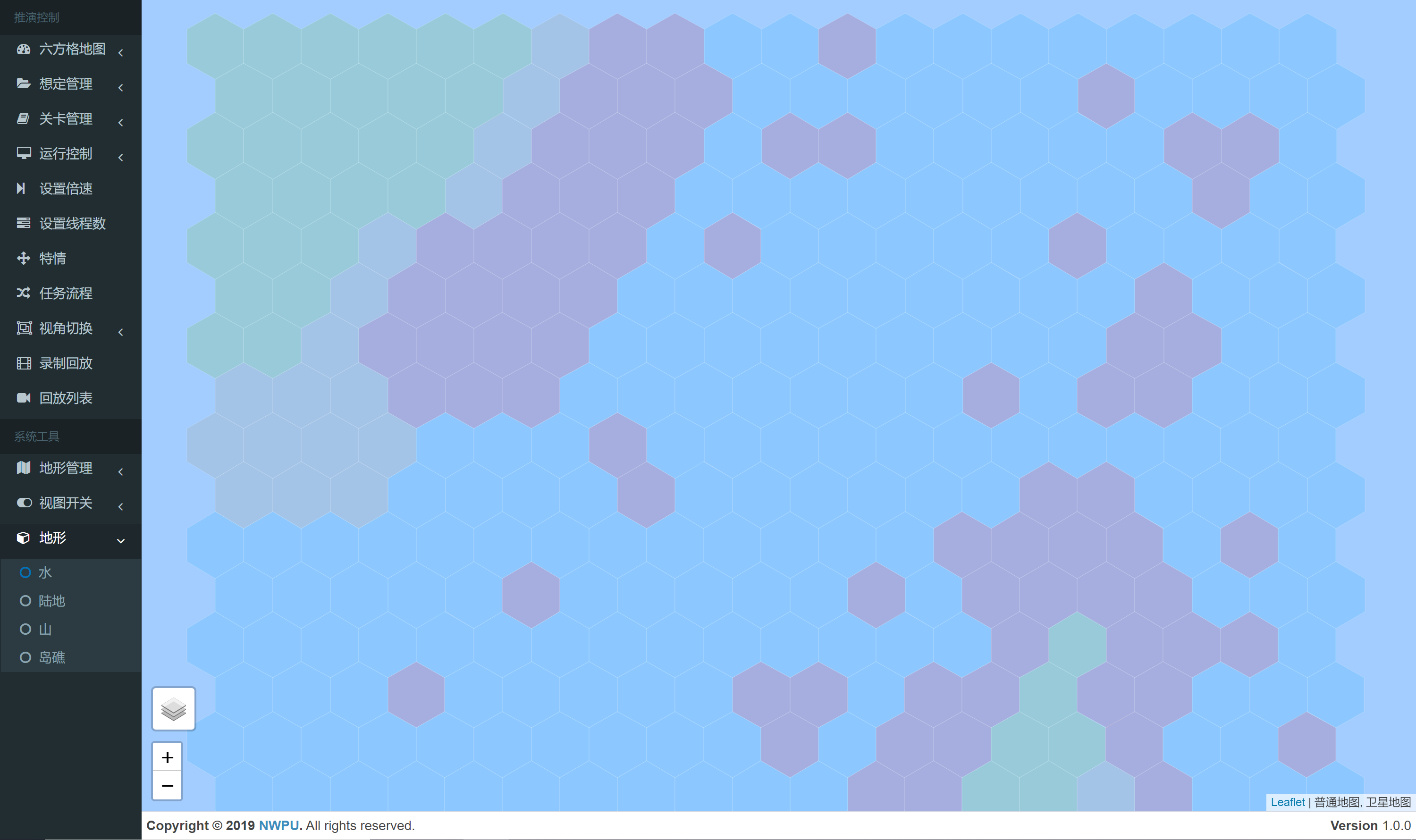
选择“推演控制”中的【六方格地图】功能，点击【新建地图】，在“新建地图”页面设置地图的经纬度、单元格的行列数和大小，确定后页面显示所设置的地图样式。

1）设置地图大小和方位

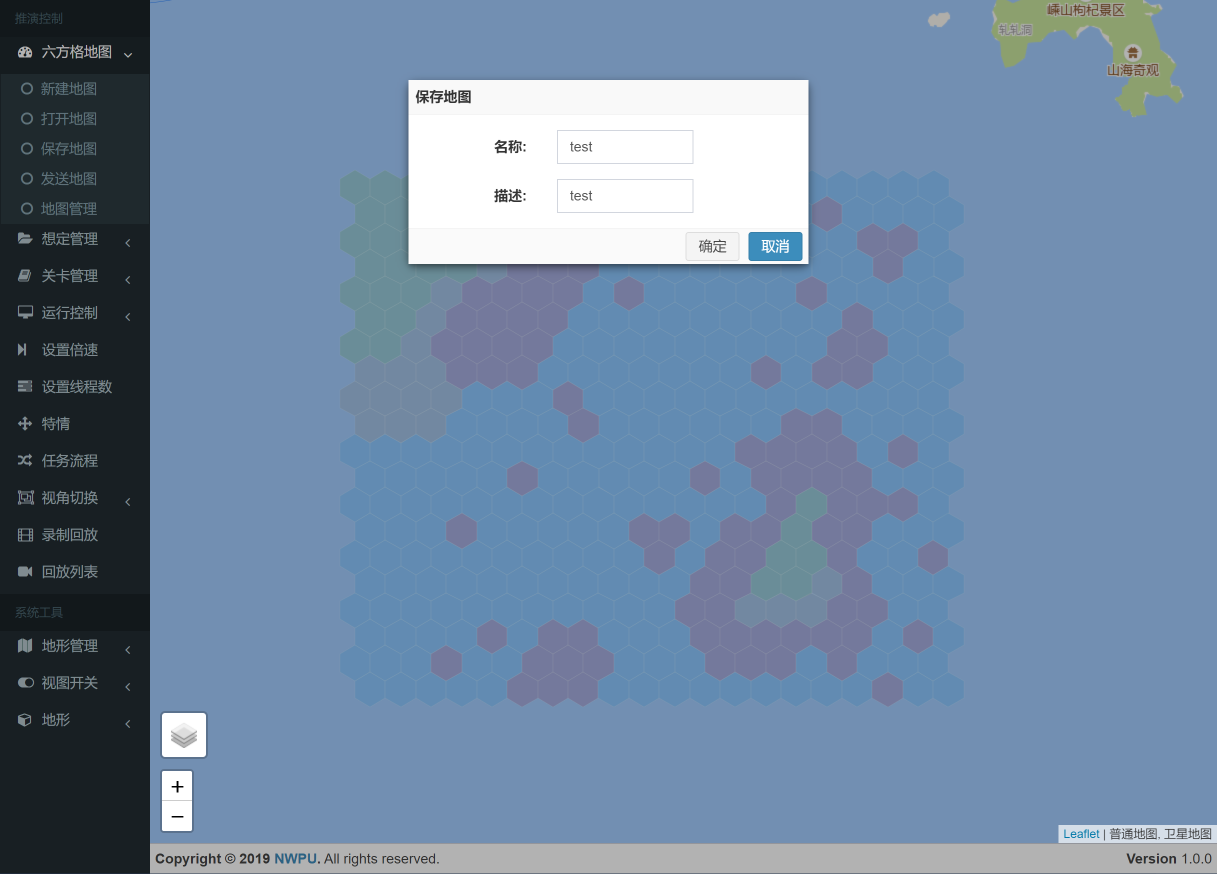




2）编辑各个六方格的地形



选择【保存地图】功能，输入待保存地图的名称和描述，点击确定后完成地图的保存功能。



新增的地图保存成功后，在【地图管理】中查看已添加的地图，并且能够对已有地图进行一些操作，包括修改和删除。

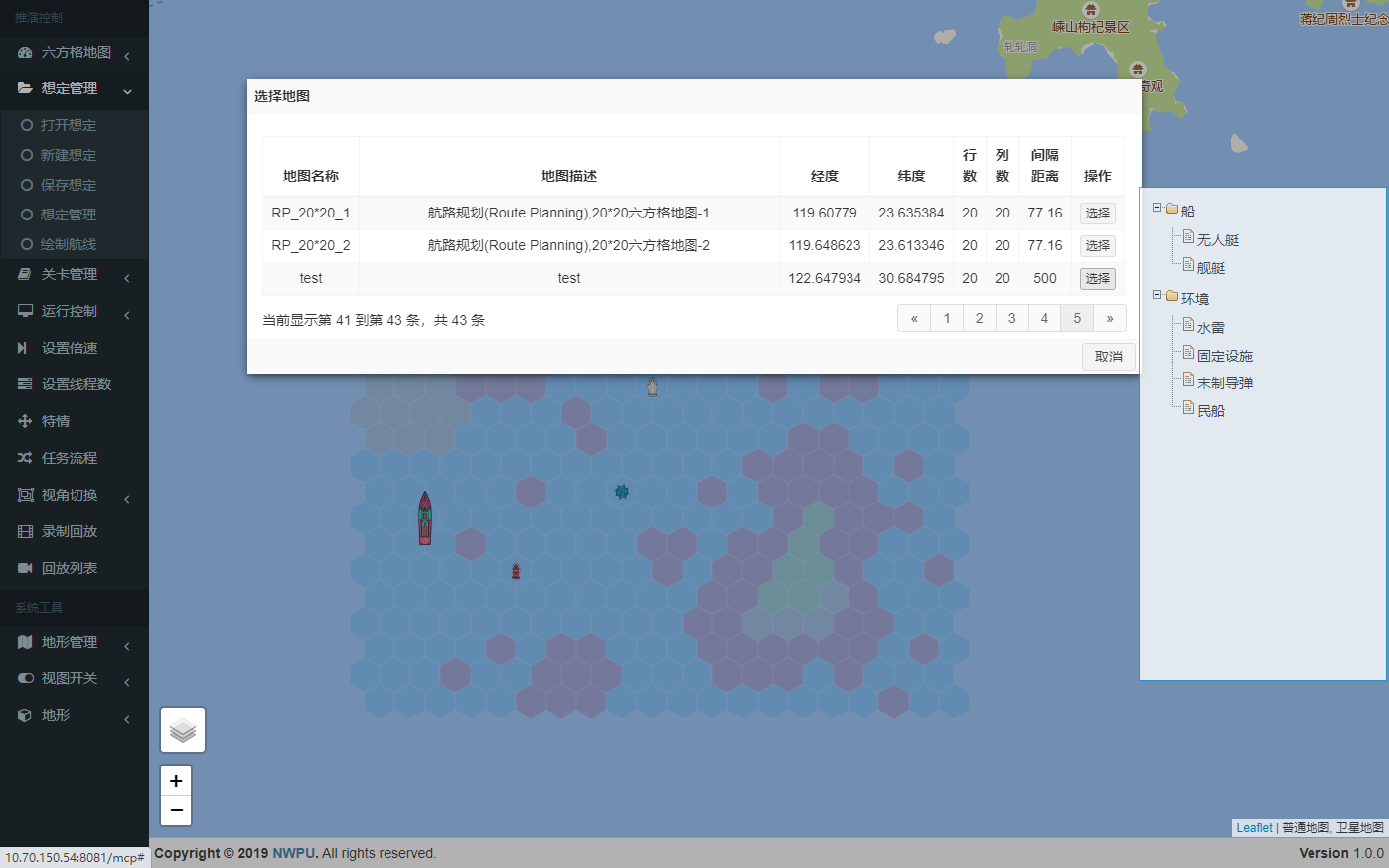


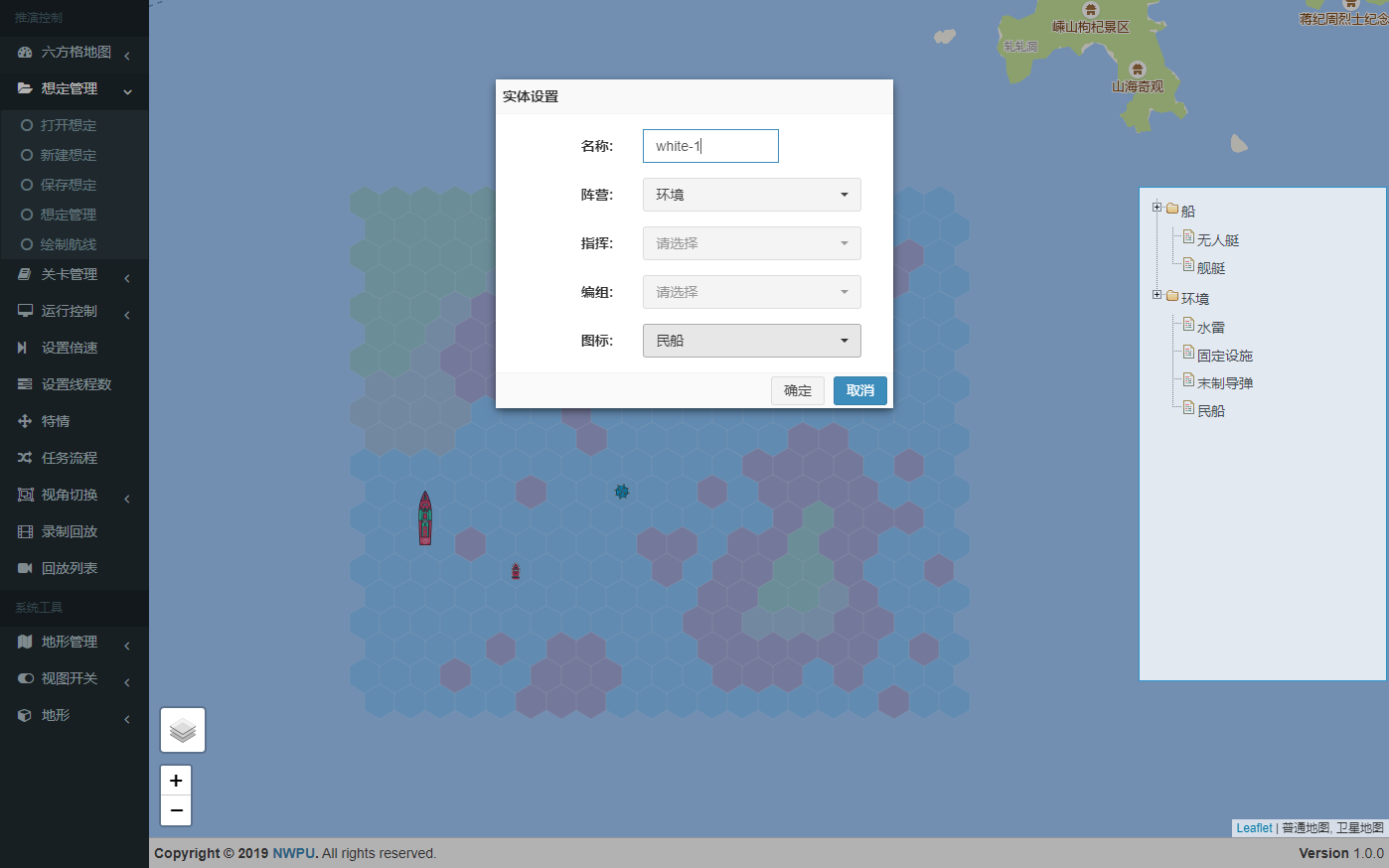
选择【打开地图】功能，选中其中一个地图文件，确定后在页面显示该地图的信息。

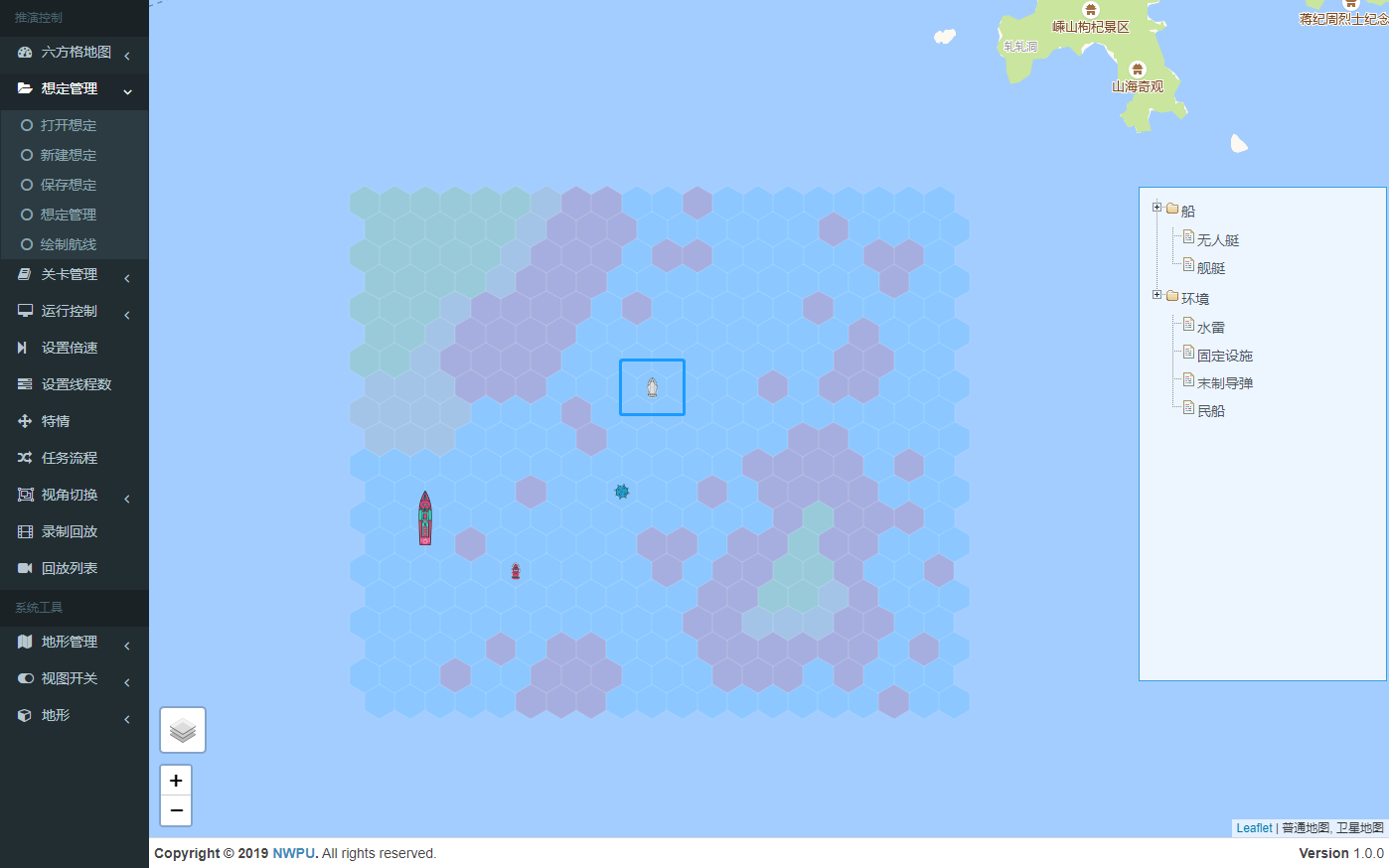


### 2.2.2 想定构建

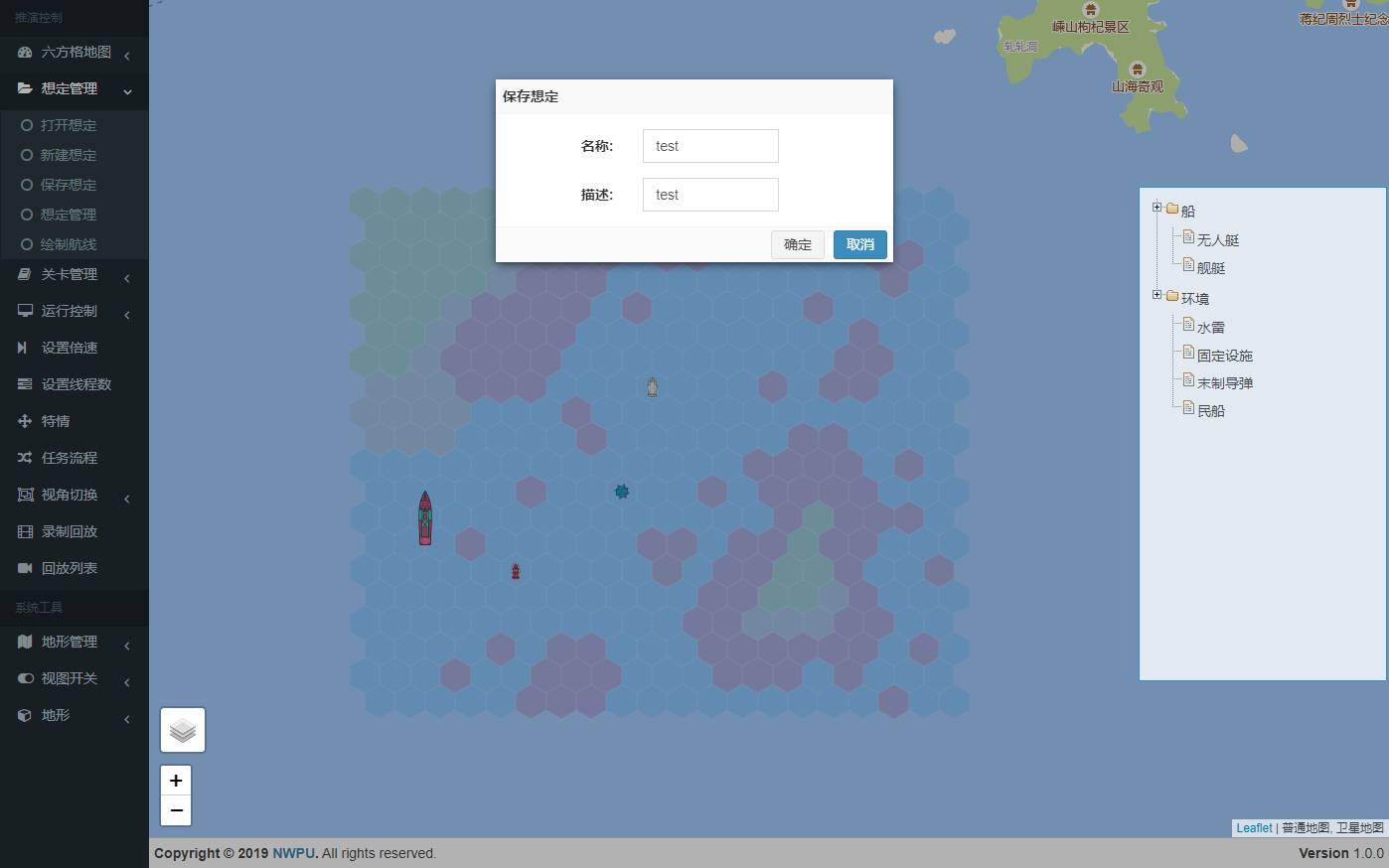
在WRT对抗系统监控平台（MCP）界面左边导航栏选择“推演控制”中的【想定管理】功能，点击【新建想定】，在“选择地图”页面选择其中一个地图，页面显示选中的地图样式以及可供选择的实体模型。选中其中一个实体模型并拖动到地图上的某一个位置，然后再“实体设置”页面设置该实体模型的具体信息，包括名称、阵营、指挥、编组和图标。重复该步骤，逐步完成想定中的实体配置。



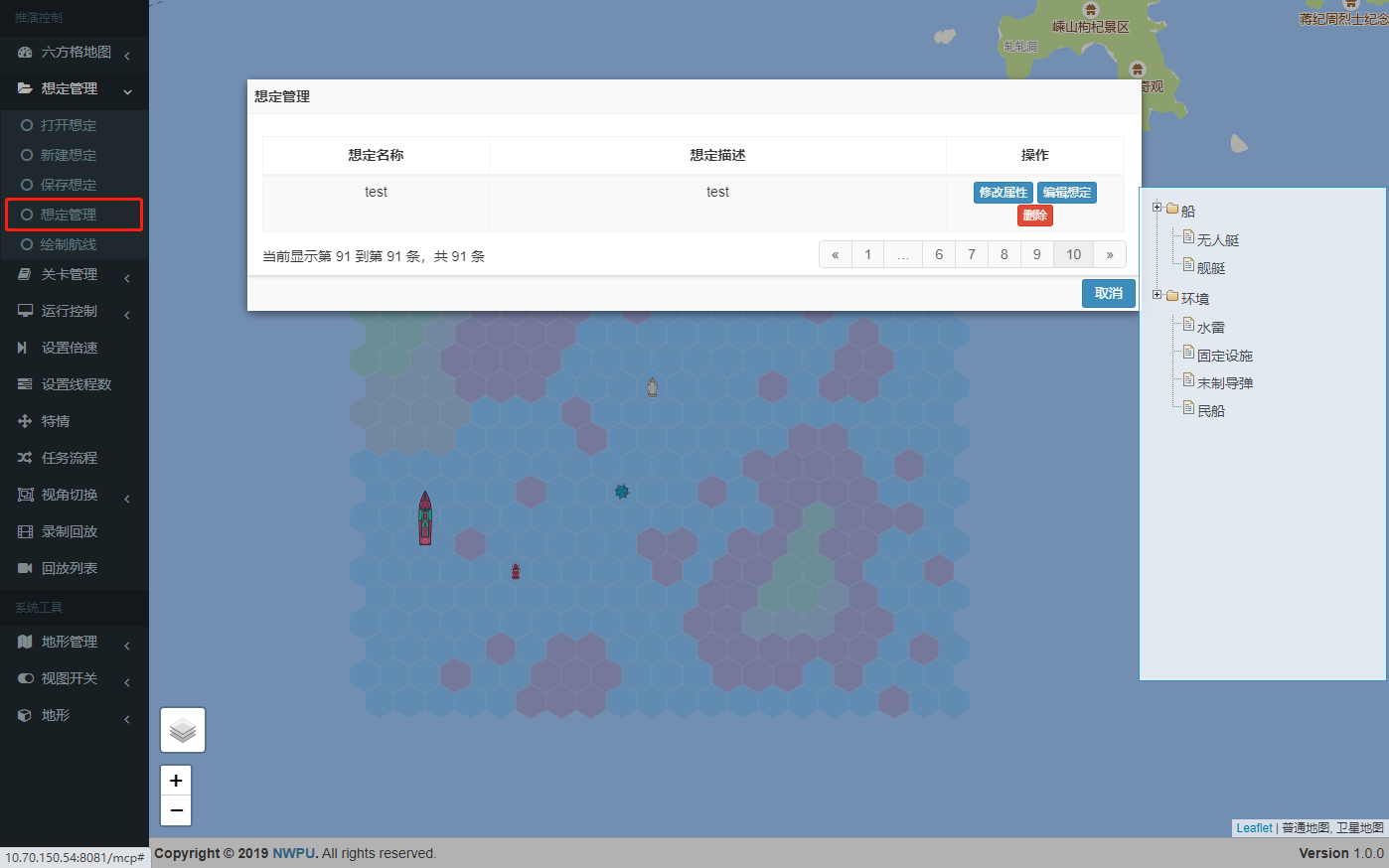




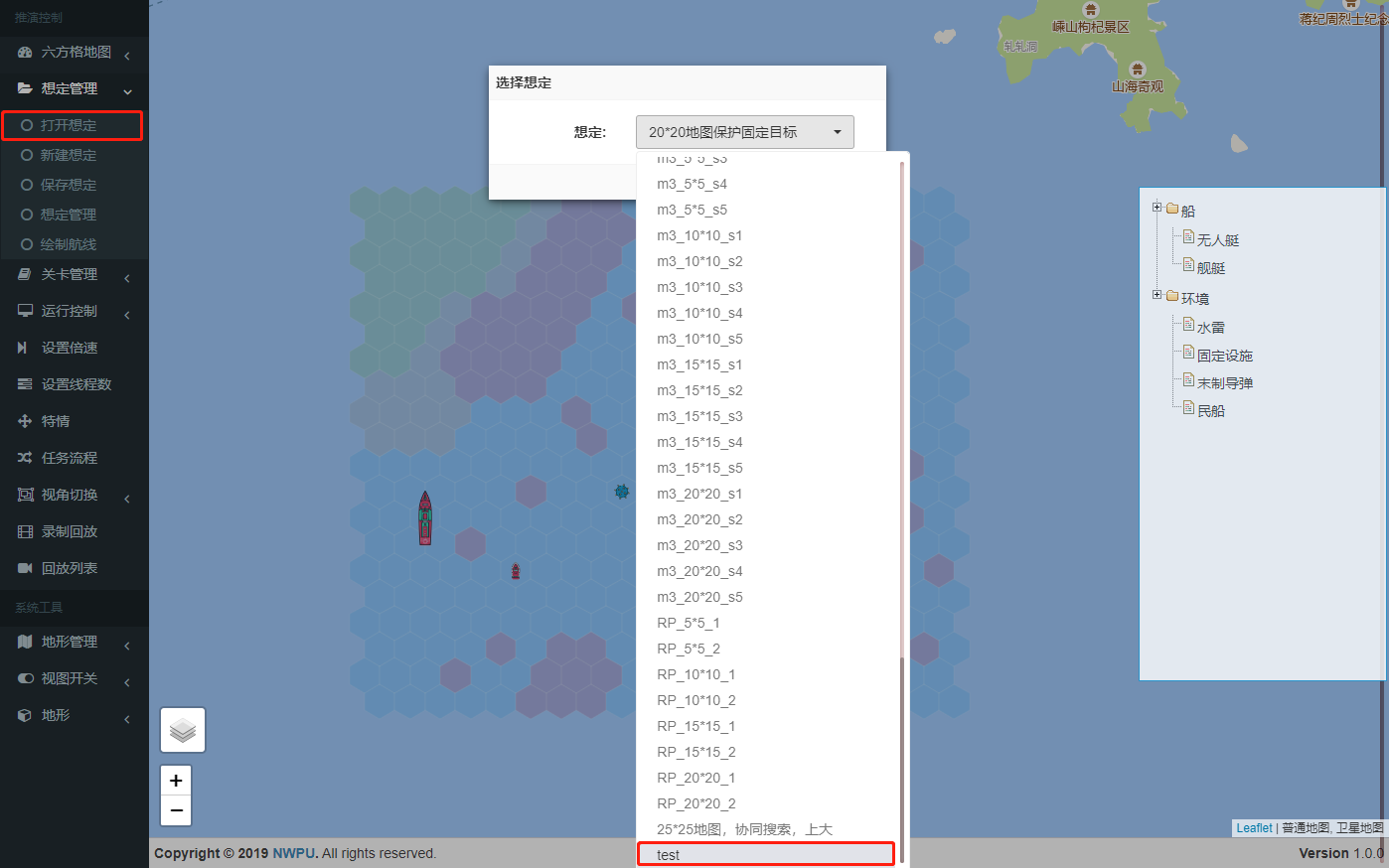
选择【保存想定】功能，输入待保存想定的名称和描述，点击确定后完成想定的保存功能。



新增的想定保存成功后，在【想定管理】中查看已添加的想定，并且能够对已有想定进行一些操作，包括修改属性，编辑想定和删除。



选择【打开想定】功能，选中其中一个想定文件，确定后在页面显示该想定的全部信息。



### 2.2.3 设置评分

在训练过程中按照规定的评分规则对训练的实体对象进行打分，评分规则结构定义为：

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 描述 |
| id | int64 | 评分规则的唯一标识 |
| sceId | int64 | 想定的唯一标识 |
| criterion | string | 规则类型名称 |
| operator | string | 操作符，加或减 |
| reward | double | 奖励值 |
| faction | string | 阵营，red红方，blue蓝方，all所有阵营 |
| entities | string | 指定的实体列表 |
| operand1 | string | 操作数1 |
| operand2 | string | 操作数2 |
| desc | string | 评分规则描述 |

评分规则可结合WRT所处不同的状态指定WRT获取的得分，其中状态包括：

* + 移动，Moving，表示实体处于移动状态下每步的得分；
  + 停止，Stop，表示实体处于停止状态下每步的得分；
  + 搁浅，RunAground，表示实体位于非水域比如陆地、岛礁状态下每步的得分；
  + 碰撞，Collide，表示实体与其他实体发生碰撞时的得分；
  + 命中目标，HitTarget，表示实体开火命中一次目标时的得分；
  + 未命中目标，HitMiss，表示实体开火后未命中目标时的得分；
  + 被命中，BeHit，表示实体被命中时的得分。

### 2.2.4 设置终止条件

对抗场景需要设定终止规则条件，终止规则的定义被描述为结构：

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 描述 |
| id | int64 | 终止规则的唯一标识 |
| sceId | int64 | 想定的唯一标识 |
| criterion | string | 终止规则类型名称 |
| operator | string | 操作符，加或减 |
| operand1 | string | 操作数1 |
| operand2 | string | 操作数2 |
| priority | int | 优先级 |
| faction | string | 阵营，red红方，blue蓝方，all所有阵营 |
| entities | string | 指定的实体列表 |
| associate | string | 关联规则 |
| type | string | 终止条件重复性类型 |
| score | string | 终止打分 |
| desc | string | 评分规则描述 |

为支持规则组合、规则的重复性和时间延续性判断，引入规则关联性描述和重复性类型定义，分别对应于associate和type字段，其中associate可定义为：

* + Final 最终决策。本条规则可最终决定回合终止。
  + SatisfyAll 所有同类规则全满足，则回合终止。
  + Satisfy: [a, b] 所有同类规则满足 a<x<b，则回合终止。
  + Satisfy: a 所有同类规则满足 x >= a 条，则回合终止。
  + Associate:[a, b, c] 与a, b, c三条规则同时满足，则回合终止。
  + Additional：其他规则的附加规则。与其他规则协作完成。

type可定义为：

* + OneShootKiller 一次命中则规则满足且终止，终止后，该规则不再重复参加检查。
  + Repeated: [a] 重复a次后，则整体条件满足。
  + KeepOn: [a] 持续a时间，则条件满足。
  + RealTime: 实时的条件。每次都检查结果，实时反馈结果。

终止规则可结合当前整个场景所处的状态设定是否终止，并可指定阵营的得分，其中状态包括：

* + 抵达目的地，ArriveAtTarget，表示实体抵达到目的地指定半径的范围圈内时对局终止，目的地可以是指定实体或位置；
  + 存活数量，AliveCount，表示某个阵营当前实体存活的数量大于、小于或等于指定值时对局终止。

## 2.3 训练方法

### 2.3.1 启动docker训练算法模型

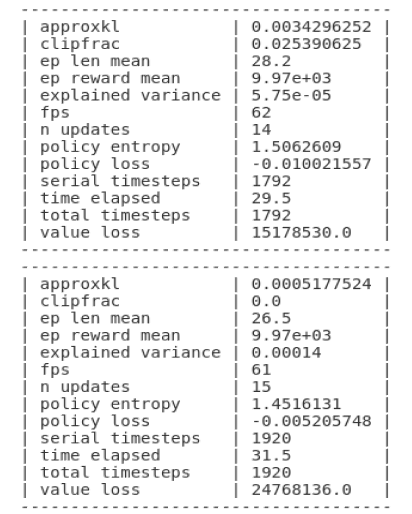
|  |
| --- |
| #docker命令示例  docker run -p 81:80 -d -v /home/test/workspace/docker/Local/workdir:/workdir -v /home/test/workspace/docker/Local/modeldir:/modeldir -v /home/test/workspace/docker/Local/logdir:/logdir -v /home/test/workspace/docker/Local/datadir:/datadir --name wrt\_nwpu wrt\_nwpu:v1.0.7 /bin/bash /home/run.sh train.py -u http://127.0.0.1:9999/common/queryScore -i 3 -c 1109 -a PPO -v SimuWRTEnv-v1 -m wrt\_sce\_3\_ppo2\_3 -d wrt\_sce\_3\_ppo2\_3\_tensorboard -s 20000 -t 100 |

Docker启动参数中bash脚本附加参数描述：

|  |
| --- |
| train.py -u <Server URL> -i <Third ID> -c <Checkpoint ID> -a <Algorithm type> -v <Env Name> -m <Model Name> -d <Log Dir> -s <Steps> -t <Times> |

|  |  |
| --- | --- |
| 参数 | 说明 |
| train.py | 训练算法主程序文件 |
| -u <Server URL> | 训练结束后上报训练结果数据的地址 |
| -i <Third ID> | 本次训练任务唯一标识 |
| -c <Checkpoint ID> | 关卡唯一标识，从数据库中选择获取 |
| -a <Algorithm type> | 模型算法类型，例如PPO，DQN |
| -v <Env Name> | Gym环境名称，从数据库中选择获取 |
| -m <Model Name> | 指定算法模型的名称，用户自定义 |
| -d <Log Dir> | 指定日志文件夹的名称，用户自定义 |
| -s <Steps> | 指定单次算法学习训练的最大步数，用户自定义 |
| -t <Times> | 指定算法学习训练的次数，用户自定义 |

在映射的logdir路径下会生成训练日志，通过tail -f train.log可查看训练过程中每次更新模型参数时，模型的平均得分、平均步数等评估数据：

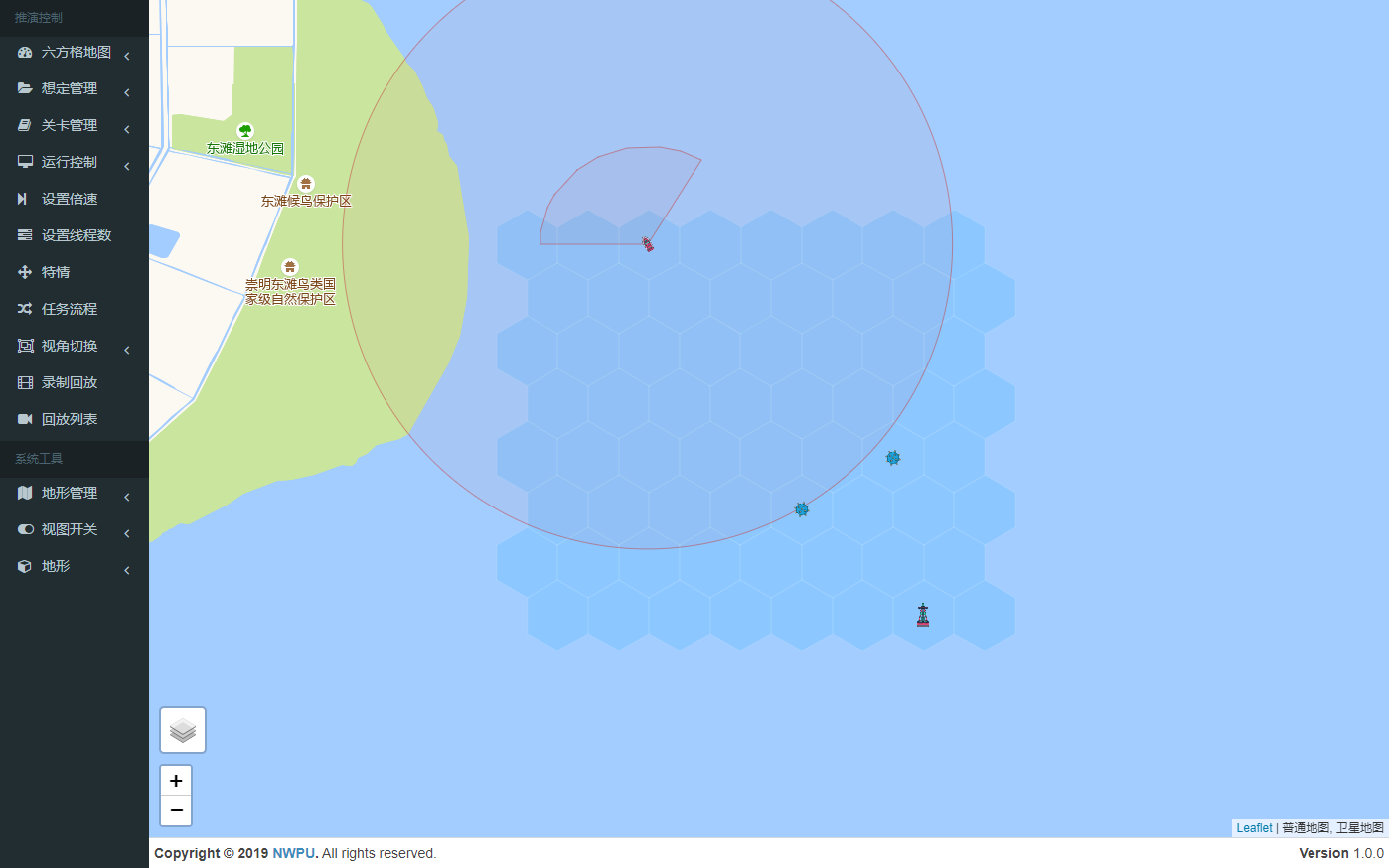


参数说明：

|  |  |
| --- | --- |
| 参数名称 | 描述 |
| approxkl | KL散度的近似 |
| clipfrac | 裁剪因子 |
| ep\_len\_mean | 每个回合平均长度 |
| ep\_reward\_mean | 每个回合平均得分 |
| explained\_variance | 降维后的各主成分的方差值 |
| fps | 每秒训练的步数 |
| n\_updates | 第n次更新神经网络 |
| policy\_entropy | 策略的熵 |
| policy\_loss | 策略损失函数值 |
| serial\_timesteps | 目前顺序的步数 |
| time\_elapsed | 花费时间 |
| total\_timesteps | 目前训练的总步数 |
| value\_loss | 值损失函数值 |

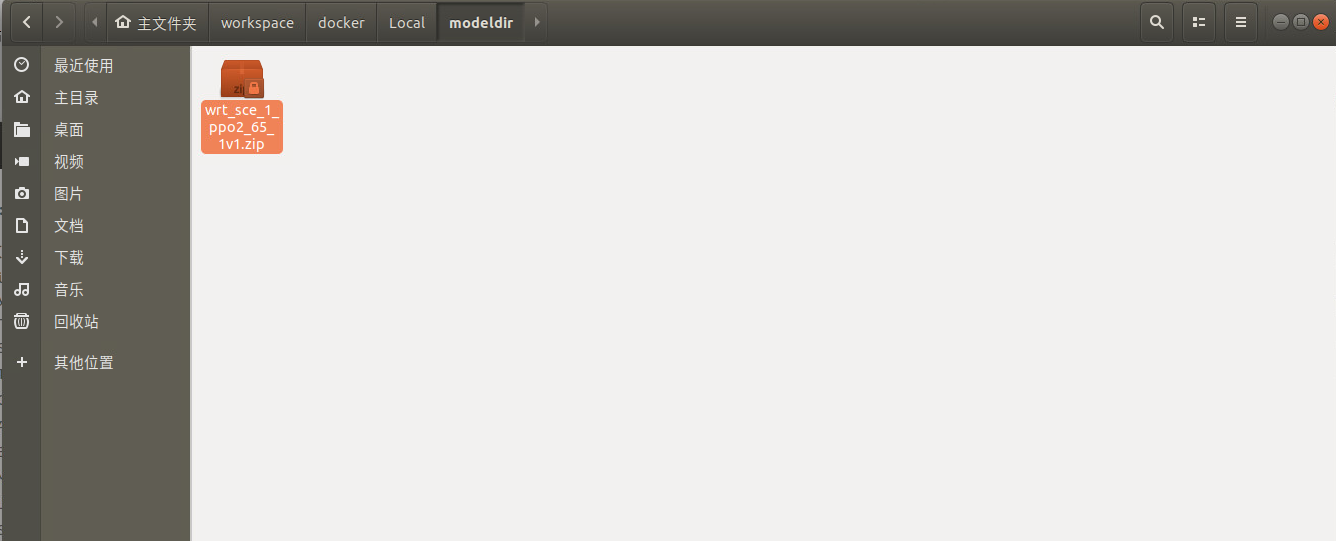
### 2.3.2 MCP监控训练过程

在MCP系统中选择“推演控制”中的【运行控制】功能，执行【智能模型训练】功能。



### 2.3.3 生成模型

训练结束后会在指定的模型生成目录下生成模型文件。



## 2.4 验证方法

### 2.4.1启动docker验证算法模型

|  |
| --- |
| #docker命令示例  docker run -p 81:80 -d -v /home/test/workspace/docker/Local/workdir:/workdir -v /home/test/workspace/docker/Local/modeldir:/modeldir -v /home/test/workspace/docker/Local/logdir:/logdir -v /home/test/workspace/docker/Local/datadir:/datadir --name wrt\_nwpu wrt\_nwpu:v1.0.7 /bin/bash /home/run.sh play.py -u 1 -i 1 -c 1109 -a PPO -v SimuWRTEnv-v8 -m wrt\_sce\_1\_ppo2\_65\_1v1 -t 4 |

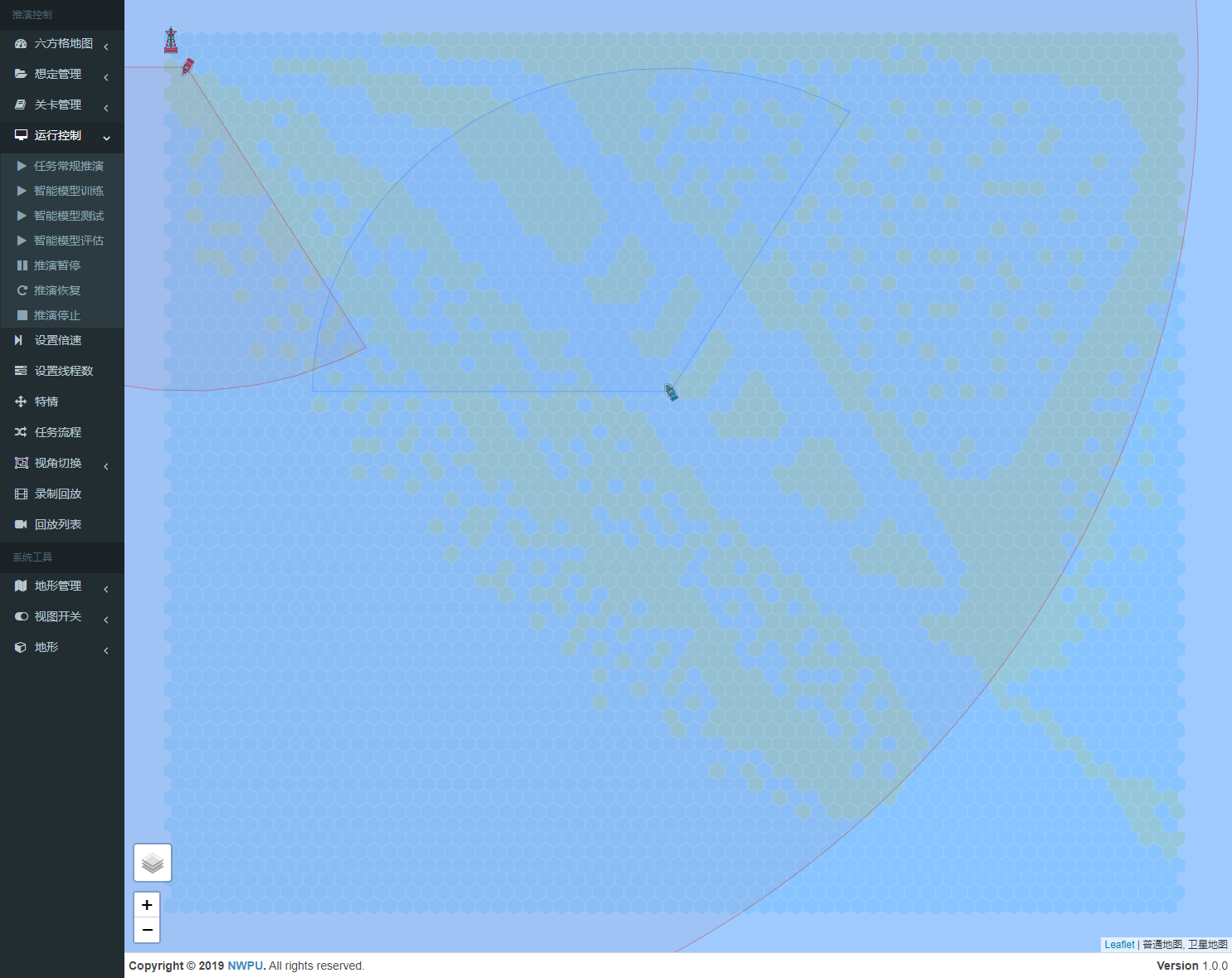
Docker启动参数中bash脚本附加参数描述：

|  |
| --- |
| play.py -u <User ID> -i <Third ID> -c <Checkpoint ID> -a <Algorithm type> -v <Env Name> -m <Model Name> -t <Times> |

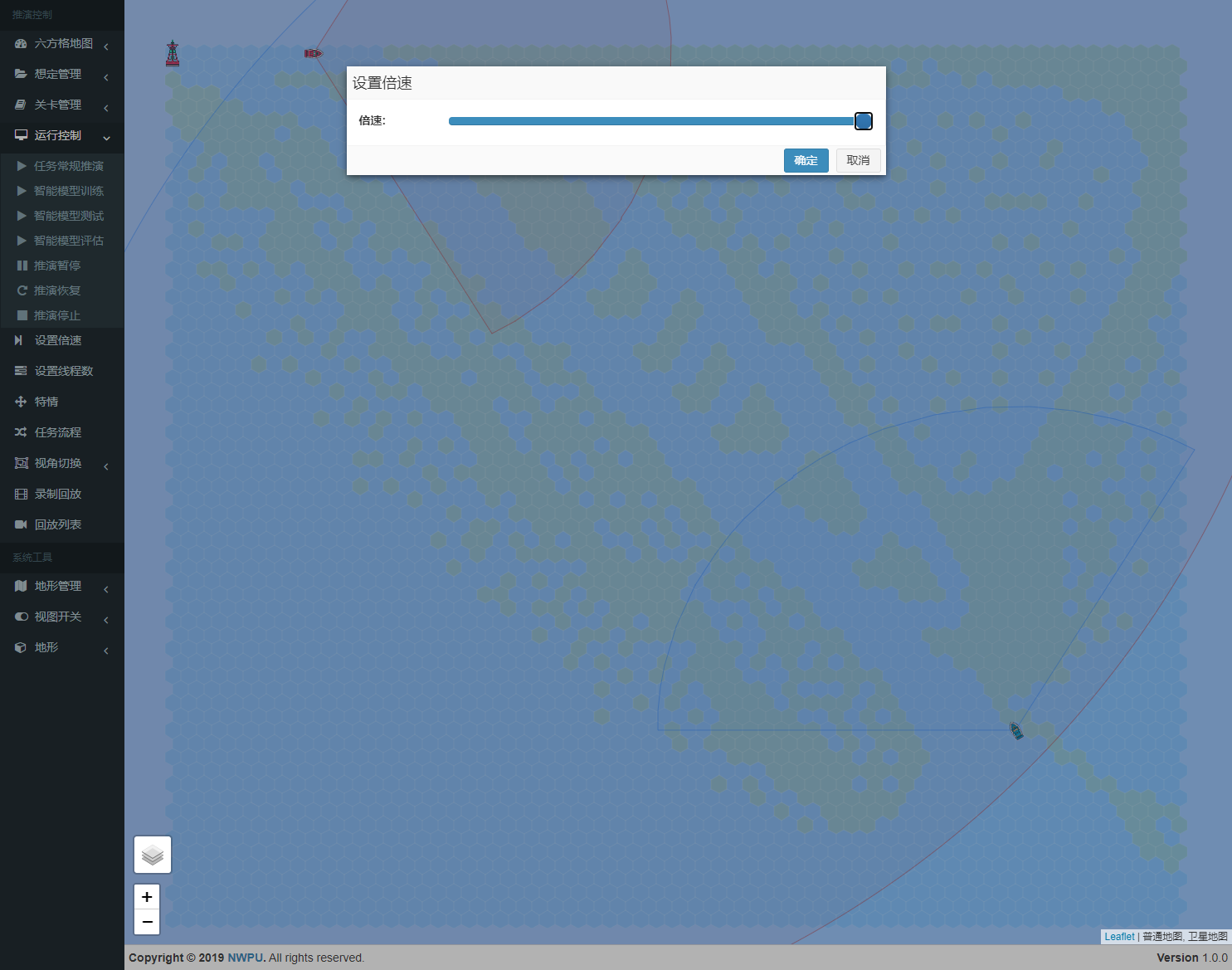
|  |  |
| --- | --- |
| 参数 | 说明 |
| play.py | 算法主程序文件 |
| -u <User ID> | 用户唯一标识 |
| -i <Third ID> | 本次训练任务唯一标识 |
| -c <Checkpoint ID> | 关卡唯一标识，从数据库中选择获取 |
| -a <Algorithm type> | 模型算法类型，例如PPO，DQN |
| -v <Env Name> | Gym环境名称，从数据库中选择获取 |
| -m <Model Name> | 指定算法模型的名称，用户自定义 |
| -t <Times> | 指定算法学习训练的次数，用户自定义 |

### 2.4.2 MCP监控模型验证过程

在MCP系统中选择“推演控制”中的【运行控制】功能，执行【智能模型测试】功能。



如果场景设定中实体模型的仿真周期很长，比如5秒，可以在MCP界面上【设置倍数】加速验证过程。

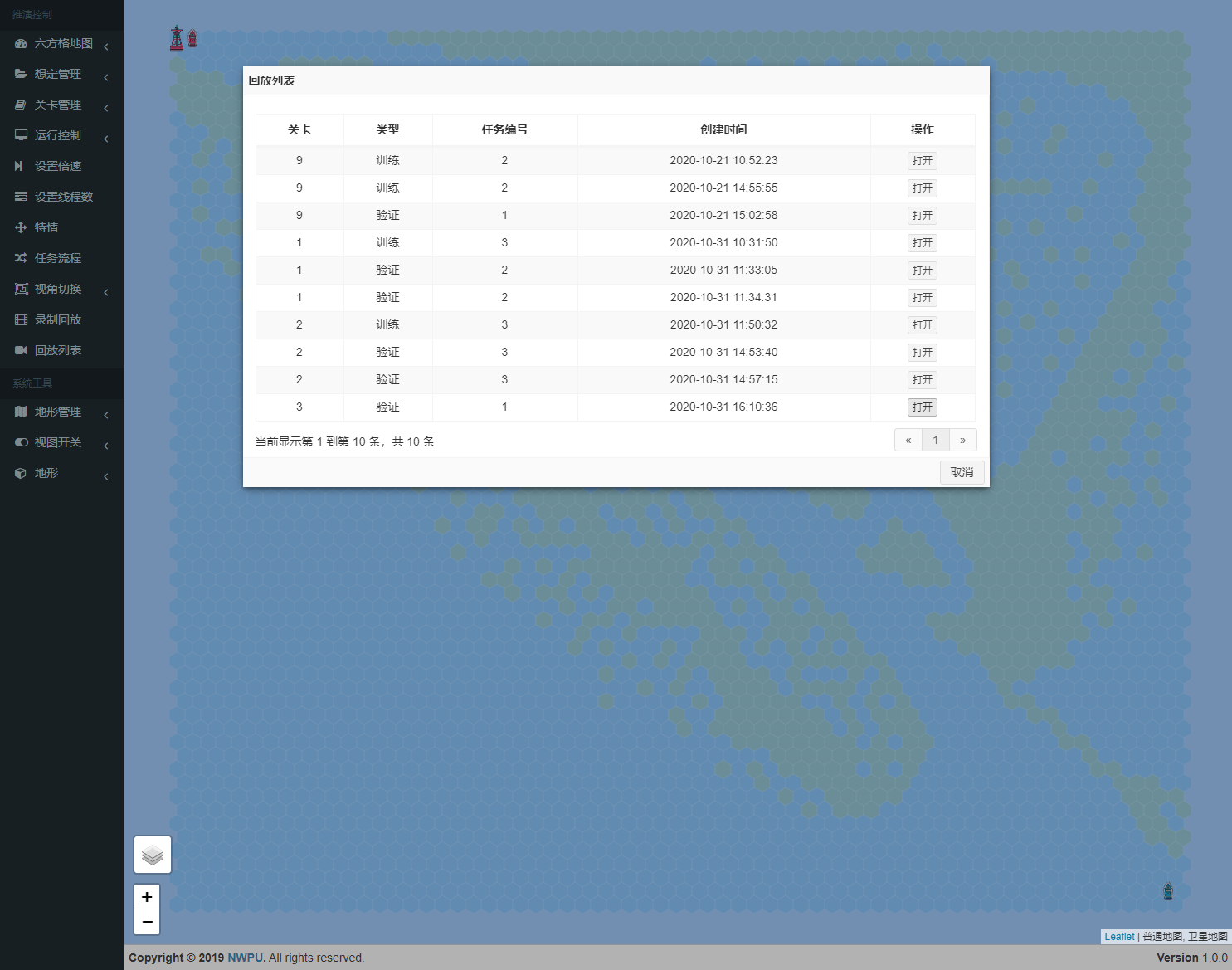


## 2.5 验证结果查看

对经过训练后的模型进行验证时，产生一系列验证结果数据，这些评估数据的含义和示例如下表所示。

|  |  |
| --- | --- |
| 表字段含义 | 示例 |
| 记录标识(id) | 1 |
| 用户标识(userId) | 1 |
| 想定标识(sceId) | 0 |
| 地图标识(mapId) | 0 |
| 关卡类型(cpType) | 0 |
| 关卡测试类型(cpInstType) | 0 |
| 关卡标识(cpId) | 1109 |
| 关卡名称(cpName) | 1109 |
| 模型标识(modelId) | 0 |
| 训练次数(episodeTime) | 4 |
| 胜率(winProb) | 0.25 |
| 平均步数(averSteps) | 85 |
| 平均得分(averReward) | -5000.75 |
| 创建时间(createTime) | 1603547505 |
| 完成时间(completeTime) | 1603547505 |

并可以通过监控平台【回访列表】查看曾经算法模型验证的记录，点击打开之后会回放算法模型验证过程中的决策信息。



# 验证环境配置

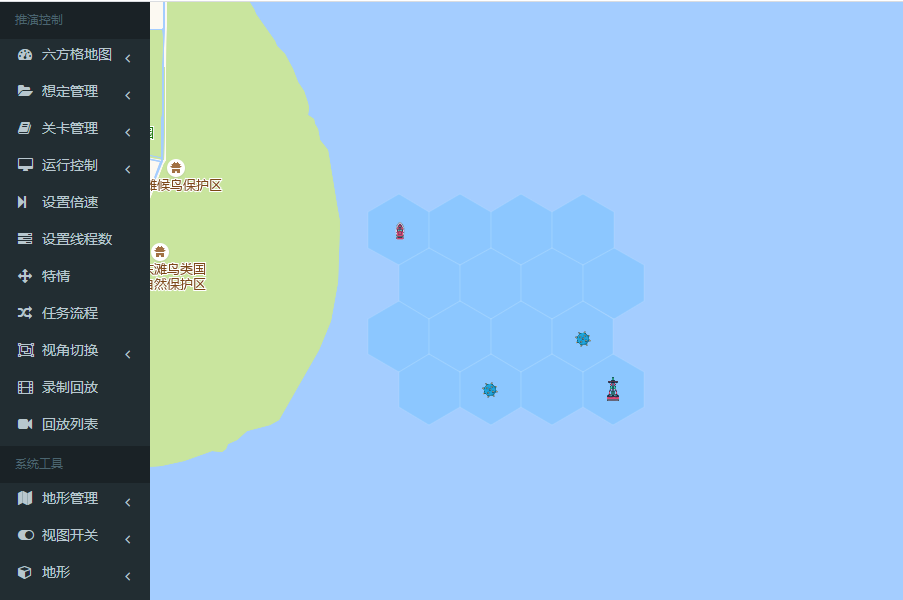
|  |  |  |
| --- | --- | --- |
| 软硬件名称 | 配置参数 | 数量 |
| 主机 | 操作系统版本：Ubuntu 18.04  处理器：Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz  系统类型： x86\_64  内存：8GB | 1 |

# 具体对抗场景实验

## 4.1 挑战-航路规划-关卡1

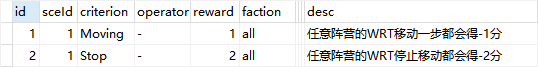
该关卡为挑战模式，类型为4\*4地图侦察探雷，场景的定义为：

* 想定和地图分别为：



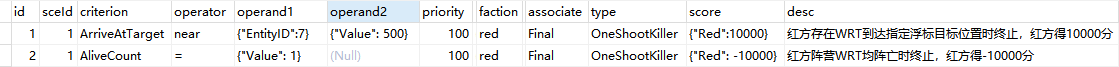
说明：场景中存在一艘红方无人艇、两个水雷和一个红方浮标，红方艇驶入到雷的方格中时雷会爆炸炸毁红方艇，红方浮标为目的地。

* 评分规则为：



说明：红方艇移动一步扣除1分，如果停下的话会扣除2分，促使红方艇找到最短路径。

* 终止规则为：

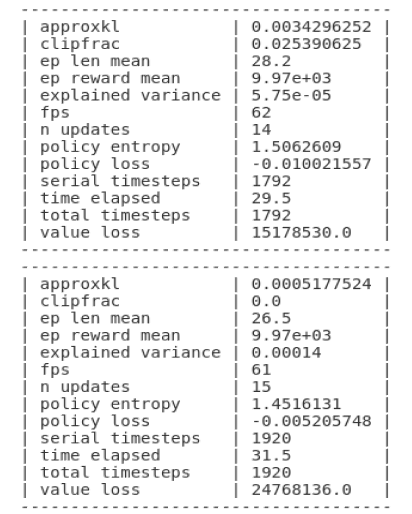


说明：当红方艇被炸毁时，对局结束并扣除10000分，当红方艇驶入到目的地时，对局结束并获得10000分奖励。

1. 启动docker训练模型；

|  |
| --- |
| train.py -u http://127.0.0.1:9999/common/queryScore -i 2 -c 1 -a PPO -v SimuWRTEnv-v4 -m wrt\_sce\_3\_ppo2\_4 -d wrt\_sce\_3\_ppo2\_4\_tensorboard -s 2000 -t 1 |

说明：尝试训练2000步。



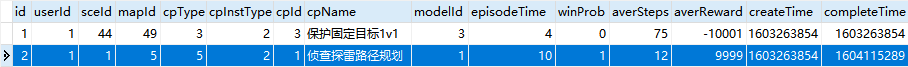
观察训练日志输出，最终回合的平均得分为9.97e+03，说明艇成功到达了目的地。

1. 启动docker验证模型；

|  |
| --- |
| play.py -u 1 -i 2 -c 1 -a PPO -v SimuWRTEnv-v4 -m wrt\_sce\_3\_ppo2\_4 -t 10 |

说明：尝试验证算法模型10个回合。

1. 生成验证结果数据。

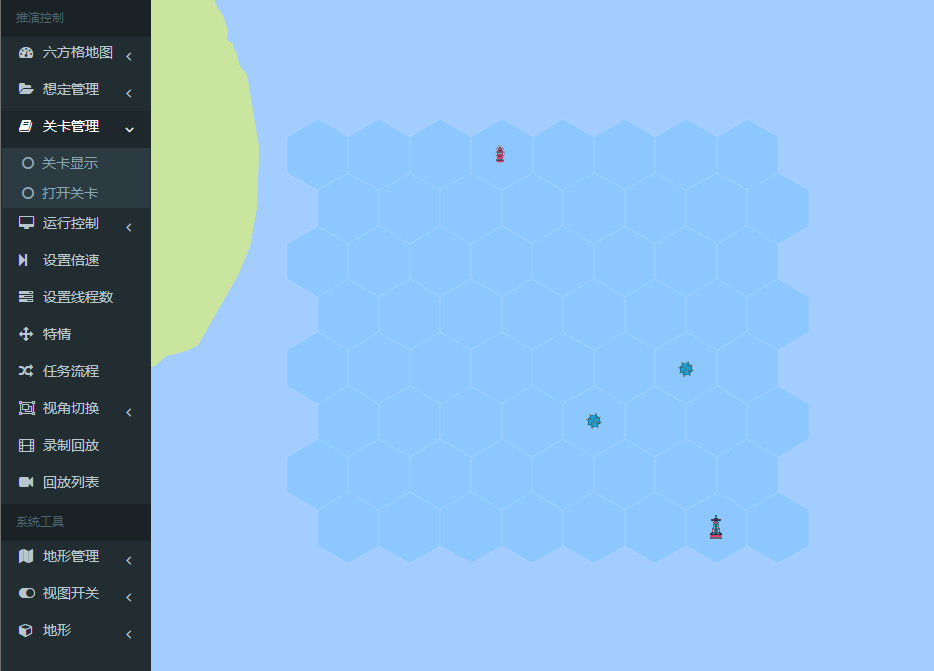


说明：从数据库中评估结果表中数据可看出，评估了10个回合，胜率为1（100%），回合平均步数为12步，回合的平均的分为9999。

## 4.2 挑战-航路规划-关卡2

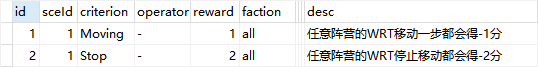
该关卡为挑战模式，类型为8\*8地图侦察探雷，场景的定义为：

* 想定和地图分别为：



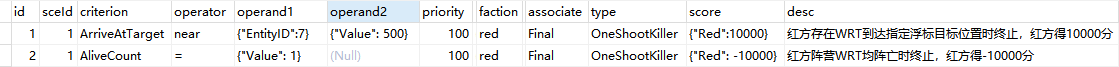
说明：场景中存在一艘红方无人艇、两个水雷和一个红方浮标，红方艇驶入到雷的方格中时雷会爆炸炸毁红方艇，红方浮标为目的地。

* 评分规则为：



说明：红方艇移动一步扣除1分，如果停下的话会扣除2分，促使红方艇找到最短路径。

* 终止规则为：

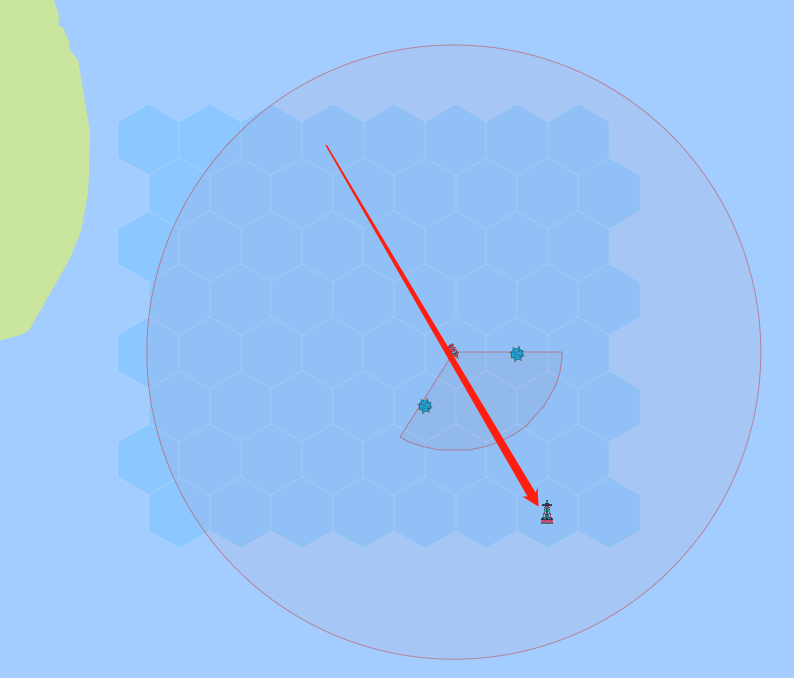
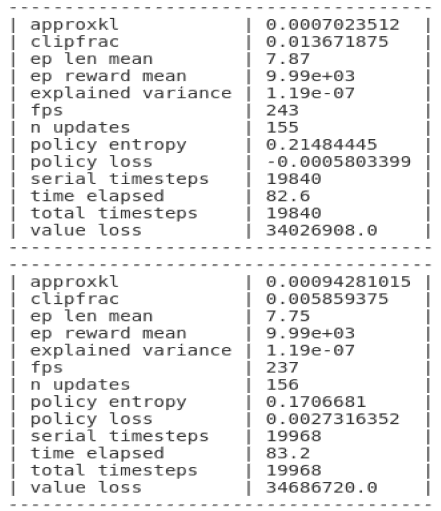


说明：当红方艇被炸毁时，对局结束并扣除10000分，当红方艇驶入到目的地时，对局结束并获得10000分奖励。

1. 启动docker训练模型；

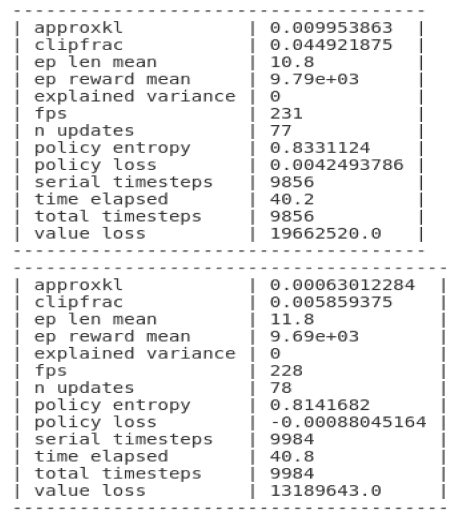
|  |
| --- |
| train.py -u http://127.0.0.1:9999/common/queryScore -i 3 -c 2 -a PPO -v SimuWRTEnv-v5 -m wrt\_sce\_3\_ppo2\_8 -d wrt\_sce\_3\_ppo2\_8\_tensorboard -s 20000 -t 2 |

说明：不同于4\*4场景，8\*8场景的状态空间已经增大了很多，我们尝试加大训练次数，尝试训练两次，每次训练20000步。



观察训练日志输出，最终回合的平均得分为9.99e+03，说明最终每个回合艇必成功到达了目的地，而回合的平均步数为7.75步，说明模型的决策已经快接近最优路线的7步，我们可以尝试再继续累加训练看模型能否做到每次均走最优路线。

|  |
| --- |
| train.py -u http://127.0.0.1:9999/common/queryScore -i 3 -c 2 -a PPO -v SimuWRTEnv-v5 -m wrt\_sce\_3\_ppo2\_8 -d wrt\_sce\_3\_ppo2\_8\_tensorboard -s 10000 -t 1 |



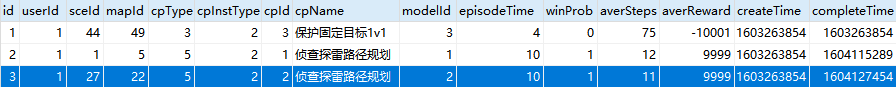
这里尝试训练10000次后，发现模型回合平均的步数增大了，平切loss值增大了，说明模型在尝试发散，探索更多的空间。

1. 启动docker验证模型；

|  |
| --- |
| play.py -u 1 -i 3 -c 2 -a PPO -v SimuWRTEnv-v5 -m wrt\_sce\_3\_ppo2\_8 -t 10 |

说明：尝试验证算法模型10个回合。

1. 生成验证结果数据。



说明：从数据库中评估结果表中数据可看出，评估了10个回合，胜率为1（100%），回合平均步数为11步，回合的平均的分为9999。

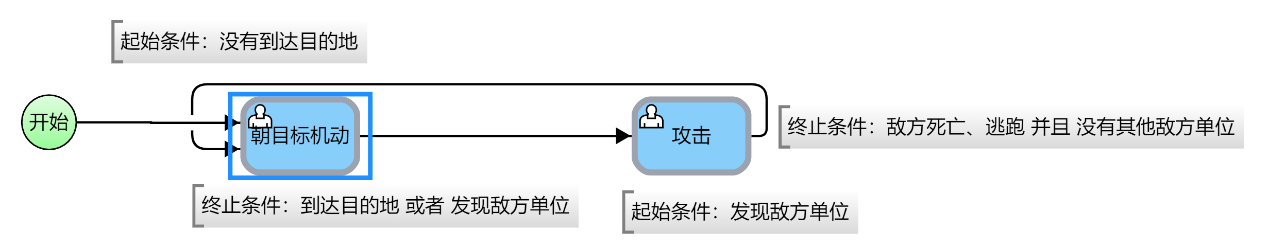
## 4.3 挑战-航路规划-关卡3

该关卡为挑战模式，类型为65\*65地图保护固定目标，场景的定义为：

* 想定和地图分别为：

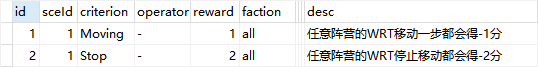


说明：场景中存在一艘红方无人艇、一个红方浮标和一艘蓝方艇，红方浮标为保护目标，蓝方艇会向红方浮标行进，过程中如果遇到红方艇会进行攻击。



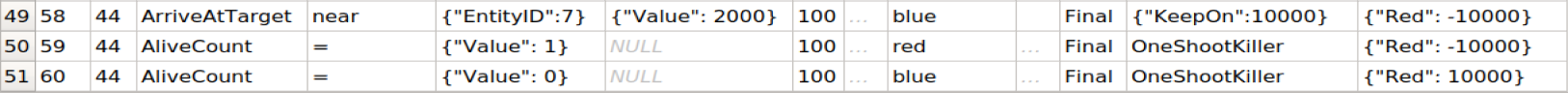
蓝方WRT按加载会加载上图的作战任务，如果蓝方WRT未到达目的地（红方的保护目标），会向目的地方向航行，航行过程中如果存在红方WRT在自己的攻击范围内，则会就近选择红方WRT进行攻击，并锁定目标向其航行，如果攻击范围内无红方WRT则会继续向目的地航行，直到到达目的地位置。

* 评分规则为：



说明：红方艇移动一步扣除1分，如果停下的话会扣除2分，促使红方艇移动不要停，以加速探索。

* 终止规则为：

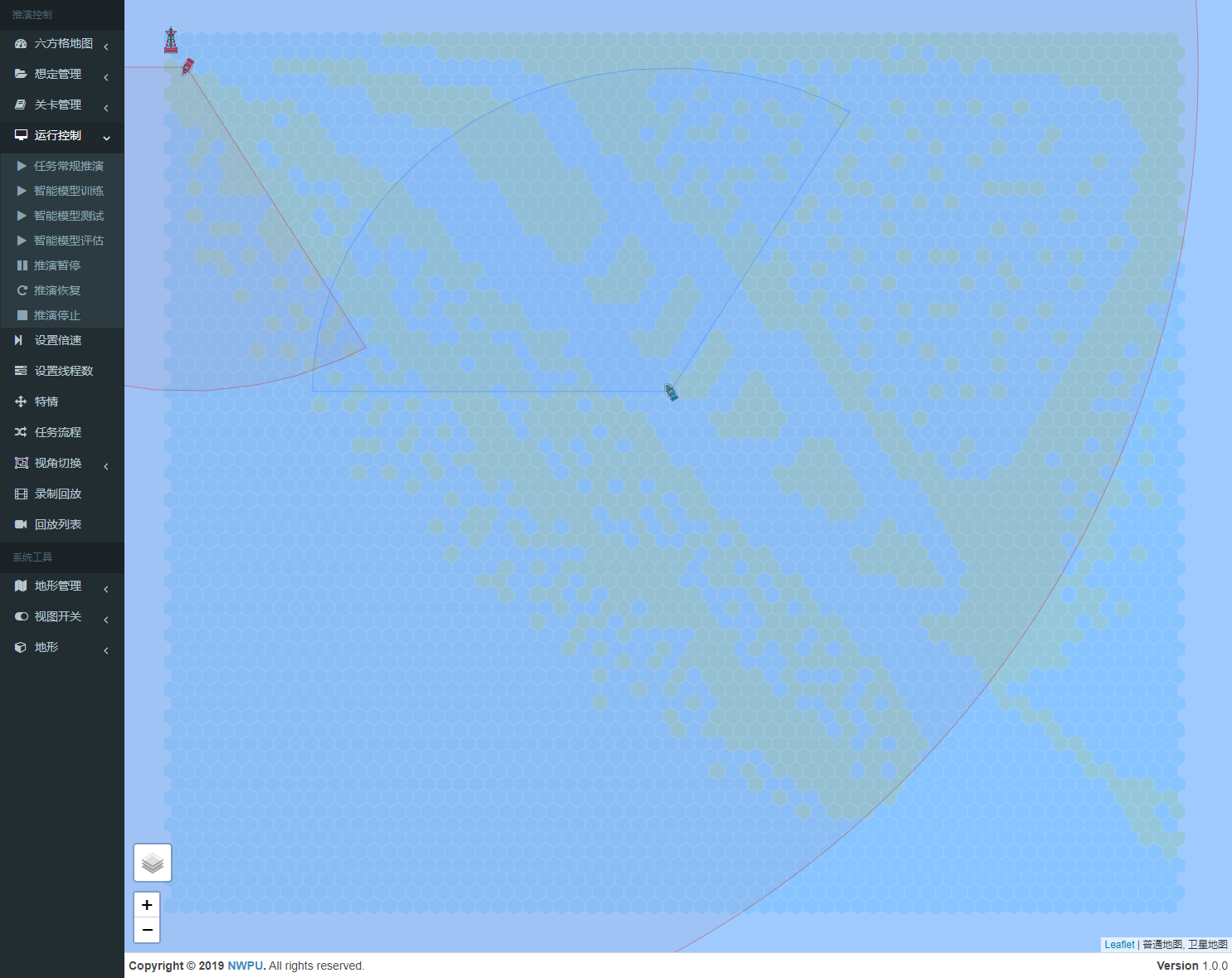


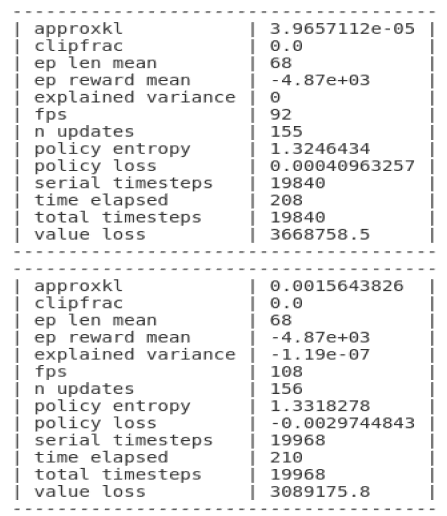
说明：当蓝方WRT靠近保护目标2KM范围内持续10秒时对局结束，红方扣除10000分；当红方WRT阵亡时对局结束，红方扣除10000分；当蓝方WRT阵亡时对局结束，红方得10000分。

1. 启动docker训练模型；

|  |
| --- |
| train.py -u http://127.0.0.1:9999/common/queryScore -i 4 -c 3 -a PPO -v SimuWRTEnv-v8 -m wrt\_sce\_1\_ppo2\_65\_1v1 -d wrt\_sce\_1\_ppo2\_65\_1v1\_tensorboard -s 20000 -t 10 |

说明：65\*65的空间会非常庞大，我们尝试训练10次每次20000步看看效果。





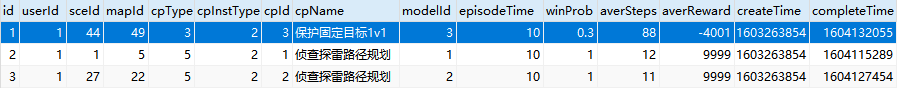
观察训练日志输出，最终回合的平均得分为-4.87e+03，相比于必败的-1e+04分数，说明此时红方已经约有25%的概率获胜了；回合的平均步数为68步。

1. 启动docker验证模型；

|  |
| --- |
| play.py -u 1 -i 1 -c 3 -a PPO -v SimuWRTEnv-v8 -m wrt\_sce\_1\_ppo2\_65\_1v1 -t 10 |

说明：尝试验证算法模型10个回合。

1. 生成验证结果数据。



说明：从数据库中评估结果表中数据可看出，评估了10个回合，胜率为0.3（30%），回合平均步数为88步，回合的平均的分为-4001分。

# 实验结果分析

采用PPO算法训练关卡1-3场景的对抗决策模型，均能得到不错的效果。结合关卡环境状态空间大小的不同，算法模型想训练出预想的能力需要训练的时长会有明显的差异。通过本次实验，也充分验证了基于强化学习的水面WRT对抗系统的可用性和有效性。