AFSIM
使用说明文档

2024 年 8 月 17 日

序号	文件版本	制修日期	编制人员	批准
1	A1	2024/8/17	孟子涵、任鹏宇	孙建壮

目录

1. 概述

1.1. AFSIM 介绍

AFSIM（The Advanced Framework for Simulation, Integration and Modeling - 用于模拟、集成、建模的高级框架）是一个用于创建在地理环境下模拟平台间相互作用的面向对象的 C++库。在这个框架中，顶层的对象被称为“平台（platform）”（也被称为个体（body）、实体（entity）或玩家（player））。平台可以被看做是简单的个体，上面附加了各种系统和属性。平台可以代表多种不同类型的实体，包括但不限于载体（地面载体、空中载体、太空载体、水面载体、水下载体）、建筑物或生物。平台间的相互作用包括但不限于传感器检测、碰撞和通信。

1.2. AFSIM 组成

AFSIM 的基本运作流程由三大模块组成：

- **Wizard**

  路径：.../bin/wizard.exe

  AFSIM 集成开发环境，对想定文件的编辑需要在此软件进行。

- **Mystic**

  路径：.../bin/mystic.exe

  推演复盘工具，用于播放推演回放文件（.evt）。

- **Warlock**

  路径：.../bin/warlock.exe

  人在回路工具，允许用户在仿真过程中实时参与和干涉。

除此之外还有一些其他的辅助模块：

- **Mover Create**

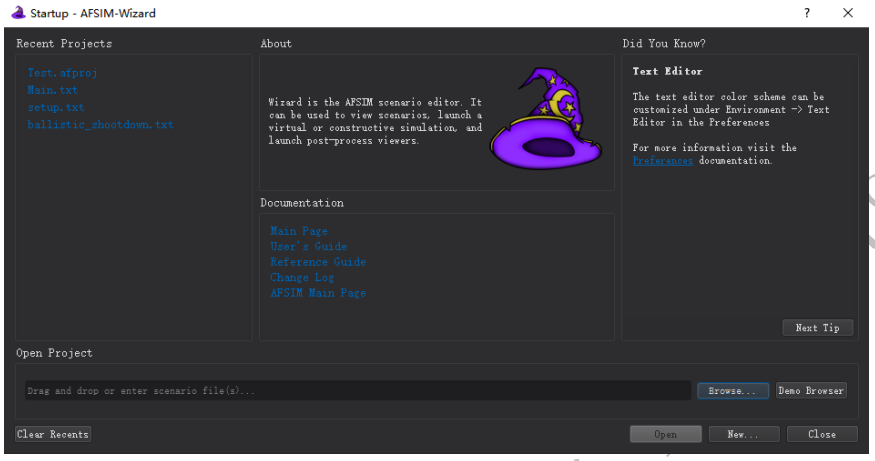
  路径：.../bin/mover_create.exe

  机动模型创建工具。

2. Wizard 基本使用

2.1. 打开/新建项目

（1）双击项目目录下的.../bin/wizard.exe，打开如下窗口。

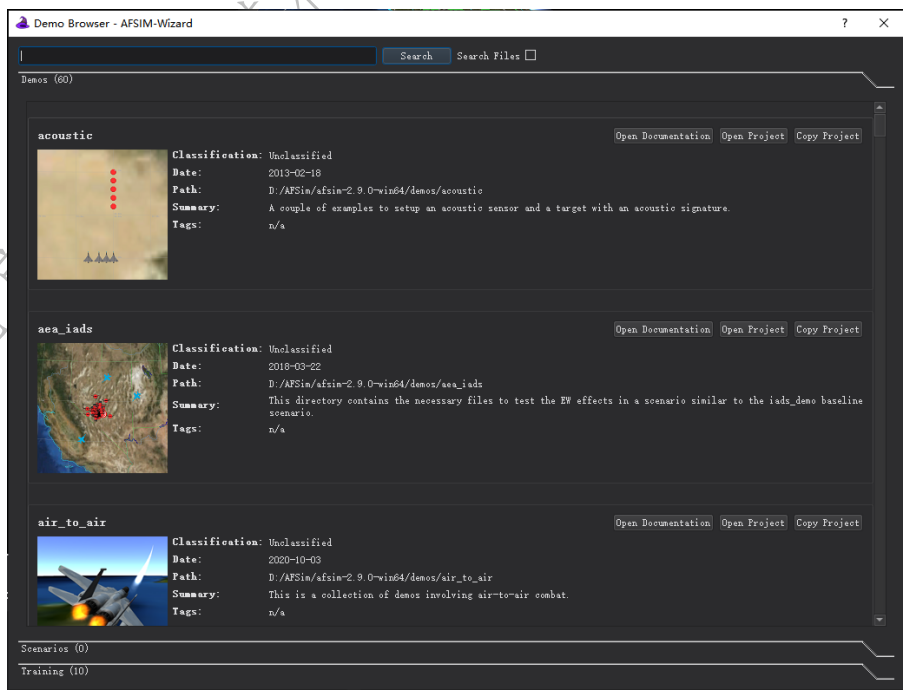


（2）左侧的 Recent Projects 列表中显示的是先前启动过的项目，点击打开对应项目。

（3）中部下侧是开发文档主要章节的的快速连接，点击跳转至开发文档。

（4）点击右下侧的“Browse...”按钮可检索文件目录（或者直接在输入框中输入文件路径，需要精确到.afproj 文件），路径输入后点击“Open”按钮打开项目文件。

（5）点击右下侧的“Demo Browser”按钮可打开示例浏览窗口，在窗口中选择指定的示例，点击“Open Project”按钮打开示例项目，点击“Copy Project”按钮则会选择一个文件夹复制示例项目（注：直接打开示例项目会导致在示例项目中的修改覆写到项目文件）。



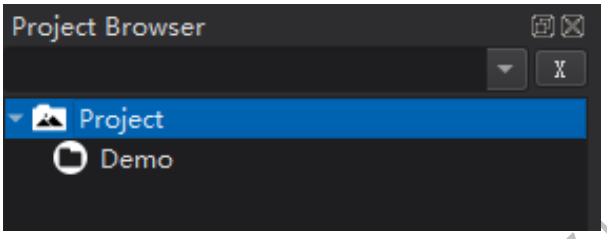
（6）点击右下侧的“New”按钮会选择一个文件夹新建项目（注：新建项目不会生成.afproj

文件，该文件会在项目第一次保存时生成）。

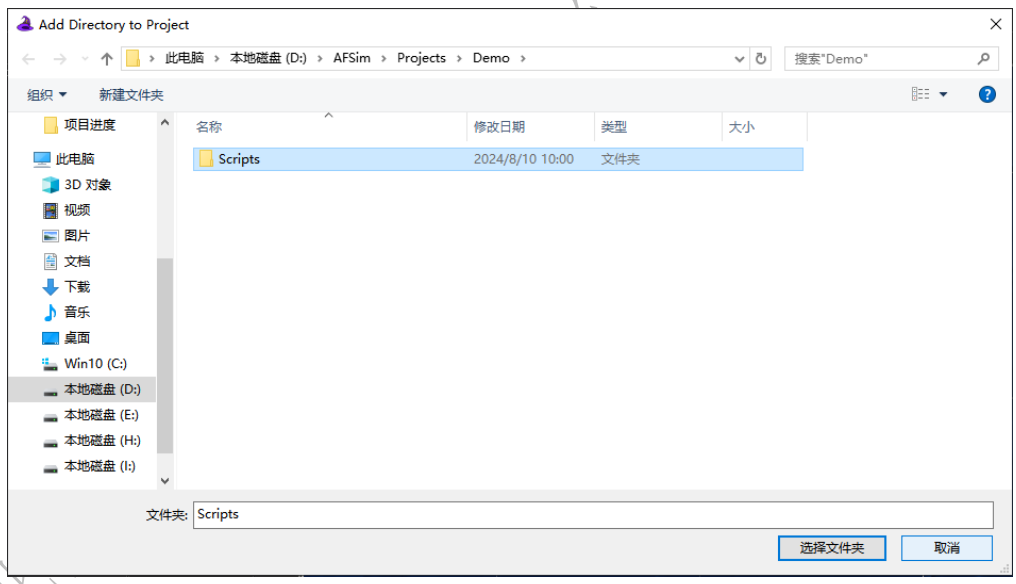
（7）也可以通过 Wizard 顶部菜单的 File/Open Project 打开项目，通过 File/Open Recent 打开近期项目，通过 File/New Project...新建项目。

2.2. 新建文件

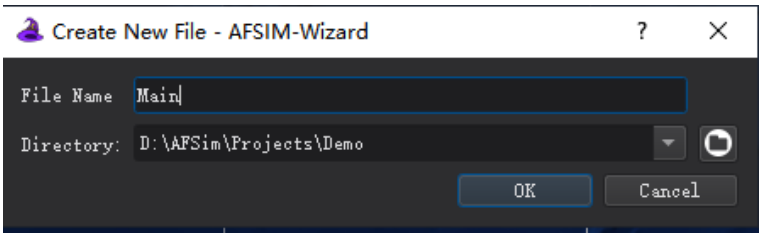
（1）窗口项目浏览器（Projet Browser）中会显示当前的项目目录（如果没有该面板则通过顶部菜单的 View/Project Browser 打开，如果该项前面有√表示窗口已经存在）。



（2）右键 Project，在右键菜单中选择 Add Directory to Project...，打开项目目录下的文件资源管理窗口，右键新建文件夹（此方法不唯一，通过任何方式在该目录下新建文件夹均可，但是在 Wizard 内部无法直接新建文件夹），然后点击“取消”按钮（点“选择文件夹”它会告诉你文件夹已存在）。

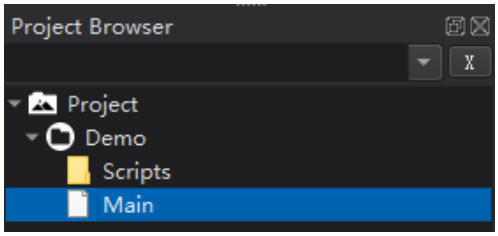


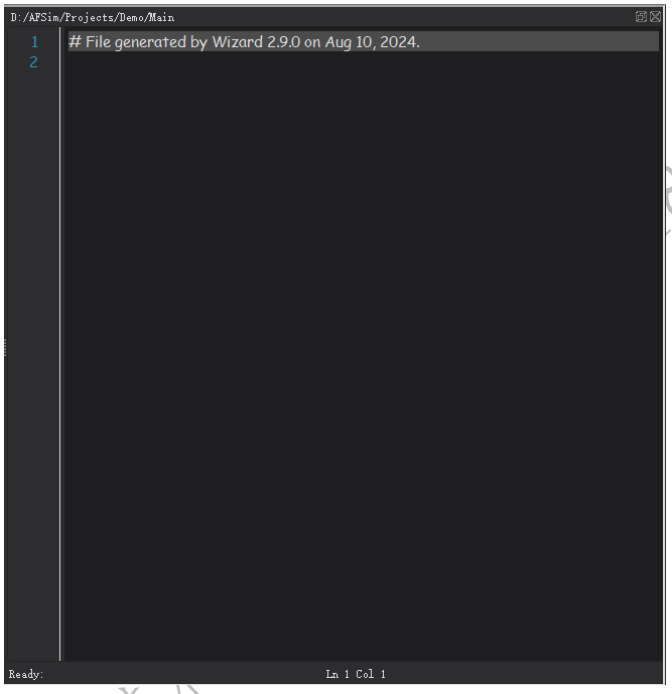
（3）右键项目根目录（注意是项目根目录不是 Project，也就是上图中的 Demo）或者项目目录下已有的文件夹，在右键菜单中选择“New File...”，打开文件创建窗口。



（4）在 File Name 中填写文件名称（包含格式后缀），在 Directory 中选择文件创建路径（默认为上一步右键的文件夹路径），点击“OK”按钮，在指定目录下会生成该名称的空文件，

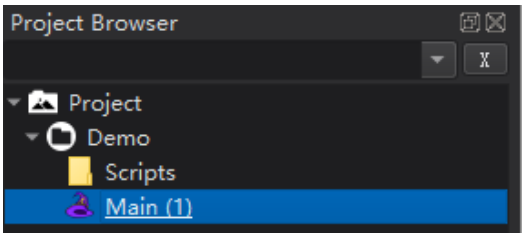
同时文件会在文件编辑面板被打开。也可以通过其他方式在项目目录下直接新建文件，此时新建的文件除了比通过项目浏览器（Project Browser）新建的文件少一行关于创建时间的注释外，并无任何区别。



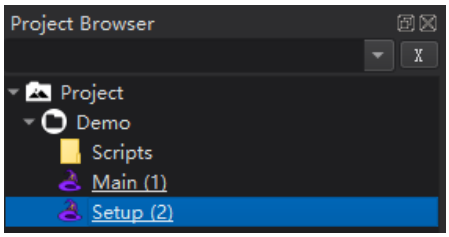


2.3. 设定启动文件

（1）右键一个文件，在右键菜单中选择“Set as Startup File”，文件将被设定为启动文件（即项目运行时被第一个执行的文件），此时在项目浏览器（Project Browser）中该文件的图标会变成 Wizard 的图标，这表明该文件已被加入模拟流程中。此时文件名称后跟的编号表示文件在模拟流程中的执行顺序。

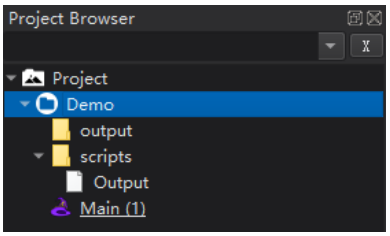


（2）右键其他文件，在右键菜单中选择“Add to Startup File”，文件也将被直接加入模拟流程中，但不是作为第一个执行的文件。



3. 基本模拟流程

1. 创建项目文件



以上图为例，新建一个名为“Demo”的项目并打开。在项目目录下新建两个文件夹，分别命名为“output”和“scripts”。在根目录下新建一个名为“Main”的文件，在“scripts”目录下新建一个名为“Output”的文件。将 Main 设定为启动文件。

2. 编辑文件内容

2.1. Main

```
1  # File generated by Wizard 2.9.0 on Aug 10, 2024.
2
3  include_once scripts/Output
4
5  execute at_time 10 s absolute
6    writeln("Hello World");
7  end_execute
8
9  end_time 60 s
```

Demo/Main

以上图为例，在 Main 中写入上图中内容。

第 3 行表示引用文件路径“Demo/scripts/”下 Output 文件中的内容，即，Output 中的内容将在该语句的位置被执行（**详见文档章节：File Commands**）。

第 5~7 行为 execute 脚本块（**详见文档章节：execute**），执行了一个 writeln 语句（**详见文档章节：__BUILTIN__（注意前后都是两个下划线）**），将会在模拟开始的第 10 秒向控制台输出“Hello World”。

第 9 行限制了模拟将在开始的第 60 秒结束（**详见文档章节：Simulation Control Commands**）。

2.2. Output

```
1  # File generated by Wizard 2.9.0 on Aug 10, 2024
2
3  event_output
4    file output/output_%d.evt
5  end_event_output
6
7  event_pipe
8    file output/output_%d.aer
9  end_event_pipe
10
11 final_run_number 3
```

Demo/output/Output

以上图为例，在“Output”中写入上图中内容。

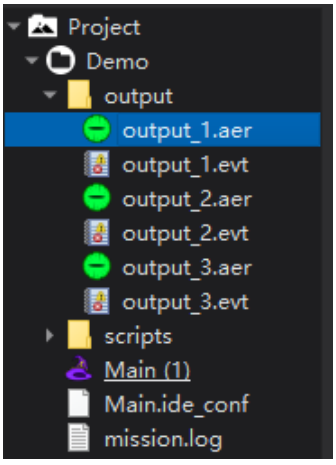
第 3~5 行为 event_output 脚本块（**详见文档章节：event_output**），在“Demo/output/”路径下创建一个名为“output_n”的包含时间戳的仿真事件文件，n 表示本轮重复推演的次数（代码中的%d 表示重名文件索引）。

第 7~9 行为 event_pipe 脚本块（**详见文档章节：event_pipe**），在“Demo/output/”路径下创建一个名为“output_n”的二进制仿真事件文件，n 表示本轮重复推演的次数。

第 11 行表示本轮将重复推演 3 次（**详见文档章节：Monte Carlo Iteration**）。

3. 输出推演文件

看到上方工具栏，初始默认的模拟执行模式为“Mission”，此时点击“Run”按钮（实心三角箭头）将会快速执行整个模拟流程，并按照 Output 中的代码输出仿真事件文件。



双击 aer 文件可以在 Mystic 中打开推演回放。

4. 推演复盘

（由于场景中没有平台时，推演的内容将为空，因此此处输出的推演复盘中没有内容，建议等到加入平台的章节后再做尝试）

打开 Mystic 后，看到上方工具栏。

点击进度条左侧第一个按钮时，可以搜索并跳转到对应的世界地点。

点击进度条左侧第二个按钮时，可以在记录当前位置并在第二个按钮右侧生成新的按钮，点击该按钮时跳转到对应位置。

点击进度条左侧第三个按钮时，可以直接跳转到推演的指定时间（也可以通过拖动进度条来跳转）。

点击进度条左侧第四个按钮时，可以调整播放速度。

点击进度条左侧第五个按钮时，推演时间重置回 0.0。

点击进度条左侧第六个按钮时，推演将会倒播。

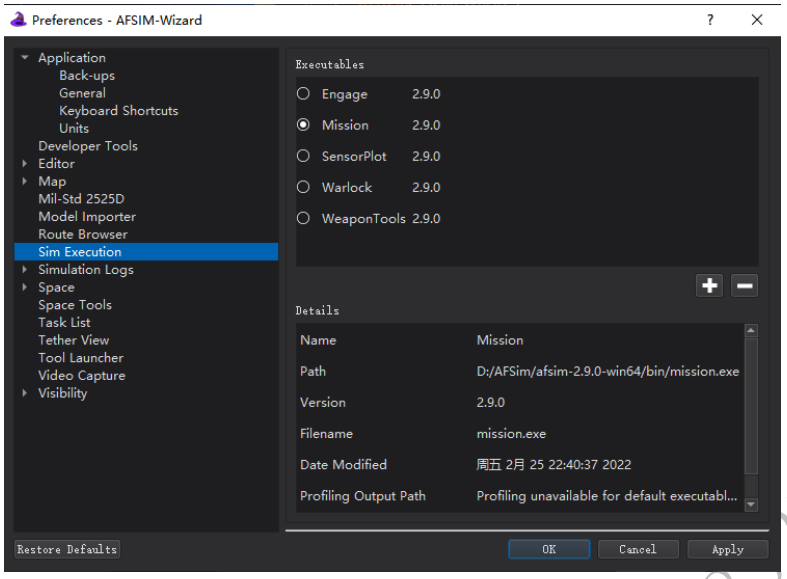
点击进度条左侧第七个按钮时，推演将会正播。

点击进度条右侧第一个按钮时，推演将会跳转到推演开始位置。

点击进度条右侧第二个按钮时，推演将会跳转到推演结束位置。

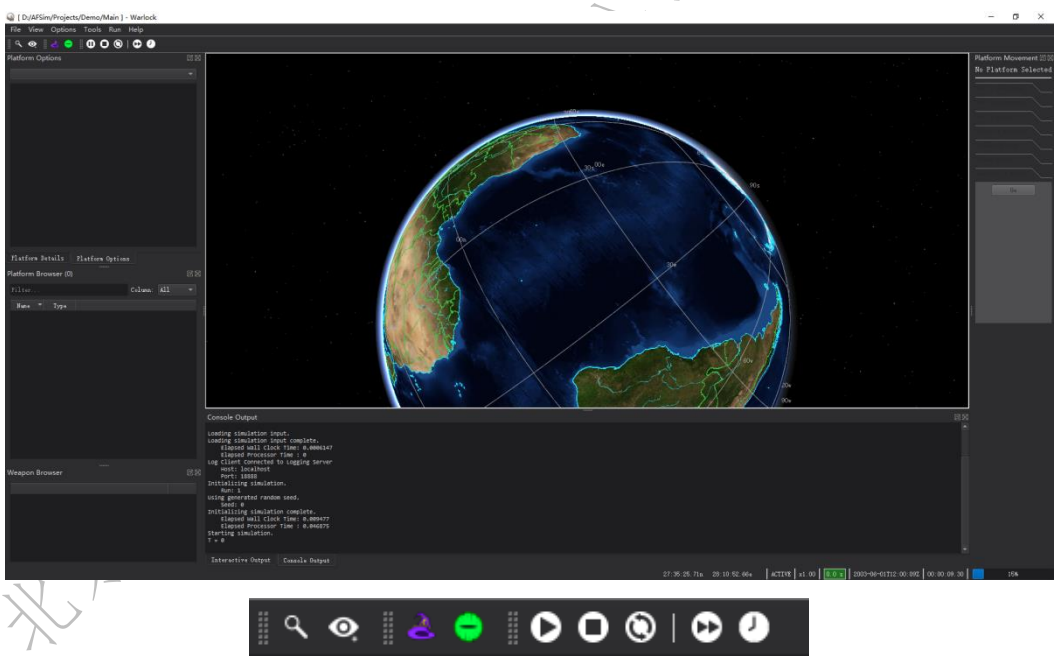
5. 实时推演

关闭 Mystic，回到 Wizard，点击上方工具栏中的“Mission”字样，可以打开模拟执行模式的设置窗口。



在 Executables 中选择 Warlock，点击“OK”按钮。

当前的模拟执行模式为“Warlock”，此时点击“Run”按钮（实心三角箭头）将会打开 Warlock 并开始实时推演。



Warlock 的工具栏与 Mystic 的工具栏大致相同，不同点在于可以快速打开 Wizard 和 Mystic，且不支持跳转到推演进度和倒播（因为是实时的），但可以向后跳转到推演的具体时间。

此外，Warlock 会在 Console Output 面板输出控制台内容，因此可以看到先前代码中在推演第 10s 输出的 Hello World 语句。

```
Starting simulation.
T = 0
Hello World
T = 10.000
T = 20.000
T = 30.000
T = 40.000
T = 50.000
T = 60.000
Simulation complete
    Elapsed Wall Clock Time: 66.7598
    Elapsed Processor Time : 12.3594
```

4. 平台

4.1. 概述

平台是 AFSIM 模拟的核心部分，所有的推演内容都围绕平台来进行（**详见文档章节：platform**）。

4.2. 平台定义基本命令

side <side-name>

阵营<阵营名称>

指定此类型平台所属的 'side' （'team' 或 'affiliation'）。side 子命令引用 蓝色、红色、国家/地区或团队名称。

默认值：none

icon <icon-name>

图标<图标名称>

指定在显示此类型的平台时要使用的图标的名称，如果平台类型在 dis_interface 模块中没有相关的 entity_type 子命令，则使用此图标。

marking <marking-name>

标记<标记名称>

指定要应用于平台的标记。此文本字段与 dis_interface 中的此平台相关联。

默认：无标记

indestructible (or destructible)

是否可被摧毁

如果设置为 indestructible 表明平台类型是不可摧毁的。每次命中都会根据以下公式更新生存的累积概率：

Ps(new) = Ps(old) * (1 - Pk)

默认:destructible(可被摧毁的)

on_broken [remove | disable | disabled_but_movable]

指示在平台被破坏时应执行的操作。

如果指定了“ remove”，则平台将在被破坏时从模拟中删除。

如果指定了“disable”（禁用），那么当平台被破坏时，它将保留在仿真中。平台的运动将被停止，所有子系统将被设置为“非运行状态”，这意味着它们不能再被重新激活。脚本方法 WsfPlatform.DamageFactor 将返回一个值为 1.0。

如果指定了“disabled_but_movable”（禁用但可移动），那么当平台被破坏时，它将保留在仿真中。平台的运动将继续进行。一些移动器可能允许运动受到破坏状态的影响，例如减速和失去控制。所有子系统将被设置为“非运行状态”，这意味着它们不能再被重新激活。脚本方法 WsfPlatform.DamageFactor 将返回一个值为 1.0。

默认值：remove

注意：每当平台受到损伤时，无论平台是否被破坏，都会调用 on_damage_received 脚本。该脚本可以用来改变信号特征、外观等。然而，应该记住的是，所有子系统都将被禁用，因此平台在被破坏后可能无法执行任何有意义的动作。

spatial_domain [land | air | subsurface | surface | space]

空间领域

指示平台主要操作的空间领域。这有时用来分类一个对象是“陆地”对象、“太空”对象等。

默认设置：如果平台有一个移动器（mover），则默认值将从移动器的类型推断得出（例如，对于 WSF_AIR_MOVER，它将是“air”即空中）。如果平台没有移动器，则默认假设为“land”即陆地。平台的空间领域可以是“land”（陆地）、“air”（空中）、“surface”（水面）、“subsurface”（水下）或“space”（太空）。

acoustic_signature

声学特性

指定此平台类型的声学特性定义。特性类型必须已使用 acoustic_signature 定义。

默认：None

infrared_signature

红外特性

指定此平台类型的红外特性定义。特性类型必须事先使用 infrared_signature 定义。

默认：None

注意：如果未指定 infrared_signature（红外特性），并且在发生需要它的感知事件时，将默认假设为 1000 w/sr（瓦特/球面度）。

inherent_contrast

视觉隐蔽性

在 AFSIM 仿真框架中，inherent_contrast 用于定义一个飞机或车辆在视觉上的隐蔽性，这个值会影响它在被敌方视觉传感器探测到的概率。如果这个参数没有被明确指定，系统可

能会使用一个默认值，这个默认值可能是基于该类型平台的典型特性。

默认：None

optical_reflectivity

光反射率

为这种平台类型指定光学反射率定义。特性类型必须已经使用 optical_reflectivity 定义过。

默认：None

如果未指定 optical_reflectivity（光学反射率），并且在发生需要它的感知事件时，将默认假设为 1.0。

optical_signature

光学特性

为这种平台类型指定光学特性定义。特性类型必须已经使用 optical_signature 定义过。

默认：None

注意：如果未指定 optical_signature（光学特性），但指定了长度、宽度和高度，那么在需要时这些值将被用来计算光学截面

注意：如果未指定 optical_signature（光学特性），并且长度、宽度和高度也未指定，那么将默认假设为 1000 平方米（m^2）。

radar_signature

雷达特性

为这种平台类型指定雷达特性定义。特性类型必须已经使用 radar_signature 定义过。

默认：None

注意：如果未指定 radar_signature（雷达特性），并且在发生需要它的感知事件时，将默认假设为 1000 平方米（m^2）。

geo_point <geo-point-name>

使用纬度、经度、海拔面元组定义命名位置。可以使用 WsfPlatform.GeoPoint 脚本方法访问该位置。

position

静态平台经纬度

指定平台的纬度和经度。此命令仅适用于没有移动器的静态平台。

默认 0n 0e

nutation_update_interval

指定用于 WCS-ECI 坐标转换的章动计算更新的时间间隔。

默认值：1000 s

mgrs_coordinate

指定平台在军事网格参考系统中的坐标。此命令仅适用于没有移动器的静态平台。

altitude [agl | msl]

静态平台高度

指定平台的高度。此命令仅适用于没有移动器的静态平台。如果平台中定义了移动器，则此命令会被忽略。

Default 0 m msl

creation_time

指定平台应该被添加到仿真中的仿真时间。平台已存在于内存中，但尚未成为仿真中的参与者。

默认值：0 sec（在仿真开始时创建）

heading

静态平台方向

指定平台的方向。此命令仅适用于没有移动器的静态平台。如果平台中定义了移动器，则此命令会被忽略。

默认值：0 deg

empty_mass

平台空重

指定平台的空重，这通常是一个固定值。空重是指平台不包括燃料、货物、人员等额外负载时的重量。这个参数对于仿真平台的性能计算、平衡分析和动力模型等都是重要的基础数据。

默认值：0 kg

注意：对于某些类型的移动器（mover），存在特定的方式来设置或计算平台的某些参数或属性，而不是使用通用的命令或方法。

fuel_mass

平台燃料质量

指定平台的燃料质量。当用户在输入文件中指定此值时，它被认为是一个固定量。然而，如果平台包含一个燃料对象，允许并且在运行时修改燃料质量，覆盖用这个关键字指定的值。

默认值：0 kg

注意：对于某些类型的移动器（mover），存在特定的方式来设置或计算平台的某些参数或属性，而不是使用通用的命令或方法。

payload_mass

指定平台的有效载荷质量。当用户在输入文件中指定此值时，它被认为是一个固定量。然而，某些运行时事件（例如，投放浮标）可能会改变有效载荷的质量。

默认值：0 kg

注意：对于某些类型的移动器（mover），存在特定的方式来设置或计算平台的某些参数或属性，而不是使用通用的命令或方法。

concealment_factor <concealment-factor>

用于表示一个平台在视觉上的隐蔽程度。值为 0.0 表示没有尝试隐藏。因子为 1.0 表示它被隐藏得无法被探测到（例如在建筑物内或地下掩体内）。端点之间的值代表不同程度的伪装。大多数传感器不会探测到隐蔽因子为 1.0 的平台，但任何较小的值都没有效果。

默认值：0.0

initial_damage_factor <initial_damage_factor>

损伤因子是一个在 [0 .. 1] 范围内的值，用来指示平台受到的损伤程度。值为 0 表示没有损伤，而值为 1 表示平台已经“损坏”。

默认值：0.0

track_manager track_manager#Commands track-manager-commands … end_track_manager

这个块中的命令由平台的主跟踪管理器处理。参见 track_manager。

aux_data <aux-data> … end_aux_data

为平台定义辅助数据。参见 aux_data。

height

length

width

指定平台的尺寸。

默认值：为所有数值均为 0。

注意：如果所有这些值都不为零，并且没有指定 optical_signature，则在需要时这些值将被用来计算光学截面。

category <category-name>

指定平台属于指定的类别。可以设置某些传感器忽略特定类别的平台。类别还可以通过 WsfPlatform.CategoryMemberOf 脚本方法访问。

注意：默认情况下，平台不属于任何类别。

clear_categories

具有取消之前任何类别命令的效果。

group_join <group-name>　　group_leave <group-name>

将平台从指定的组中添加或移除。

commander <commander-name>

command_chain <command-chain-name> <commander-name>

commander 指定平台在默认指挥链中的直接上级（指挥官）。

command_chain 指定平台在给定指挥链中的直接上级（指挥官）。

指挥链被决策程序用来确定命令和/或报告的接收者。

注意：有关更多信息和示例，请参见“指挥链”。

<command-chain-name>

这条命令所适用的指挥链的名称。（对于 commander，默认的指挥链是隐含的）。

<commander-name> | SELF

这个平台的直接上级（指挥官）的名称。

默认值：commander SELF

route route-commands … end_route

route 子命令提供了一组航点，这些航点定义了平台移动器的路径或路线。

add_mover <mover-type> mover-commands … end_mover

mover <mover-type> mover-commands … end_mover

移动器定义了一个平台可以移动的空间领域以及它在该领域内如何移动。

add mover 命令仅需要在 platform 命令内部使用（即，不是在 platform_type 命令内部）。

fuel <fuel-type> fuel-commands .. end_fuel

可以将燃料对象附加到平台以模拟燃料消耗的效果。

默认：没有定义燃料对象（不模拟燃料消耗）。

注意：一些移动器在内部模拟燃料消耗。

<component> <component-name> [<component-type>] component-commands … end_<component>

add <component> <component-name> <component-type> component-commands … end_<component>

edit <component> <component-name> component-commands … end_<component>

delete <component> <component-name>

向平台添加一个组件，修改或移除一个现有组件。

add <component> 命令向平台/平台类型添加一个新的组件。

edit <component> 命令修改平台/平台类型上的现有组件。

delete <component> 命令移除平台/平台类型上的现有组件。

<component> 命令是一个缩写版本，如果在平台类型内部使用则添加一个组件，如果在平台内部使用则修改现有组件。

有效的 <component> 包括通信（comm）、燃料（fuel）、移动器（mover）、处理器（processor）、传感器（sensor）等。

注意：当使用修改或删除命令对燃料或移动器进行操作时，你不需要指定

<component-name>。

注意：区域可以通过删除命令来移除，但不能通过添加或编辑命令来修改。

track … end_track

定义了一个预先通报的跟踪轨迹。

use_zone <shared-zone-type> as <zone-name>

将指定区域的副本附加到平台，并使用名称 <zone-name>。如果指定的区域没有定义姿态，复制的区域将使用平台当前的姿态。根据需要可以重复此命令。

可以将一个区域的定义复制并附加到一个平台对象上，并且可以为这个副本指定一个名字。如果原始区域没有指定具体的姿态（位置和方向），那么这个副本将采用平台当前的姿态。这个命令可以根据需要多次使用，以便为平台附加多个区域。这种功能在设置复杂仿真场景时非常有用，尤其是当需要将多个区域与特定平台相关联时。

zone … end_zone

zone_set … end_zone_set

zone 和 zone_set 子命令定义：

相对于平台姿态对齐的区域。

保持静态姿态的相对区域。

保持静态姿态的绝对区域。

另见：zone 和 zone_set 用于影响模拟行为的子命令。

navigation_errors … end_navigation_errors

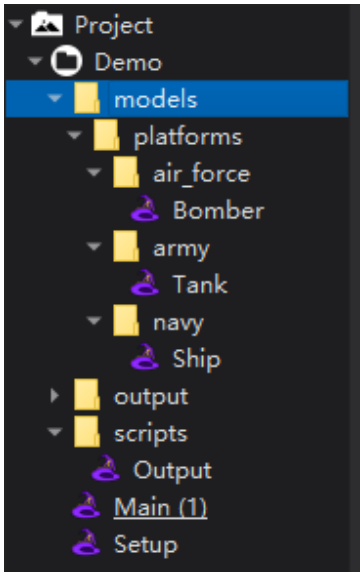
navigation_errors 块提供了一种方法来定义平台认为它所在位置与实际位置之间的误差。

默认：无导航错误。

4.3. 创建平台

4.3.1. 新建文件

新建下列文件夹和文件。（新建的文件有：Bomber、Tank、Ship、Setup）



在 Main 中引用 Setup，在 Setup 中引用 Bomber、Tank 和 Ship。

Setup 中的内容如图所示：

```
1  # File generated by Wizard 2.9.0 on Aug 10, 2024.
2
3  file_path models/platforms
4  include_once air_force/Bomber
5  include_once army/Tank
6  include_once navy/Ship
```

Demo/Setup

第 3 行表明引用了 platforms 这个文件夹，因此其下内容中对文件的引用都不需要重复写 platform 之前的路径，这种方式可以用在同一文件夹下需要引用的文件较多的情况下。

4.3.2. 创建平台类型

在创建平台前现需要创建平台的类型。

以 Bomber 为例，双击打开 Bomber 文件，写入下图中的内容。

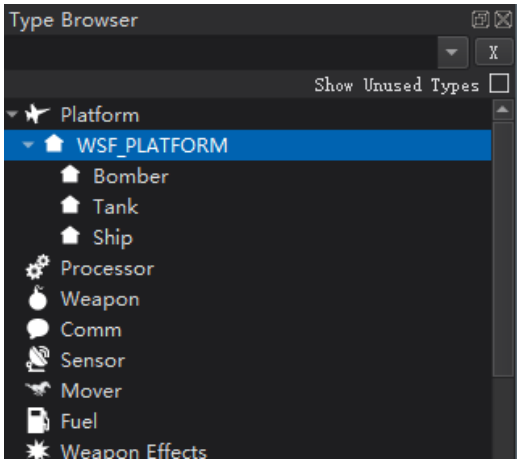
```
1  # File generated by Wizard 2.9.0 on Aug 10, 2024
2
3  platform_type Bomber WSF_PLATFORM
4    icon bomber
5  end_platform_type
```

Demo/models/platforms/air_force/Bomber

第 3~5 行为 platform_type 脚本块，创建的平台类型名称为 Bomber，继承自类型 WSF_PLATFORM（**详见文档章节：WSF_PLATFORM**），采用的模型名称为 bomber。

以同样的形式补全 Tank 和 Ship 中的代码。

然后看到 Wizard 的 Type Browser 面板（如果没有该面板则通过顶部菜单的 View/Type Browser 打开，如果该项前面有√表示窗口已经存在）。

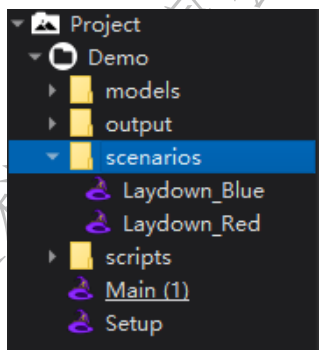


此面板会展示出所有已存在于项目中的脚本类型以及其继承的基类，选中 Show Unused Types 后可以展示出框架内包含的但未被此项目继承过的所有基类。

双击类型可以打开该类型声明的文件，双击基类可以打开该基类在开发文档中的对应章节。

4.3.3. 创建平台

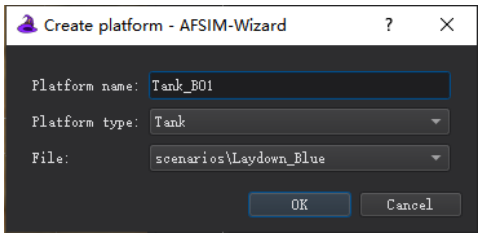
新建下列文件夹和文件（新建的文件有：Laydown_Blue、Laydown_Red），并在 Setup 中引用它们（注意引用的顺序，平台类型的定义必须要在平台的定义前）。



以 Laydown_Blue 为例，此文件中存放的是蓝队单位的定义数据。

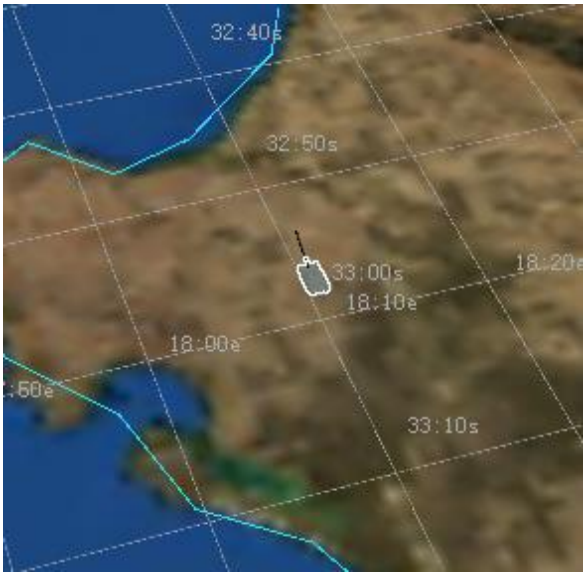
我们计划先为其创建一个 Tank。

在地图上选择一处合适的陆地位置（当然海洋也可以，但是这不太合理），右键选择 Add at Location/Platform，打开平台创建窗口。

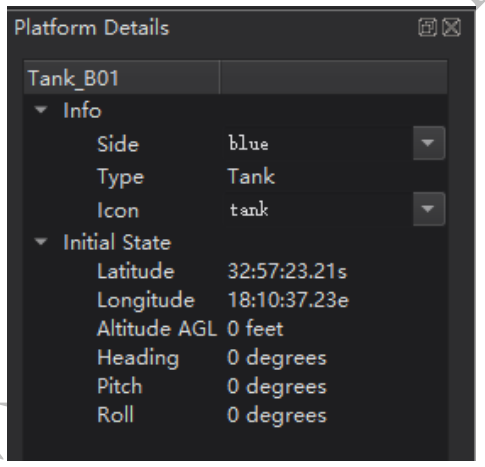


Platform type 选择 Tank，File 选择 Laydown_Blue，命名为“Tank_B01”点击“OK”按钮，平台被创建完成。

此时“Tank_B01”被创建。



在地图上选中方才创建的单位，在 Platform Details 面板中将 Side 选择为 blue，“Tank_B01”被分配至蓝队（如果没有该面板则通过顶部菜单的 View/Platform Details 打开，如果该项前面有√表示窗口已经存在）。



打开 Laydown_Blue 可以看到“Tank_B01”的定义代码。

```
1  # File generated by Wizard 2.9.0 on Aug 10, 2024.
2
3  platform Tank_B01 Tank
4    position 32:57:23.207s 18:10:37.232e
5    side blue
6  end_platform
7
```

Demo/scenarios/Laydown_Blue

第 3~5 行为 platform 脚本块，创建的平台名称为“Tank_B01”，继承自类型 Tank，位置为纬度 32:57:23.207、经度 18:10:37.232，所属方为蓝方。

根据你的需要以同样的方式定义红蓝双方的单位，可以在文件中以上述格式定义平台，也可以通过地图右键快速定义。

平台还有一些常用属性，我们将以蓝方的“Bomber_B01”为例。

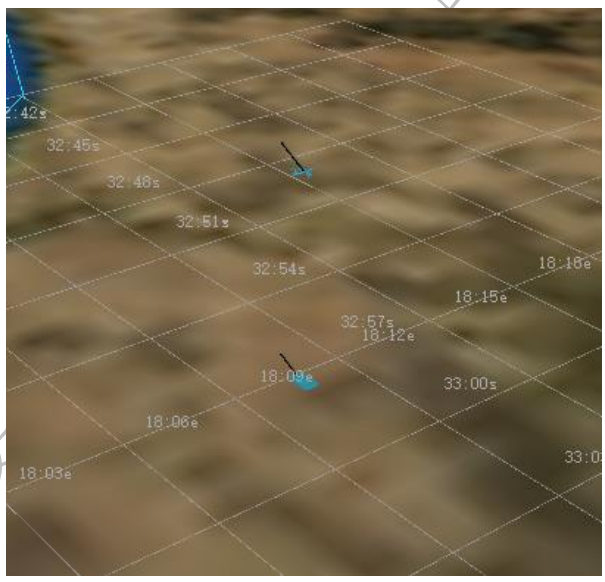
```
1   # File generated by Wizard 2.9.0 on Aug 10, 2024
2
3   platform Tank_B01 Tank
4     position 32:57:23.207s 18:10:37.232e
5     side blue
6   end_platform
7
8   platform Bomber_B01 Bomber
9     position 32:57:23.207s 18:10:37.232e
10    side blue
11    altitude 27000 feet
12    heading 90 deg
13  end_platform
14
```

Demo/scenarios/Laydown_Blue

如图所示，“Bomber_B01”的平面坐标于“Tank_B01”完全相同，但二者的高度和朝向不同。

第 11 行定义了“Bomber_B01”的高度为 27000 英尺，第 12 行定义了“Bomber_B01”的朝向为 90 度（常用单位详见文档章节：Argument Types）。

在地图中按住鼠标中键拖动可以调节视角，调整到合适的角度后可以看出二者的高度及朝向区别。



按住 Ctrl 时用鼠标左键可以拖动平台。

4.3.4. 可创建的平台类型

在 AFSIM（Advanced Framework for Simulation, Integration, and Modeling）仿真框架中，你可以创建多种类型的平台，这些平台代表了仿真环境中的不同实体。以下是一些常见的平台类型：

1. 空中平台：包括各种类型的飞机、直升机、无人机（UAV）、导弹等。
2. 地面平台：包括坦克、装甲车、运输车辆、步兵等。
3. 海上平台：包括各种类型的舰船、潜艇、巡逻艇、航空母舰等。

4. 太空平台：包括卫星、宇宙飞船、空间站等。
5. 静态平台：如建筑物、基础设施、桥梁、通信塔等，这些通常不移动。
6. 虚拟传感器平台：用于模拟雷达、声纳、光学传感器等传感器的虚拟实体。
7. 电子战平台：模拟电子战设备，如干扰器、通信系统、电子支援措施（ESM）等。
8. 网络战平台：模拟网络攻击和防御系统。
9. 生物实体：在某些仿真中，可能需要模拟人类或其他生物的行为。
10. 环境实体：如气象条件、海洋流、地形等，这些虽然不是传统意义上的平台，但可以作为影响其他平台行为的环境因素。
11. 综合系统：集成了多种传感器、武器系统和通信设备的复杂平台。
12. 自定义平台：根据特定仿真需求，用户可以创建自定义类型的平台。

AFSIM 提供了灵活性，允许用户根据仿真的目的和复杂性来定义和创建所需的平台类型。这些平台可以具有不同的属性、行为和交互方式，以模拟现实世界中的各种情况和条件。通过脚本和仿真配置文件，用户可以详细地定义每个平台的初始状态、行为模型和交互规则。

4.4. 平台组件

平台往往会包含不同类型的各种组件，平台的各种功能也是由组件提供的。

平台上会挂载的组件通常分为五大类：

（1）运动组件 Mover - 定义平台的运动

（2）传感器组件 Sensor - 定义平台对周围环境的感知

（3）武器组件 Weapon - 定义平台的武器

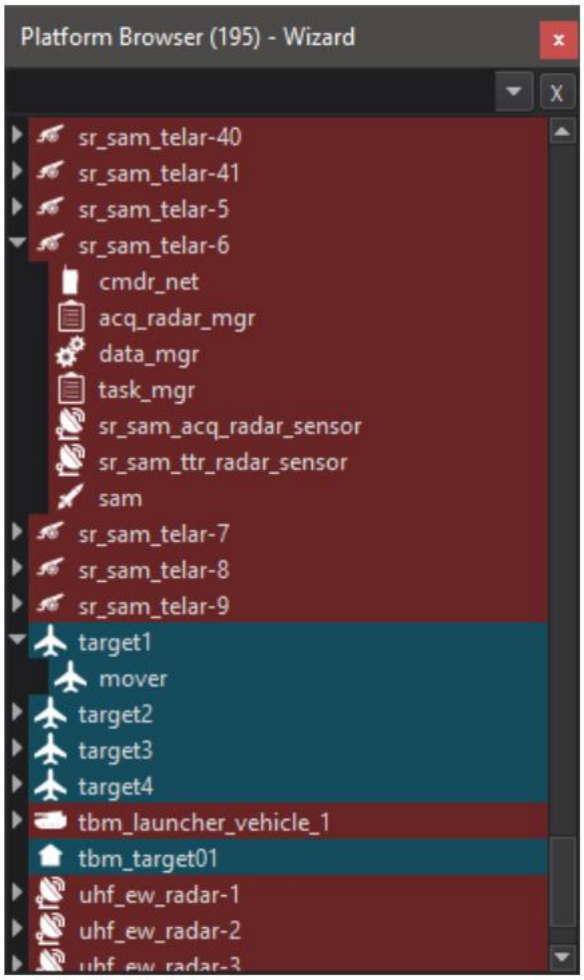
（4）通信组件 Comm - 定义平台间的通信链路

（5）处理器组件 Processor - 定义平台的行为

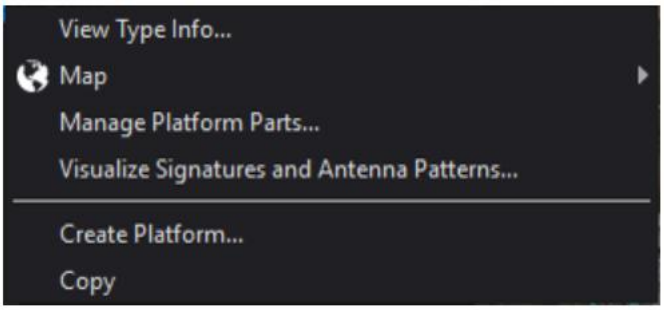
4.5. 平台浏览器

Platform Browser（平台浏览器） 对话框显示当前方案中存在的平台列表。浏览器中的平台按团队进行颜色编码。平台浏览器可通过“View”菜单访问。它可以停靠在主窗口的左侧或右侧停靠区域中。浏览器也可能作为单独的窗口弹出。

平台浏览器对话框如下图所示:



右键单击平台将显示平台上下文菜单，用户可以从中执行各种操作，例如添加范围环、使用测量工具从平台测量到另一个位置、添加装饰以及将屏幕置于平台中心。

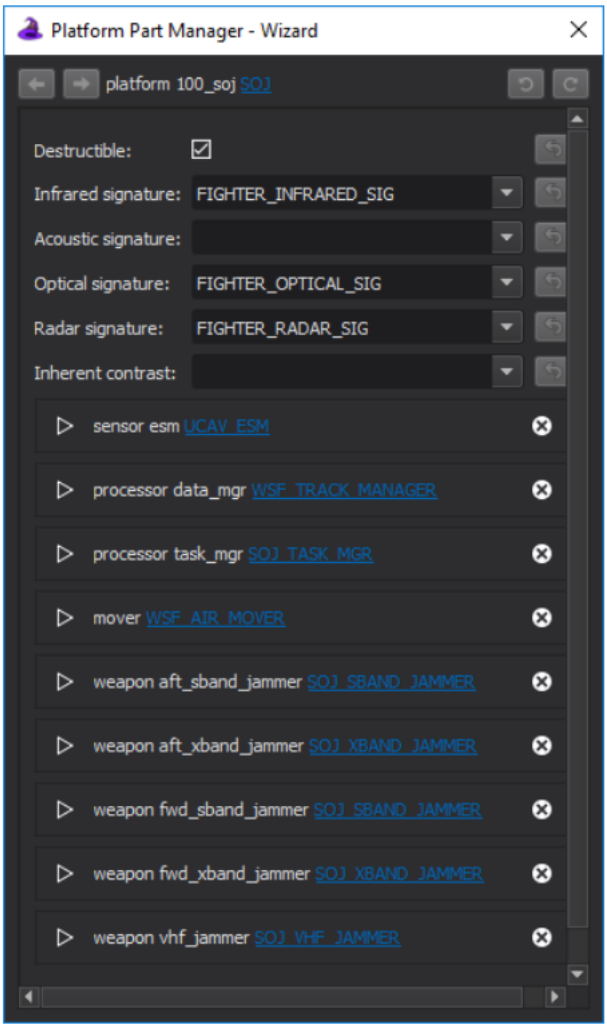


Wizard Platform Browser 上下文菜单还包含用于显示平台类型信息的选项。选择此选项将显示定义平台的文件和行号，以及其类型及其定义在文件中的位置。可以使用“管理平台部件...”来添加和编辑平台组件平台上下文菜单中的选项。这将打开 Platform Part Manager。Platform Browser 中的每个平台项都可以展开以显示其组件。双击平台或平台组件将打开 文本编辑器 并导航到相应的定义。

场景中的平台数显示在平台浏览器标题栏中的括号中。浏览器的顶部包含一个搜索字段。在此字段中键入内容将进行筛选，以显示名称中包含输入文本的平台或平台组件。

4.6. 平台部件管理器

Platform Part Manager 提供了一种便捷的方式来查看和更改属于平台、平台类型和平台组件的各种属性的值。



可以通过右键单击 Platform Browser 中的平台或 Type Browser 中的平台类型并选择“Manage platform parts...”来访问它。在上下文菜单中。

该窗口的顶部有一个标题部分，显示正在查看的平台或平台类型的声明，以及后退/前进导航按钮（位于左侧）和撤消/重做按钮（位于右侧）。

窗口的主体包含每个平台部分的可编辑属性值和可展开的部分。展开平台部件部分将显示该部件的可编辑属性。

4.6.1. 属性

属性在窗口正文中以表单样式列出，标签在左侧，可编辑值在右侧。

某些属性可能具有多个可编辑值和/或其他特殊功能。如有疑问，请尝试将鼠标悬停在此类元素上以显示其工具提示。

每个属性的右侧都有一个“重置为继承的值”按钮，该按钮会将属性重置为平台或平台类

型的父级中的值。已被覆盖的属性将具有白色背景。

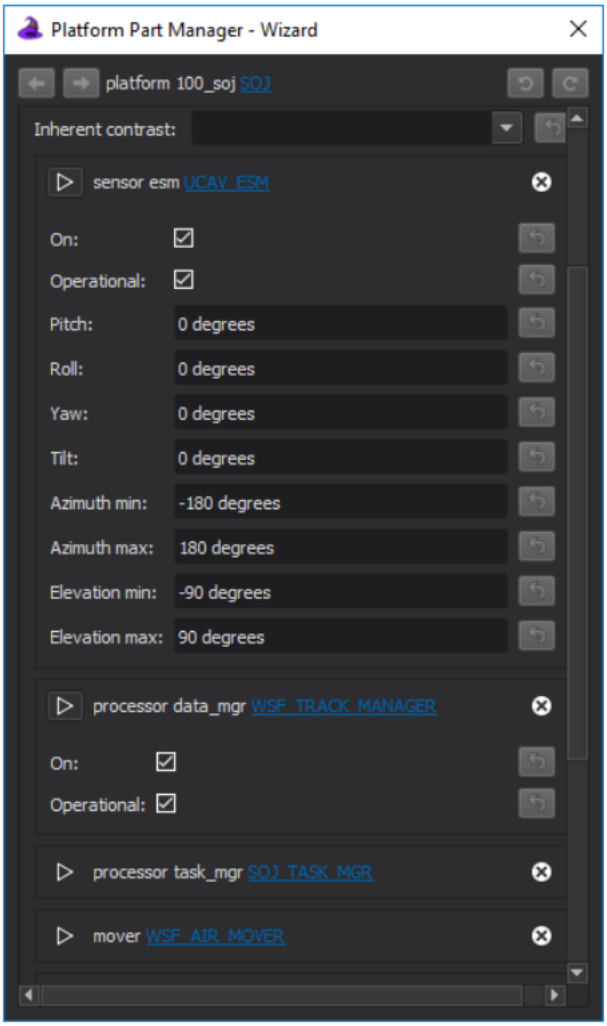
编辑完成后，可以完全还原更改或将更改应用于方案文件。

4.6.2. 位置

Position 属性具有多个可编辑值和专用的 Copy/Paste 按钮。Copy 按钮会将位置数据的文本表示复制到系统剪贴板。Paste （粘贴） 按钮将从系统剪贴板中读取相同的格式，并在表单上填充属性值。

4.6.3. 部件

展开的零件。此外，还可以看到用于删除和添加零件的按钮。



属性下方是在平台或平台类型上定义的平台部分的可展开部分。这些部分还具有可编辑的属性。

通过单击与其部分关联的 X 按钮并按照提示操作，可以从平台或平台类型中删除部件。

通过滚动到窗口的最底部并选择 “Add part” 按钮，可以将新部件添加到平台或平台类型中。这将打开一个提示，询问哪种部件（传感器、武器等）、具体类型和名称。

4.6.4. 导航

在平台、平台类型或平台部分的标题部分中，将有一个链接用于导航到其各自的父项 - 即， 它源自何处 - 如果存在父级。单击该链接将在同一窗口中导航到父项。如果平台部件中的链接 ，则其在父级上的关联部分将在导航后展开。

后退/前进导航按钮将带您在您访问过的各个页面之间来回切换，就像在 Web 浏览器中一样。

5. 运动和航线

5.1. 运动组件

平台如果需要移动，就需要在其上添加运动组件，运动组件有预设的类型，如下述表格所示。**（详见文档章节：Predefined Mover Types）**

运动组件类型	运动组件含义
WSF_AIR_MOVER	空中运动组件
WSF_ARGO8_MOVER	
WSF_BRAWLER_MOVER	
WSF_FIRES_MOVER	
WSF_FORMATION_FLYER	
WSF_GROUND_MOVER	地面运动组件
WSF_GUIDED_MOVER	
WSF_HYBRID_MOVER	
WSF_INTEGRATING_SPACE_MOVER	
WSF_KINEMATIC_MOVER	
WSF_NORAD_SPACE_MOVER	
WSF_OFFSET_MOVER	跟随运动组件
WSF_P6DOF_MOVER	
WSF_PARABOLIC_MOVER	
WSF_POINT_MASS_SIX_DOF_MOVER	
WSF_RIGID_BODY_SIX_DOF_MOVER	
WSF_ROAD_MOVER	
WSF_ROTORCRAFT_MOVER	
WSF_SIX_DOF_MOVER_BASE	
WSF_SPACE_MOVER	太空运动组件
WSF_STRAIGHT_LINE_MOVER	
WSF_SUBSURFACE_MOVER	水下运动组件
WSF_SURFACE_MOVER	水上运动组件
WSF_TBM_MOVER	
WSF_TOWED_MOVER	
WSF_TSPI_MOVER	
WSF_UNGUIDED_MOVER	

其各推进器属性说明详见附录四。

以 Bomber 为例，在 Bomber 文件中添加 mover 脚本块（**详见文档章节：mover**），继承自 WSF_AIR_MOVER（**详见文档章节：WSF_AIR_MOVER**）。

```
1  # File generated by Wizard 2.9.0 on Aug 10, 2024.
2
3  platform_type Bomber WSF_PLATFORM
4    icon bomber
5
6    mover WSF_AIR_MOVER
7    end_mover
8
9  end_platform_type
```

Demo/models/platforms/air_force/Bomber

以同样的方式给 Tank 加上 WSF_GROUND_MOVER，给 Ship 加上 WSF_SURFACE_MOVER。

5.2. 航线组件

5.2.1 航线定义

平台如果需要移动，不仅仅需要运动组件，还需要航线，即平台需要往什么方向移动。

以“Tank_B01”为例，在地图上选中它，右键并选择“Route/Create Route on Tank_B01”，此时航线的定义代码被加入了“Tank_B01”的定义中。

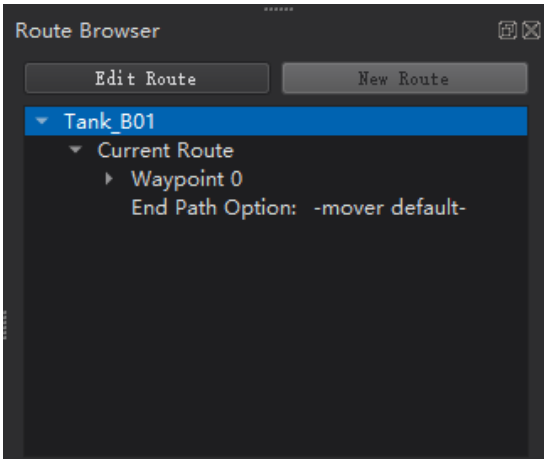
```
3  platform Tank_B01 Tank
4    position 32:57:23.207s 18:10:37.232e
5    side blue
6
7    route
8      position 32:57:23.207s 18:10:37.232e altitude 0.00 ft agl
9        speed 50 mph
10   end_route
11
12 end_platform
```

Demo/scenarios/Laydown_Blue

第 7~10 行为 route 脚本块（详见 route），第 8 行为航线第一个点的位置，第 9 行从第一个点到下一次设定速度为止期间的移动速度。

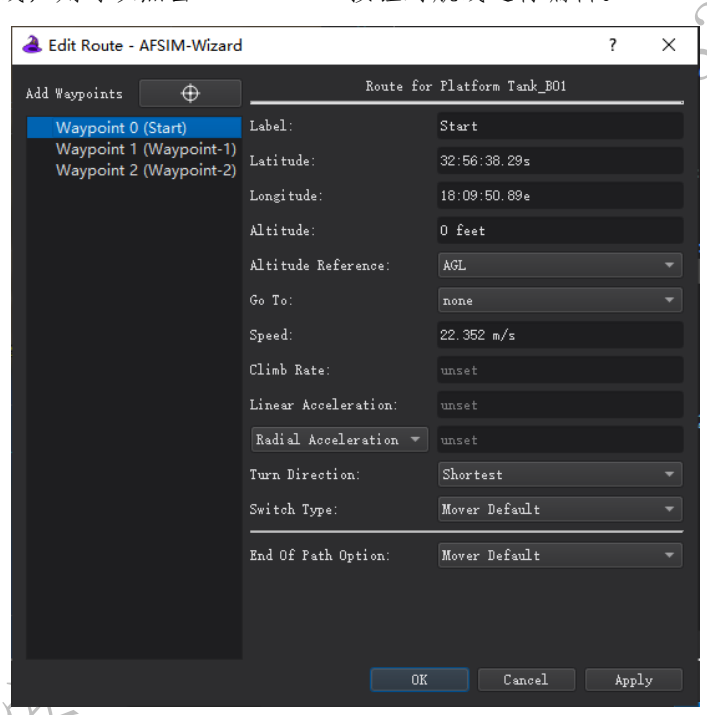
5.2.2. 航线浏览器

选中平台，在 Route Browser 中可以快速编辑航线（如果没有该面板则通过顶部菜单的 View/Route Browser 打开，如果该项前面有√表示窗口已经存在）。



如果选中的平台当前没有航线则可以点击“New Route”按钮为其添加一个航线。

如果已有航线，则可以点击“Edit Route”按钮对航线进行编辑。



在航线编辑窗口中，点击 Add Waypoints 右侧的按钮可以直接在地图上选择一个位置作为新添加的航路点（连续添加，右键结束）。

为“Tank_B01”添加两个新的航路点（第一个航路点为“Tank_B01”的初始位置）。

选中一个航路点，在右侧可以对其进行编辑。将 Waypoint 0 的 Label 改为“Start”，然后将 Waypoint 2 的 Go To 选择为 Start，点击“OK”按钮。

此时“Tank_B01”的航路将形成循环，“Tank_B01”将沿航线在三个航路点间循环移动，直至被外部打断或者模拟结束。

若 Waypoint 2 并未设定 Go to，则“Tank_B01”会在 Waypoint 2 的位置停下。

与之不同的是，如果“Bomber_B01”的末位航路点未设定 Go to，则其会沿着抵达末位航路点时的方向继续前进。

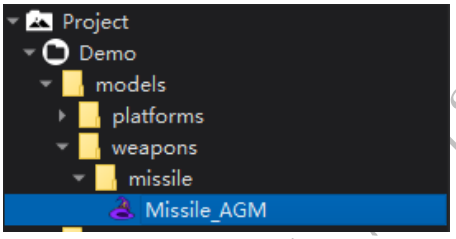
6. 武器

在 AFSIM 中，武器被分为显式武器和隐式武器，显式武器表示那些在开火后会离开发射平台的武器。本章节主要介绍显式武器。

6.1. 显式武器的定义结构

AFSIM 中的显式武器并不是只需要定义武器本身，它由多个负责不同功能的块组成。我们将以空对地导弹（AGM Missile）为例介绍这个结构。

6.1.1 新建文件



新建文件“Missile_AGM”并在 Bomber 中引用它。

Bomber 中的引用语句如图所示：

```
3   include_once ../weapons/missile/Missile_AGM
```

Demo/models/platforms/air_force/Bomber

该语句中的“..”表示忽略上级路径，从根目录往下找直到找到有目标路径的文件。

6.1.2. 定义特性

```
3   radar_signature AGM_Missile_SIG
4     constant 1 m^2
5   end_radar_signature
```

Demo/models/weapons/missile/Missile_AGM

以定义雷达特性（radar_signature）为例，上图定义了使用雷达特性“AGM_Missile_SIG”的雷达的 RCS 值为常量 1 平方米（**详见文档章节：radar_signature**）。

特性内容详见特性。

6.1.3. 定义平台类型

```
platform_type AGM_Missile_Bomb WSF_PLATFORM
  icon jdam
  radar_signature AGM_Missile_SIG

  mover WSF_GUIDED_MOVER
    aero AGM_Missile_Aero
    mass 500 lbs
    update_interval 0.5 s
  end_mover

  processor tracker WSF_PERFECT_TRACKER
    update_interval  0.5 s
  end_processor

  processor Guidance_Computer WSF_GUIDANCE_COMPUTER
    proportional_navigation_gain 10.0
    velocity_pursuit_gain 10.0
    g_bias 1.0
    maximum_commanded_g 25.0 g
    guidance_delay 0.0 sec
  end_processor

  processor fuse WSF_AIR_TARGET_FUSE
    maximum_time_of_flight 900 sec
  end_processor

end_platform_type
```

Demo/models/weapons/missile/Missile_AGM

显式武器需要定义一个平台类型，该平台类型的实例会在发射平台开火成功时被创建。

武器平台和其他平台一样需要对其图像赋值，以上图中的空对地导弹为例，它包含运动系统、追踪系统、导引系统和引信系统。

上图中的空对地导弹的运动组件采用的是“WSF_GUIDED_MOVER”，这种运动组件需要指定空气动力学模型（aero）（**详见文档章节：WSF_GUIDED_MOVER**）。

```
aero AGM_Missile_Aero WSF_AERO
  cd_zero_subsonic 0.100
  cd_zero_supersonic 0.40
  mach_begin_cd_rise 0.800
  mach_end_cd_rise 1.200
  mach_max_supersonic 2.000
  reference_area 0.059 m2
  cl_max 10.400
  aspect_ratio 4.000
end_aero
```

Demo/models/weapons/missile/Missile_AGM

为其指定的空气动力学模型如上图所示。

追踪处理器采用的是“WSF_PERFERCT_TRACKER”，并设定了刷新间隔为 0.5 秒（**详见文档章节：WSF_PERFERCT_TRACKER**）。还采用了处理器“WSF_GUIDANCE_COMPUTER”（**详见文档章节：WSF_GUIDANCE_COMPUTER**）。

引信部分采用的是“WSF_AIR_TARGET_FUSE”，并设定了最大飞行时间为 900 秒（**详见文档章节：WSF_AIR_TARGET_FUSE**）。

综上所述，武器平台类型根据其各自的应用环境需要挂载不同的组件，但是通常需要定

义模型（radar）、雷达特性（radar_signature）、处理器（processor）和引信（fuse）。

6.1.4. 定义武器效果

武器需要定义武器的效果（**详见文档章节：weapon_effects**），通常能被定义的武器效果类型如下述表格所示：

武器效果类型	武器效果含义
WSF_SPHERICAL_LETHALITY	球型爆炸面
WSF_GRADUATED_LETHALITY	
WSF_CARLTON_LETHALITY	
WSF_ENGAGE_LAUNCH_PK_TABLE_LETHALITY	
WSF_EXOATMOSPHERIC_LETHALITY	
WSF_HEL_LETHALITY	
WSF_MOBILITY_AND_FIREPOWER_LETHALITY	

我们以球型爆炸面为例，如下图，定义了一个球型爆炸面，允许附带伤害，爆炸半径为 25~30 米，爆炸伤害为 0.2~1，指数为 1（**详见文档章节：WSF_SPHERICAL_LETHALITY**）。

```
46  weapon_effects AGM_Missile_Effects WSF_SPHERICAL_LETHALITY
47    allow_incidental_damage
48    minimum_radius 25.0 m
49    maximum_radius 30.0 m
50    minimum_damage 0.1
51    maximum_damage 1.0
52    threshold_damage 0.2
53    exponent 1.0
54  end_weapon_effects
```

Demo/models/weapons/missile/Missile_AGM

6.1.5. 定义武器实例

```
56  weapon AGM_Missile WSF_EXPLICIT_WEAPON
57    launched_platform_type AGM_Missile_Bomb
58    weapon_effects AGM_Missile_Effects
59    aux_data
60      double lar_meters = 18520
61    end_aux_data
62  end_weapon
```

Demo/models/weapons/missile/Missile_AGM

如上图所示，显式武器实例需要设定武器平台类型以及武器效果（**详见文档章节：weapon**）。aux_data 脚本块中包含的是一些可供外部引用的参数，引用方式如下图所示（其中“wp”是上述武器的实例）（**详见文档章节：aux_data**）。

```
if(wp.IsValid() && wp.AuxDataExists(LAR) && PLATFORM.GroundRangeTo(track) < wp.AuxDataDouble(LAR))
{
  // Try fire and return the result
  if(wp.FireSalvo(track, bomb_once_fire))
  {
    writeln(PLATFORM.Name(), " fired ", wp, " at ", track.TargetName(), " at time ", TIME_NOW);
    return true;
  }
}
```

6.2. 武器的应用

6.2.1. 平台挂载

```
11  weapon agm AGM_Missile
12    quantity 4
13  end_weapon
```

Demo/models/platforms/air_force/Bomber

在使用时，武器将作为组件被挂载在指定平台上。

在平台上的武器脚本块中也可以对其参数进行修改，但修改的内容仅限于挂载于指定平台（或平台类型）上的对应实例。上图所示就是挂载于 Bomber 这一平台类型上的 AGM_Missile 的实例“agm”，这一实例的弹药数量为 4。

6.2.2. 开火

```
5  platform_type Bomber WSF_PLATFORM
6    icon bomber
7
8    mover WSF_AIR_MOVER
9    end_mover
10
11   weapon agm AGM_Missile
12     quantity 4
13   end_weapon
14
15   execute at_time 100 s absolute
16     Weapon("agm").Fire(MasterTrackList().TrackEntry(0));
17   end_execute
18
19 end_platform_type
```

武器的开火是一条指令（**详见文档章节：WsfWeapon**），因此需要在 execute 块或者一个脚本类中执行，调用时需要明确指定的武器实例，例如上述 Weapon 字段就需要在声明了指定名称武器的平台或者平台类型中使用，否则无效。

上图中语句的含义是在模拟第 100 秒时用武器“agm”向主追踪列表中的第一项开火。

7. 特性

1. 定义特性

特性（signature）可以被定义为空间属性（方位、仰角）、电磁属性（频率、极化）、状态。平台（platform）的特性都有以下默认值：

声学特性（acoustic_signature）	100dB-20uPa @1kHz
红外特性（inherent_contrast）	1000w/sr
光学对比度（optical_reflectivity）	0.5
光学特性（optical_signature）	1000m^2
雷达特性（radar_signature）	1000m^2

2. 内联表

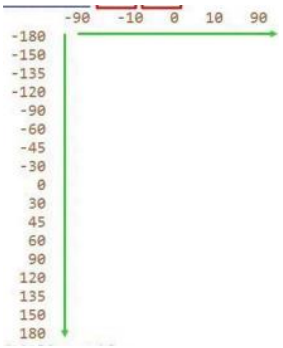
内联表（inline_table）可以用来定义复杂的特性，其通过以方位/仰角坐标系定义的点，用以确定在数据点之间的插值。

定义内联表包含单位、维度的范围（方位和仰角）、坐标系、每个坐标的数据几个元素。

```
radar_signature BOMBER_RADAR_SIG
  state default
    inline_table dbsm 24 5
          -90.0  -30.0   0.0   30.0   90.0
    -180.0  20.0  20.0   0.0  20.0  20.0
    -145.0  20.0  20.0   0.0  20.0  20.0
    -140.0  20.0  20.0  40.0  20.0  20.0
    -135.0  20.0  20.0  40.0  20.0  20.0
    -130.0  20.0  20.0   0.0  20.0  20.0
     -95.0  20.0  20.0   0.0  20.0  20.0
     -90.0  20.0  20.0  40.0  20.0  20.0
     -85.0  20.0  20.0   0.0  20.0  20.0
     -50.0  20.0  20.0   0.0  20.0  20.0
     -45.0  20.0  20.0  40.0  20.0  20.0
     -35.0  20.0  20.0  40.0  20.0  20.0
     -30.0  20.0  20.0   0.0  20.0  20.0
      30.0  20.0  20.0   0.0  20.0  20.0
      35.0  20.0  20.0  40.0  20.0  20.0
      45.0  20.0  20.0  40.0  20.0  20.0
      50.0  20.0  20.0   0.0  20.0  20.0
      85.0  20.0  20.0   0.0  20.0  20.0
      90.0  20.0  20.0  40.0  20.0  20.0
      95.0  20.0  20.0   0.0  20.0  20.0
     130.0  20.0  20.0   0.0  20.0  20.0
     135.0  20.0  20.0  40.0  20.0  20.0
     140.0  20.0  20.0  40.0  20.0  20.0
     145.0  20.0  20.0   0.0  20.0  20.0
     180.0  20.0  20.0   0.0  20.0  20.0
    end_inline_table
end_radar_signature
```

上图定义了内联表，单位为分贝平方米（dbsm），表格列数为 24、行数为 5，方位角范围为-180 度到 180 度，俯仰角的范围为-90 度到 90 度。

在指定方位角和指定俯仰角下返回的 RCS 值为对应坐标位置的值，例如方位角-180 度、俯仰角-90 度时返回 RCS 值为 20.0 分贝平方米。



方位角沿坐标系 Y 轴向下增加，俯仰角沿坐标系 X 轴向右增加。

3. 状态/频段

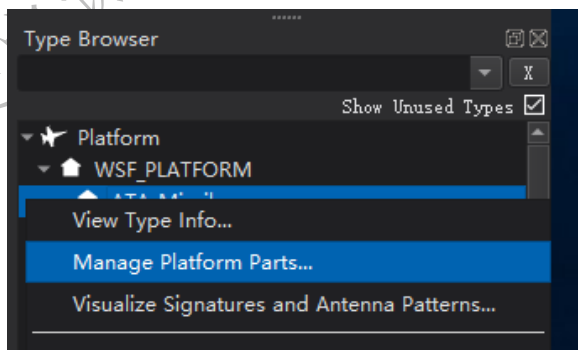
```
infrared_signature ir_states
   state default
      band medium
         constant 10 w/sr
      band default
         constant 20 w/sr
   state launching
      constant 50 w/sr
end_infrared_signature
```

可以给特性设定不同状态（state）不同波段（band）下的不同值。

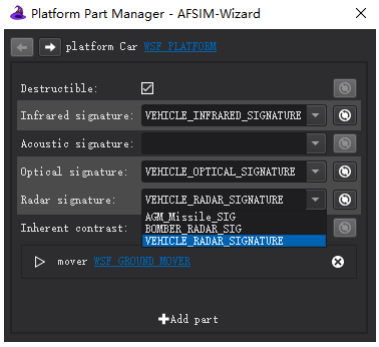
上图表示在状态”default”下中频波段的辐射强度为 10w/sr，默认波段的辐射强度为 20w/sr，状态”launching”下辐射强度为 50w/sr。

可通过脚本对状态进行设定。

4. 部件管理器

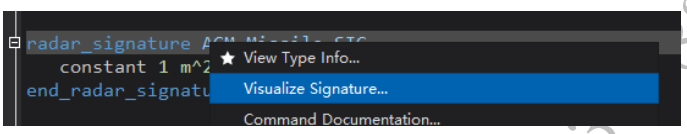


右键类型窗口中的平台类型，选择”Manage Platform Parts”可以打开部件管理器。

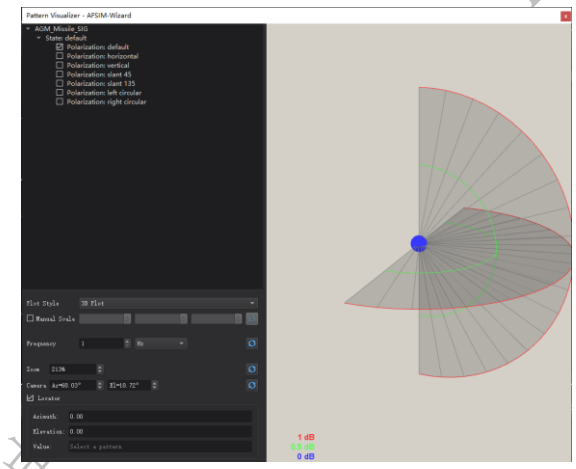


可以在部件管理器中设定平台的特性，相关代码会同步到平台类型声明的位置。

5. 模块可视化工具



右键一个特性定义的位置，选择”Visualize Signature...”，可以打开模块可视化工具。



8. 传感器

8.1. 传感器分类

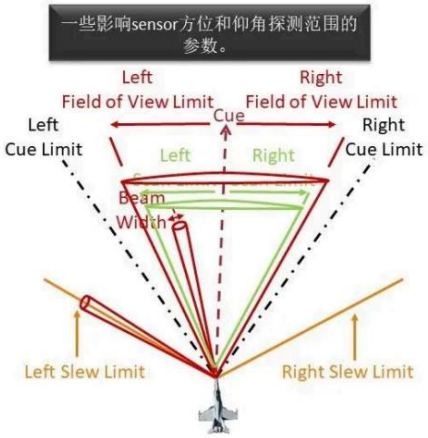
传感器（sensor）主要负责对平台感知能力的控制（**详见文档章节：Predefined Sensor Types**）。

其有大量不同类型，如下述表格所示。

类型名称	类型含义
WSF_RADAR_SENSOR	雷达传感器
WSF_ACOUSTIC_SENSOR	声学传感器
WSF_BEAM_DIRECTOR	光束指向控制器
WSF_COMPOSITE_SENSOR	复合传感器
WSF_EOIR_SENSOR	EOIR 传感器（对射式光电开关传感器）
WSF_ESM_SENSOR	电子支持测量传感器
WSF_GEOMETRIC_SENSOR	几何传感器
WSF_IRST_SENSOR	IRST 传感器（红外探测系统）
WSF_LADAR_SENSOR	Ladar 传感器（激光雷达）
WSF_LASER_DESIGNATOR（无手册）	激光指示器
WSF_LASER_TRACKER（无手册）	激光跟踪仪
WSF_NULL_SENSOR（无手册）	
WSF_OPTICAL_SENSOR	光学传感器
WSF_OTH_RADAR_SENSOR	超视距雷达传感器
WSF_PASSIVE_SENSOR	被动式传感器
WSF_SAR_SENSOR	接近式传感器
WSF_SOSM_SENSOR（无手册）	
WSF_SURFACE_WAVE_RADAR_SENSOR	声表面波雷达传感器

此处以雷达传感器为例进行讨论。

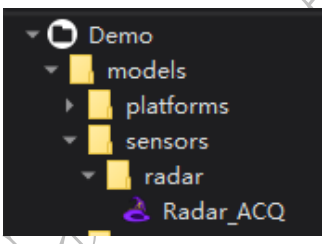
8.2. 传感器参数



- 回转限制，有时称为视域限制，定义部件相对于ECS的回转程度。
- 波束宽度
  - 天线波束中心宽度受回转限制影响，不能超过回转限制。
  - 天线波束宽度可以超过回转限制。
- 扫描限制
  - 定义了传感器模式中心左侧和右侧的搜索区域覆盖范围。
- 提示显示
  - 通常小于回转限制。
- 视域限制
  - 大于等于回转限制。
  - 覆盖扫描限制区。
  - 充当过滤器，用于忽略不在field_of_view_limits范围内的目标
    - 防止不必要的检测，让仿真运行更快。

8.3. 创建传感器

8.3.1. 新建文件



如上图所示，新建文件 Radar_ACQ。

8.3.2. 定义天线

部分传感器有天线部件的需求（antenna），需要事先定义天线类型（antenna_type），如下图所示。

（**详见文档：antenna_pattern**）

```
3  antenna_pattern ACQ_RADAR_ANTENNA
4    rectangular_pattern
5      peak_gain 35 db
6      minimum_gain -10 db
7      azimuth_beamwidth 10 deg
8      elevation_beamwidth 10 deg
9    end_rectangular_pattern
10 end_antenna_pattern
```

Demo/models/sensors/radar/Radar_ACQ

上图所示天线为矩形波束模式，增益峰值为 35 分贝，增益谷值为-10 分贝，方位波束宽度为 10 度，仰角波束宽度为 10 度。

同特性，右键天线声明的位置也可以打开其模块可视化工具。

天线主要是供给给接发装置使用。

8.3.3. 定义接发装置

传感器接收和发出信号的功能由接收装置（receiver）和发射装置（transmitter）提供，如下图所示。

（详见文档章节：receiver、transmitter）

```
transmitter
  antenna_pattern ACQ_RADAR_ANTENNA
  power 1000 kw
  frequency 3000 mhz
  internal_loss 2 db
end_transmitter

receiver
  antenna_pattern ACQ_RADAR_ANTENNA
  bandwidth 2 mhz
  noise_power -160 dbw
  internal_loss 7 db
end_receiver
```

Demo/models/sensors/radar/Radar_ACQ

上图的发射装置发射功率 1000 千瓦、工作频率 3000 兆赫、内损耗 2 分贝，接收装置带宽 2 兆赫、噪波功率-160 分贝瓦、内损耗 7 分贝。

接收装置和发射装置是作为一种组件在传感器的定义内部定义的。

8.3.4. 定义传感器

```
sensor ACQ_RADAR WSF_RADAR_SENSOR

   #Sensor Commands
   required_pd 0.5
   frame_time 10 s
   hits_to_establish_track 3 5
   hits_to_maintain_track  1 5
   reports_range
   reports_iff

   #Antenna Commands
   antenna_height 5 m
   maximum_range 150 nm
   scan_mode azimuth_and_elevation
   azimuth_scan_limits -180 deg 180 deg
   elevation_scan_limits 0 deg 50 deg

   transmitter

   receiver

   #Beam Commands
   swerling_case 1
   probability_of_false_alarm 1.0e-6
   one_m2_detect_range 50 nm
   reports_bearing
   reports_elevation
   track_quality 0.6

end_sensor
```

Demo/models/sensors/radar/Radar_ACQ

传感器的定义结构如上，除接发装置外还有大量其他的参数，根据雷达的功能需求差异

对不同的参数有不同的要求。

（详见文档章节：**Sensor**、**WSF_RADAR_SENSOR**）

8.4. 传感器的应用

新建文件 MissileLauncher、Radar_TTR、Missile_SAM。

文件内容如下：

```
# File generated by Wizard 2.9.0 on Aug 17, 2024.

include_once ../weapons/missile/Missile_SAM
include_once models/sensors/radar/Radar_ACQ
include_once models/sensors/radar/Radar_TTR
include_once models/processors/Processor_MissileLauncher

infrared_signature  VEHICLE_INFRARED_SIGNATURE
   constant  10 watts/steradian
end_infrared_signature

optical_signature  VEHICLE_OPTICAL_SIGNATURE
   constant  10 m^2
end_optical_signature

radar_signature    VEHICLE_RADAR_SIGNATURE
   constant  1 m^2
end_radar_signature

platform_type MissileLauncher WSF_PLATFORM
   icon TWIN_BOX

   infrared_signature   VEHICLE_INFRARED_SIGNATURE
   optical_signature    VEHICLE_OPTICAL_SIGNATURE
   radar_signature      VEHICLE_RADAR_SIGNATURE

   mover WSF_GROUND_MOVER
   end_mover

   zone full_kinematic
      circular
         maximum_altitude 30 km
         maximum_radius   25 nm
   end_zone

   sensor acq ACQ_RADAR
      on
      internal_link data_mgr
   end_sensor

   sensor ttr TTR_RADAR
   end_sensor

   weapon sam SAM_Missile_Launcher
      quantity 16
   end_weapon

   processor data_mgr WSF_TRACK_PROCESSOR
      purge_interval 60 seconds
   end_processor

   processor task_mgr SAMLauncher_Tactics
   end_processor

end_platform_type
```

Demo/models/platforms/army/MissileLauncher

```
# File generated by Wizard 2.9.0 on Aug 17, 2024.

antenna_pattern TTR_RADAR_ANTENNA
   sine_pattern
      peak_gain          35 dB
      minimum_gain       -10.0 db
      azimuth_beamwidth   1 deg
      elevation_beamwidth 1 deg
   end_sine_pattern
end_antenna_pattern

sensor TTR_RADAR WSF_RADAR_SENSOR

   selection_mode             multiple

   slew_mode                  azimuth_and_elevation
   azimuth_slew_limits        -180 deg 180 deg
   elevation_slew_limits      0.0 deg 80.0 deg

   mode_template
      one_m2_detect_range          35.0 nm

      antenna_height            4.0 m

      transmitter
         antenna_pattern       TTR_RADAR_ANTENNA
         power                 1000.0 kw
         frequency             9500 mhz
         internal_loss         2 db
      end_transmitter

      receiver
         antenna_pattern       TTR_RADAR_ANTENNA
         bandwidth             500.0 khz
         noise_power           -160 dBw  // will be calibra
         internal_loss         7 dB
      end_receiver

      probability_of_false_alarm    1.0e-6
      required_pd                   0.5
      swerling_case                 1

      reports_range
      reports_bearing
      reports_elevation
      reports_velocity
   end_mode_template

   mode ACQUIRE
      maximum_request_count      1

      scan_mode                  azimuth_and_elevation
      azimuth_scan_limits        -5 deg 5 deg
      elevation_scan_limits      -5 deg 5 deg

      frame_time                 2.0 sec

      hits_to_establish_track    3 5
      hits_to_maintain_track     1 3

      track_quality              0.9  // near 'weapons' grade

      on_success TRACK
   end_mode

   mode TRACK
      maximum_request_count      6

      scan_mode                  azimuth_and_elevation
      azimuth_scan_limits        -1 deg 1 deg
      elevation_scan_limits      -1 deg 1 deg
      frame_time                 1.0 sec

      hits_to_establish_track    3 5
      hits_to_maintain_track     1 3

      track_quality              1.0  // 'weapons' grade
   end_mode
end_sensor
```

Demo/models/sensors/radar/Radar_TTR

```
# File generated by Wizard 2.9.0 on Aug 10, 2024.

infrared_signature LARGE_SAM_INFRARED_SIGNATURE
   constant 1 watts/steradian
end_infrared_signature

optical_signature  LARGE_SAM_OPTICAL_SIGNATURE
   constant 1 m^2
end_optical_signature

radar_signature    LARGE_SAM_RADAR_SIGNATURE
   constant 1 m^2
end_radar_signature

platform_type SAM_Missile WSF_PLATFORM
   icon SA-10_Missile

   infrared_signature LARGE_SAM_INFRARED_SIGNATURE
   optical_signature  LARGE_SAM_OPTICAL_SIGNATURE
   radar_signature    LARGE_SAM_RADAR_SIGNATURE

   mover WSF_STRAIGHT_LINE_MOVER
      average_speed              2643.0 kts  # mach 4
      maximum_lateral_acceleration 20.0 g
      guidance_mode              lead_pursuit
   end_mover

   processor fuse WSF_AIR_TARGET_FUSE
      max_time_of_flight_to_detonate  37.0 sec  # 25 nm range
   end_processor

   processor seeker WSF_PERFECT_TRACKER
      update_interval  0.5 s
   end_processor
end_platform_type

weapon_effects LARGE_SAM_EFFECT WSF_GRADUATED_LETHALITY
   radius_and_pk  100.0 m  0.7
end_weapon_effects

weapon SAM_Missile_Launcher WSF_EXPLICIT_WEAPON
   launched_platform_type SAM_Missile
   maximum_request_count 10

   weapon_effects LARGE_SAM_EFFECT

   quantity          4

   firing_delay      0.5 sec
   salvo_interval    5.0 sec

   slew_mode         azimuth_and_elevation
   azimuth_slew_limits   -180.0 deg 180.0 deg
   elevation_slew_limits  10.0 deg  70.0 deg
end_weapon
```

Demo/models/weapons/missile/Missile_SAM

传感器 Radar_ACQ 在第一组图（右）36~39 行被使用。

其中的 internal_link 语句表示将数据关联到第 48~50 行的航迹处理器 data_mgr。

8.5. 衰减模型（attenuation_model）

attenuation_model.....end_attenuation_model

```
attenuation_model <derived-name> <base-name>
   ... Input for the attenuation model ...
end_attenuation_model
```

8.5.1. 概述

attenuation_model 用于创建配置的衰减模型，可以在发射器(transmitter)定义的 attenuation_model 块中引用。

<derived-name> 是你希望分配给你配置好的衰减模型的名称。<derived-name> 是用户希望用来引用配置好的衰减模型的名称。

<base-name> 是可用的衰减模型之一。

8.5.2. 有效使用衰减模型

衰减模型定义可以直接嵌入到雷达的定义中。例如，假设您有一个 名为“acq.txt”的文件：图中 33-34 行定义衰减模型，使用 itu 模型。

sensor/acq.txt

```
1   # File generated by Wizard 2.9.0 on Aug 21, 2024.
2   antenna_pattern ACQ_RADAR_ANTENNA
10
11  clutter_model ACQ_RADAR_CLUTTER surface_clutter
14
15  sensor ACQ_RADAR WSF_RADAR_SENSOR
16      one_m2_detect_range 50 nm
17      maximum_range 150 nm
18
19      antenna_height 10 m
20
21      frame_time 5 sec
22
23      scan_mode azimuth_and_elevation
24      azimuth_scan_limits -180 deg 180 deg
25      elevation_scan_limits 0 deg 50 deg
26
27      transmitter
28          antenna_pattern ACQ_RADAR_ANTENNA
29          power 1000 kw
30          frequency 3000 mhz
31          internal_loss 2 db
32
33          attenuation_model itu
34          end_attenuation_model
```

```
attenuation_model itu
```

这种方法的问题在于，必须修改雷达定义以改变或消除衰减模型。 在许多生产用途中，这是不可取的或不可行的。更可取的是提供“默认值” 可以覆盖的衰减模型定义。

新的“ex_radar.txt”现在将包含：

sensor/acq.txt

```
# File generated by Wizard 2.9.0 on Aug 21, 2024.
antenna_pattern ACQ_RADAR_ANTENNA

clutter_model ACQ_RADAR_CLUTTER surface_clutter

attenuation_model ACQ_RADAR_ATTENUATION itu

end_attenuation_model

#attenuation_model ACQ simple
sensor ACQ_RADAR WSF_RADAR_SENSOR
   one_m2_detect_range 50 nm
   maximum_range 150 nm

   antenna_height 10 m

   frame_time 5 sec

   scan_mode azimuth_and_elevation
   azimuth_scan_limits -180 deg 180 deg
   elevation_scan_limits 0 deg 50 deg

   transmitter
      antenna_pattern ACQ_RADAR_ANTENNA
      power 1000 kw
      frequency 3000 mhz
      internal_loss 2 db

      attenuation_model ACQ_RADAR_ATTENUATION
```

然后，要覆盖衰减模型：

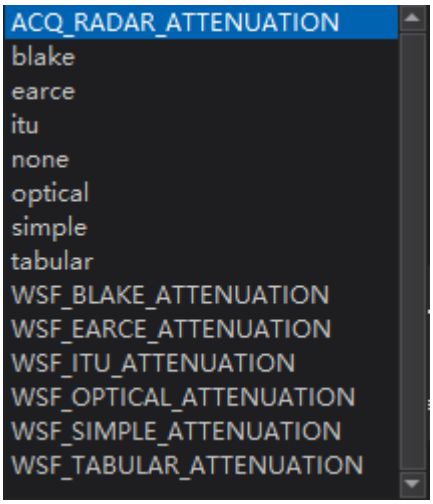
platforms/sam.txt

```
# File generated by Wizard 2.9.0 on Aug 21, 2024.
include_once sensor/acq.txt


#clutter_model ACQ_RADAR_CLUTTER none

attenuation_model ACQ_RADAR_ATTENUATION blake

end_attenuation_model
```

当雷达模型最终创建雷达实例时，它将使用 ACQ_RADAR_ATTENUATION 的最后一

个定义在模拟中。

8.5.3. 可用的衰减模型



8.5.3.1. none

一个“虚拟”衰减模型，不会导致任何效果。

```
transmitter
   antenna_pattern ACQ_RADAR_ANTENNA
   power 1000 kw
   frequency 3000 mhz
   internal_loss 2 db

   attenuation_model none
end_transmitter
```

8.5.3.2. simple

该模型提供了一种机制来指定一个常数特定的衰减(单位长度的信号损耗)或一个常数因子(衰减总是相同的)。该模型适用于具有相对几何无关的条件，且不需要计算更复杂的模型。它也适用于一些其他模型无法处理的简单情况。

可以指定以下任一选项：

specific_attenuation <value> <db-ration-unit>/<length-unit>

指定每单位长度的信号损失。这适用于路径几乎与地球平行的情况表面（例如在空对空中，参与者的高度大致相同）。

这是损耗因子（出现在分母中），通常指定为正分贝/公里（导致 值大于或等于 1）。

例：

```
specific_attenuation 0.001 db/km
```

```
attenuation_model ACQ simple
   specific_attenuation 0.001 db/km
end_attenuation_model
```

attenuation_factor

指定信号值的常数乘法器（增益因子）。因子必须在 [0 .. 1] 范围内 绝对单位（小于零的 dB 值）。此选项适用于几何图形固定（或接近固定）的情况， 例如两个地面站之间或地面站与地球同步卫星之间的通信。

例：

```
attenuation_factor -3.0 db
```

```
attenuation_model ACQ simple
   attenuation_factor -3.0 db
end_attenuation_model
```

8.5.3.3. ★itu

该模型利用国际电信联盟（ITU）下列建议书中定义的方法，确定 1-1000 GHz 频率的射频信号的衰减因子：

国际电联 ITU-R P.676-8 建议书，大气气体的衰减

国际电联 ITU-R P.838-3 建议书，用于预测方法的降雨特定衰减模型

国际电联 ITU-R P.840-4 建议书，云和雾引起的衰减

如果在 global_environment 中定义了降雨率（rain_rate），则仅计算由于降雨导致的衰减贡献。如果未指定降雨高度限制（rain_altitude_limit），则将使用云层高度限制（cloud_altitude_limits）的下限值。如果该值也未定义，则将使用 10000 米作为值。

如果在 global_environment 中定义了 cloud_altitude_limits 和 cloud_water_density，则计算云或雾引起的衰减的贡献。

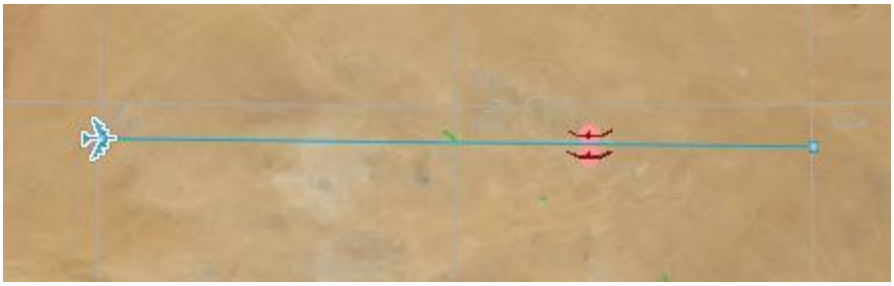
该模型通过一系列 1 公里厚的地球大气层沿路径进行积分。除非雨或云的高度限制更高，并且提供了相应的降雨率或云水密度，否则积分不会超过 30 公里（在 30 公里以上，大气气体的额外衰减可以忽略不计）。

```
attenuation_model ACQ_RADAR_ATTENUATION itu

end_attenuation_model
```

通过设置环境参数，进行测试

对环境进行修改进行测试：

假设目前有以两个雷达在地面，有一架固定翼飞机从雷达上空飞过

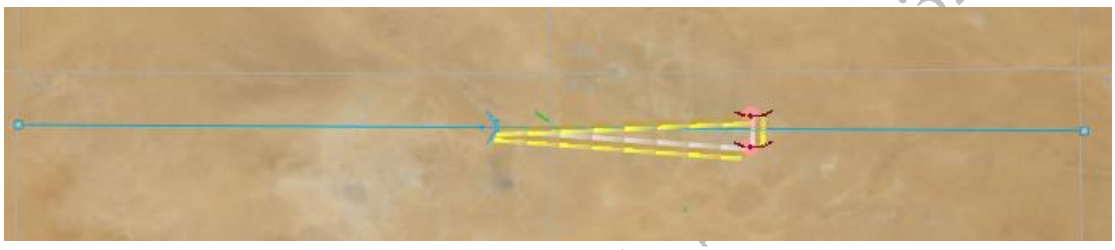


首先设置飞机飞行高度 2500m 保持不变行驶

```
platform bomber BomBer
   position 21:48:55.471n 01:59:17.665w
   altitude 2500 m
   heading 90 degrees
   route
      label Waypoint-0
      position 21:48:55.471n 01:59:17.665w altitude 2500 m
         speed 300 m/s
      label Waypoint-1
      position 21:46:46.485n 02:00:22.974e altitude 2500 m
         speed 300 m/s
   end_route
   side blue
end_platform
```

在不加环境设置（降雨率、云层高度等）并且雷达未定义衰减模型的情况下运行

雷达在此时发现飞机

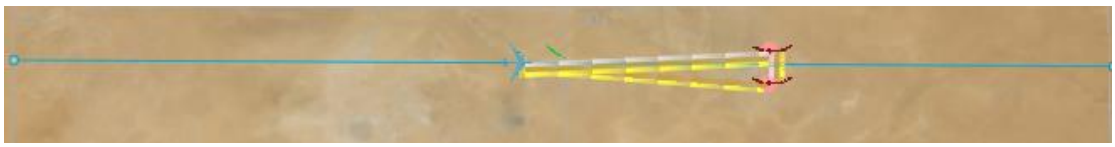


然后我们为雷达传感器添加衰减模型 itu,如图 15-17 行 37 行

```
sensor/acq.txt *
 1  # File generated by Wizard 2.9.0 on Aug 21, 2024.
 2  antenna_pattern ACQ_RADAR_ANTENNA
10
11  clutter_model ACQ_RADAR_CLUTTER surface_clutter
14
15  attenuation_model ACQ_RADAR_ATTENUATION itu
16
17  end_attenuation_model
18
19  sensor ACQ_RADAR WSF_RADAR_SENSOR
20     one_m2_detect_range 50 nm
21     maximum_range 150 nm
22
23     antenna_height 10 m
24
25     frame_time 5 sec
26
27     scan_mode azimuth_and_elevation
28     azimuth_scan_limits -180 deg 180 deg
29     elevation_scan_limits 0 deg 50 deg
30
31     transmitter
32        antenna_pattern ACQ_RADAR_ANTENNA
33        power 1000 kw
34        frequency 3000 mhz
35        internal_loss 2 db
36
37        attenuation_model ACQ_RADAR_ATTENUATION
38
```

添加衰减模型后，在不设置环境参数的情况下运行：

雷达在此时发现飞机，通过是以上测试我们可以得出，单独为雷达添加 itu 衰减模型，但是不设置环境参数，对于雷达的衰减很小或者没有影响



然后我们 Main 中添加全局环境，云层高度限制（2000m-4000m）云中水密度（1 g/m^3）降雨限制高度 2500m,降水率 2.5 mm/h

```
global_environment
    cloud_altitude_limits 2000 m 4000 m
    cloud_water_density 1 g/m^3
    rain_altitude_limit 2500 m
    rain_rate 2.5 mm/h
end_global_environment
```

设置完成后，运行一下，雷达在此时发现飞机，所以我们可以明显的看出，环境设置对于雷达的影响，那下面我们对各个参数进行细致分析

首先我们保持其余参数不变，修改 cloud_water_density(云中水密度)为 10 g/m^3

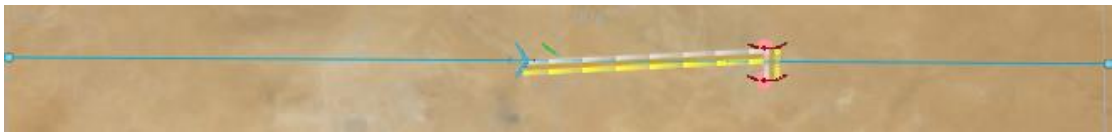
```
global_environment
    cloud_altitude_limits 2000 m 4000 m
    cloud_water_density 10 g/m^3
    rain_altitude_limit 2500 m
    rain_rate 2.5 mm/h
end_global_environment
```

运行！根据运行结果我们可以明显的看出，当云中水密度变大时，对雷达的衰减越大。

然后，我将云中水密度改回 1 g/m^3,保持参数不变，调整 rain_rate（降水率）为 10 mm/h

```
global_environment
    cloud_altitude_limits 2000 m 4000 m
    cloud_water_density 1 g/m^3
    rain_altitude_limit 2500 m
    rain_rate 10 mm/h
end_global_environment
```

运行！



我们经过对比发现并未产生太大影响，我们再次加大降水率，修改为 50mm/h

```
global_environment
    cloud_altitude_limits 2000 m 4000 m
    cloud_water_density 1 g/m^3
    rain_altitude_limit 2500 m
    rain_rate 50 mm/h
end_global_environment
```

运行！此时我们发现，雷达的性能被衰弱，所以可以得出随着降水率的增大，对雷达的衰减越大。

根据分析云层高度和降雨高度目前未发现对雷达性能有很大的影响。

8.5.3.4 ．blake

该模型利用 L.V. Blake 在海军研究实验室编写的大气吸收模型来确定衰减因子。该模型基于一系列 42 条衰减曲线，这些曲线适用于 100 MHz 至 10 GHz 的频率和 0 至 10 度的仰角。这些曲线在 300 海里之外是平坦的。这些表格发表在《雷达系统分析，第 15.1 节，David K. Barton，Artech 出版社》。

注意：这种选择仅在发射机或目标位于（或非常接近）地表时有效。

```
attenuation_model ACQ_RADAR_ATTENUATION blake

end_attenuation_model
```

8.5.3.5. earce

该模型利用来自 ESAMS/ALARM/RADGUNS 共同环境（EARCE）的大气吸收模型来确定衰减因子。这是一系列预先计算好的表格，适用于 100 MHz 至 18 GHz 和 27 GHz 至 40 GHz 范围内的频率。低于 100 MHz 的频率将假设为 100 MHz。在 18 GHz 至 27 GHz 之间以及高于 40 GHz 的频率将使用一种计算量非常大的方法来确定衰减，应避免使用。

注意：这种选择仅在发射机或目标位于（或非常接近）地表时有效。

```
attenuation_model ACQ earce
end_attenuation_model
```

8.5.3.6. WSF_OPTICAL_ATTENUATION

WSF_OPTICAL_ATTENUATION 衰减模型用于视觉和红外系统，如 **WSF_EOIR_SENSOR** 和 **WSF_IRST_SENSOR.**

8.6. 杂波模型（clutter model）

8.6.1、概念

"clutter_model" 指的是用于模拟和生成杂波信号的模型，这些信号可以被用于 WSF_RADAR_SENSOR（一种雷达传感器模型）的定义中。杂波是指雷达接收器接收到的来自非目标物体（如地面、建筑物、植被等）的反射信号，它们可能会干扰对真实目标的检测。

8.6.2、使用方式

clutter_model ...end_clutter_model

```
clutter model <derived-name> <base-name>
   ... Input for the clutter model ...
end_clutter_model
```

以下是有关 "clutter_model" 的几个要点：

<derived-name> 是用户希望用于引用配置好的杂波模型的名称。这是一个自定义标签，用户在定义雷达传感器时可以使用它来指定应该使用哪个杂波模型。

<base-name> 是可用杂波模型的名称之一，它指定了要使用的基础杂波模型类型。这些基础模型可能是预定义的，并且具有特定的特性，用于模拟不同类型的杂波环境。

8.6.3、有效使用杂波模型

杂波模型定义可以直接嵌入到雷达的定义中。假如您有一个文件称为“acq.txt”:图中 37-39 行直接嵌入到雷达的定义中

sensor/acq.txt

```
# File generated by Wizard 2.9.0 on Aug 21, 2024.
antenna_pattern ACQ_RADAR_ANTENNA
   rectangular_pattern
   end_rectangular_pattern
end_antenna_pattern

sensor ACQ_RADAR WSF_RADAR_SENSOR
   one_m2_detect_range 50 nm
   maximum_range 150 nm

   antenna_height 10 m

   frame_time 5 sec

   scan_mode azimuth_and_elevation
   azimuth_scan_limits -180 deg 180 deg
   elevation_scan_limits 0 deg 50 deg

   transmitter

   receiver

   clutter_model surface_clutter

   end_clutter_model
```

这种方法的问题在于必须修改雷达定义来改变或消除杂波模型。在许多生产用途中，这是不希望的或不可行的。更理想的是提供一个可以被覆盖的“默认”杂乱模型定义。

新的“ex_radar.txt”现在将包含：

sensor/acq.txt

```
# File generated by Wizard 2.9.0 on Aug 21, 2024.
antenna_pattern ACQ_RADAR_ANTENNA
   rectangular_pattern
   end_rectangular_pattern
end_antenna_pattern

clutter_model ACQ_RADAR_CLUTTER surface_clutter
end_clutter_model

sensor ACQ_RADAR WSF_RADAR_SENSOR
   one_m2_detect_range 50 nm
   maximum_range 150 nm

   antenna_height 10 m

   frame_time 5 sec

   scan_mode azimuth_and_elevation
   azimuth_scan_limits -180 deg 180 deg
   elevation_scan_limits 0 deg 50 deg

   transmitter

   receiver

   clutter_model ACQ_RADAR_CLUTTER
```

然后，要覆盖杂波模型（可以覆盖杂波模型）：

platforms/sam.txt

```
# File generated by Wizard 2.9.0 on Aug 21, 2024.
include_once sensor/acq.txt


clutter_model ACQ_RADAR_CLUTTER none

end_clutter_model

```

当雷达模型最终在仿真中创建雷达实例时，它将使用 ACQ_RADAR_CLUTTER 的最后一个定义。

8.6.4、可用的杂波模型

8.6.4.1. none

一个“虚拟”杂波模型，等同于没有杂波。

```
clutter_model ACQ_RADAR_CLUTTER none

end_clutter_model
```

8.6.4.2. surface_clutter_table

Surface_clutter_table 用表表示杂乱，该表通常由 sensor_plot 生成。该表包含杂波数据作为目标高度和目标距离的函数。此外，如果该表是特定于站点的，它还将包含作为目标方位函数的数据。

与站点无关的杂乱表具有以下形式：

```
clutter_model <derived-name> WSF_SURFACE_CLUTTER_TABLE
   clutters
      altitude <length-value>
        range <length-value>  clutter <power-value>
        ...
        range <length-value>  clutter <power-value>
      altitude <length-value>
        range <length-value>  clutter <power-value>
        ...
        range <length-value>  clutter <power-value>
        ...
      altitude <length-value>
        range <length-value>  clutter <power-value>
        ...
        range <length-value>  clutter <power-value>
      ...
   end_clutters
end_clutter_model
```

特定于站点的表在形式上与此类似，不同之处在于范围集包含方位角数据，如下所示：

```
clutter_model <derived-name> WSF_SURFACE_CLUTTER_TABLE
   clutters
      altitude <length-value>
         bearing <angle-value>
           range <length-value>  clutter <power-value>
           ...
           range <length-value>  clutter <power-value>
           ...
         bearing <angle-value>
           // Full set of ranges:
           range <length-value>  clutter <power-value>
           ...
           range <length-value>  clutter <power-value>
           ...
         ...
      altitude <length-value>
         bearing <angle-value>
           range <length-value>  clutter <power-value>
           ...
           range <length-value>  clutter <power-value>
           ...
         ...
      ...
   end_clutters
end_clutter_model
```

8.6.4.3. alarm

ALARM 杂波模型利用了 ALARM 模型的高保真模型。

8.6.4.4. ★surface_clutter

这是一个计算杂波功率的算法模型。它使用来自 global-environment 的参数来确定陆地形状、陆地覆盖范围和海洋状态。

```
clutter_model <derived-name> surface_clutter
end_clutter_model
```

示例：

```
clutter_model ACQ_RADAR_CLUTTER surface_clutter
end_clutter_model
```

use_legacy_data<bool>

指定是否使用一组较旧的杂波强度表。

默认禁用

9. 处理器

9.1. 处理器的分类

处理器（processor）主要管理负责对平台行为的控制。

其有大量不同基本类型（**详见文档章节：Predefined Processor Types**），如下述表格所示。各类型处理器命令详见附录五（处理器）。

类型名称	类型含义
WSF_TASK_PROCESSOR	
WSF_AIR_TARGET_FUSE	
WSF_ASSET_MANAGER	
WSF_BATTLE_MANAGER	
WSF_BRAWLER_PROCESSOR	
WSF_COHERENT_SENSOR_PROCESSOR	
WSF_DELAY_PROCESSOR	
WSF_DIRECTION_FINDER_PROCESSOR	
WSF_DISSEMINATE_C2	
WSF_EXCHANGE_PROCESSOR	
WSF_FUSION_CENTER	
WSF_GROUND_TARGET_FUSE	
WSF_GUIDANCE_COMPUTER	
WSF_IMAGE_PROCESSOR	
WSF_INTERSECT_PROCESSOR	
WSF_LINK16_COMPUTER	
WSF_LINKED_PROCESSOR	
WSF_LINKED_SCRIPT_PROCESSOR	
WSF_MESSAGE_PROCESSOR	
WSF_MOVE_PLAN_PROCESSOR	
WSF_ORBITAL_CONJUNCTION_PROCESSOR	
WSF_P6DOF_GUIDANCE_COMPUTER	
WSF_PERCEPTION_PROCESSOR	
WSF_PERFECT_TRACKER	
WSF_QUANTUM_TASKER_PROCESSOR	
WSF_PIPR_PROCESSOR	

类型名称	类型含义
WSF_SA_PROCESSOR	
WSF_SCRIPT_PROCESSOR	允许开发者通过代码对平台的行为进行定制化的处理。
WSF_SENSORS_MANAGER	
WSF_SENSORS_MANAGER_FOV	
WSF_SIMPLE_SENSORS_MANAGER	
WSF_SIX_DOF_GUIDANCE_COMPUTER	
WSF_STATE_MACHINE	
WSF_THREAT_PROCESSOR	
WSF_TRACK_CLASSIFIER	
WSF_TRACK_PROCESSOR	
WSF_TRACK_STATE_CONTROLLER	
WSF_TRIMSIM_PROCESSOR	
WSF_UNCLASS_ASSET_MANAGER	
WSF_UNCLASS_BM	
WSF_UNCLASS_DISSEMINATE_C2	
WSF_UPLINK_PROCESSOR	
WSF_VIDEO_PROCESSOR	
WSF_WEAPONS_MANAGER	
WSF_WEAPONS_MANAGER_AI	
WSF_WEAPONS_MANAGER_SAM	
WSF_WEAPON_FUSE	
WSF_WEAPON_SERVER_PROCESSOR	
WSF_WEAPON_THREAT_PROCESSOR	
WSF_WEAPON_TRACK_PROCESSOR	

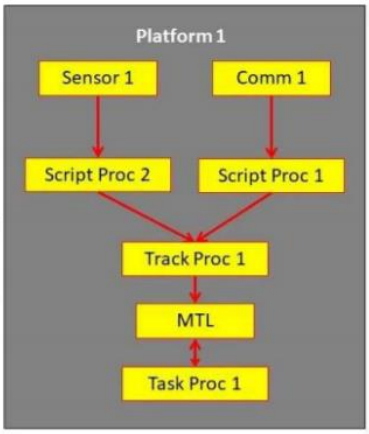
9.2. 平台追踪逻辑流程

平台的处理器通常包含两种，轨迹处理器（track processor）和任务处理器（task processor），以 8.4 中第一组图中的“data_mgr”和“task_mgr”为例，其中 data_mgr 是轨迹处理器，而 task_mgr 是任务处理器。

由传感器捕获到指定平台后产生追踪轨迹（track），也可以通过通信组件（comm）从外部接收到追踪轨迹。

接收到的轨迹将被递交给轨迹处理器，轨迹处理器将 track 放入平台的主轨迹表（MTL）中，此时任务处理器可以从 MTL 中获取到轨迹并进行对应的行为。

- sensor产生track
- comm通信设备接收外源track
- 两者发送给processor
- 这些信息在到达processor之前也同时发送给其它processor
  - *on_message块可以捕获这些消息*
  - 捕捉后我们可以修改、延迟或终止该消息



- 当消息到达processor时会触发一个脚本块。
  - comm, sensor, 和其它processor的消息通过“internal_link”来发送
- 可以根据消息类型来设置发送条件

“WSF_SCRIPT_PROCESSOR” 帮助文档

Type String	Script Class
WSF_ASSOCIATION_MESSAGE	WsfAssociationMessage
WSF_CONTROL_MESSAGE	WsfControlMessage
WSF_IMAGE_MESSAGE	WsfImageMessage
WSF_STATUS_MESSAGE	WsfStatusMessage
WSF_DROP_TRACK_MESSAGE WSF_TRACK_DROP_MESSAGE (See note below)	WsfTrackDropMessage
WSF_TRACK_MESSAGE	WsfTrackMessage
WSF_TRACK_NOTIFY_MESSAGE	WsfTrackNotifyMessage
WSF_VIDEO_MESSAGE	WsfVideoMessage

Type String	Script Class
WSF_TASK_ASSIGN_MESSAGE	WsfTaskAssignMessage
WSF_TASK_CANCEL_MESSAGE	WsfTaskCancelMessage
WSF_TASK_CONTROL_MESSAGE	WsfTaskControlMessage
WSF_TASK_STATUS_MESSAGE	WsfTaskStatusMessage
WSF_TASK_ASSIGN_MESSAGE	WsfTaskAssignMessage
WSF_DROP_TRACK_MESSAGE WSF_TRACK_DROP_MESSAGE	WsfTrackDropMessage

- 可以有许多“type”
- “default” (可选) 在消息不匹配任何类型时使用
  - 类似于switch/case的default声明
- 看看下图消息使用方法

```
on_message
   type WSF_TRACK_MESSAGE
      script
         WsfTrackMessage trackMsg = (WsfTrackMessage)MESSAGE;
         writeln(TIME_NOW, " -- Received track: ", trackMsg.Track().TrackId().ToString());
      end_script
   default
      script
         writeln(TIME_NOW, " -- Received message of type: ", MESSAGE.Type());
      end_script
end_on_message
```

- 我们可以将WsfMessage (MESSAGE) 转换为传入类型
  - 这将允许我们访问该类型的脚本方法

10. 脚本

10.1. 脚本块

AFSIM 的脚本有着特殊的语言结构，如下图所示，脚本使用的变量需要在 script_variables 脚本块中被定义。（**详见文档章节：Script Variables**）

```
script_variables
   double range = 20.0 * Math.M_PER_NM();
   int bomb_once_fire = 2;
   string weapon = "sam";
end_script_variables
```

脚本方法则需要在 script 脚本块中定义。（**详见文档章节：script**）

```
// Check if the engagement will happen
// return <- can engage
script bool CanEngage()
{
   bool result = false;
   WsfWeapon wp = PLATFORM.Weapon(weapon);
   // Check if weapon and target are avaliable
   if(!wp.IsValid() || !TRACK.Target().IsValid())
   {
      return false;
   }
   // Check if weapon needs to fire
   if(!(PLATFORM.GroundRangeTo(TRACK) < range)
   || !(WeaponsActiveFor(TRACK.TrackId()) < 1))
   {
      return false;
   }
   // Check if bomb's quantity is enough
   if(!(wp.QuantityRemaining() > bomb_once_fire))
   {
      return false;
   }

   return true;
}
end_script
```

10.2. 方法的使用

即使是已定义的方法也需要在生命周期内或者其他方法内使用。

已实现通用脚本接口的 AFSIM 组件有：simulation，platform，processor，weapon

接口元素：脚本变量块、脚本方法块、脚本指令块、通用脚本块（execute、on_initialize、on_initialize2、on_update、on_message）、预定义变量。

```
46    update_interval 3 s
47    on_update
48      // Find the target which has not been attacked
49      for(int i = 0; i < PLATFORM.MasterTrackList().Count(); i+= 1)
50      {
51        if(!tgt[i])
52        {
53          Drawer_Processor_Bomber_Weapon();
54          // Try attack
55          WsfTrack track = PLATFORM.MasterTrackList().TrackEntry(i);
56          tgt[i] = fire(track, 2);
57        }
58      }
59    end_on_update
```

如上图所示，“Drawer_Processor_Bomber_Weapon”方法在 on_update 中被调用了，同时规定了 update_interval 为 3 秒，这意味着 on_update 中的脚本逻辑将会每 3 秒执行一次。

```
execute at_time 2 sec absolute
   print("HELLO WORLD");
end_execute
```

上图则表示在模拟开始的第 2 秒将会在控制台输出“HELLO WORLD”字样。

10.3. 常用脚本内容

10.3.1. 静态类型

常用的静态类型有如下几种：

类型	类型含义
__BUILTIN__	这个静态类是嵌入框架的，可以直接使用其下的方法。
Earth	提供物理系统相关的函数和变量
Math	提供数学相关的函数和变量
Moon	提供月球数据相关的函数和变量
Sun	提供太阳数据相关的函数和变量
System	用于对系统底层函数进行操作
WsfSimulation	用于获取模拟流程中的实例

10.3.2. Math 类

Math 类常用函数如下：

函数	返回值类型	返回值含义
PI()	double	Π
PI_OVER_2()	double	Π/2
M_PER_FT()	double	每英尺多少米
M_PER_MI()	double	每英里多少米
M_PER_NM()	double	每海里多少米
FT_PER_M()	double	每米多少英尺
Pow(double x, double y)	double	x^y
Floor(double num)	double	向下取整
Ceil(double num)	double	向上取整
Fmod(double x, double y)	double	x%y

10.3.3. 变量类型

数值类型：

类型名称	类型含义
int	整型
bool	布尔值
double	浮点数
string	字符串

引用类型：

类型名称	类型含义
Array	数组
Map	字典
Set	列表
Vec3	三维向量
QuadTree	四叉树
Calender	时间
Signal	信号（类同委托）
Struct	结构体
FileIO	文件读写

10.3.4. 关键字和操作符

关键字	关键字含义	用例
extern	表示变量在块的外部声明	extern int A

操作符	操作符含义	用例
->	用于访问变量	A -> x

10.3.5. 特定范围变量

变量名称	变量含义
PLATFORM	当前平台
WEAPON	当前武器
PROCESSOR	当前处理器
TRACK	状态机中正在处理的当前轨迹
MESSAGE	on_message 块中当前消息

10.4. 平台的生命周期

01 -> 从输入文件中初始化位置（LLA 考虑地形，NED 考虑姿态）

02 -> 计算坐标系，ECI 坐标系需要考虑地球偏转角

03 -> 计算所有 group（可选）

04 -> 执行 on_initialize 函数

05 -> signature, height, length, width, mass（empty, fuel, payload）（可选）

06 -> 处理指挥链通信

07 -> 组件 track manager 初始化

08 -> 组件 mover 初始化

09 -> 组件 fuel 初始化（可选）

10 -> 计算导航偏差（可选）

11 -> platform 部件初始化

12 -> 执行 on_initialize2 函数

10.5. 状态机

10.5.1. 状态机

```
evaluation_interval detected 10 s
state detected
  next_state ignore
    return (TRACK.IFF_Friend());
  end_next_state
  next_state engage
    return CanEngage();
  end_next_state
end_state

 evaluation_interval engage 10 s
state engage
  next_state wait
    return Fire(TRACK, "FIRE", weapon, bomb_once_fire, PLATFORM);
  end_next_state
end_state

evaluation_interval wait 10 s
state wait
  next_state engage
  end_next_state
end_state

evaluation_interval ignore 24 hr
state ignore
end_state
```

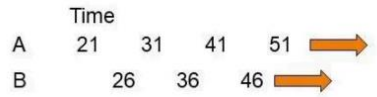
状态机（state）通常基于 Track 来执行，如上图中的状态机位于某个 WSF_TASK_PROCESSOR 内，通过 TaskManager 获取到 Track。

状态机中的状态有先后顺序，根据 state -> next state 的顺序可以排列成状态链，在满足 ne

xt state 块中的条件时跳转到该状态。

如上图所示，状态机初始进入”detected”状态，随即进行判断，当 TRACK.IFF_Friend()返回 true 时，跳转到”ignore”状态，否则进行后续判断，当 CanEngage()返回 true 时，跳转到”engage”状态，否则根据 evaluation_interval detected 10 s 语句等待 10s 后再次进入初始状态。

10.5.2. 执行顺序



不同 Track 的状态机之间相互独立。

如上图所示，Track_A 在 21s 时进入初始状态，Track_B 在 26s 时进入初始状态，初始状态的 evaluation_interval 为 10s，如果约定 45s 时某个条件达成使状态机可以跳出至下一个状态，那么，由于 A 在 21s 进入，则 45s 后 A 重新进入初始状态的最早时间为 51s，B 在 26s 进入，则 45s 后 B 重新键入初始状态的最早时间为 46s，因而 B 比 A 先跳出至下一个状态。

10.6. 观察者

```
script void SimulationComplete()
  writeln("Simulation Complete");
end_script

observer
  enable SIMULATION_COMPLETE
end_observer
```

观察者（observer）的用法为在“observer”块中对具体的观察事件进行启用，当相应观察事件被触发时，模拟器会调用其对应的接口。

以上图为例，观察者启用了“SIMULATION_COMPLETE”，这将使得函数“SimulationComplete”在模拟结束时被调用，将会在控制台输出“Simulation Complete”字段。

（**详见文档章节：observer**）

11. 指挥链

11.1. 指挥链使用方式

指挥链（Command Chain）有两种使用方式：指定平台指挥官、指定指挥链指挥官。

指挥链由指挥官（commanders）、下级（subordinates）和同级（peers）组成，每个指挥链必须有一个指挥官（可以是（自己）SELF）

如果指挥官不存在，不会产生编译错误，WIZARD 将标记平台名称，但程序将继续执行。

11.1.1. 指定平台指挥官

commander<commander-name>

直接指定平台的指挥官。配合以下这种方式进行通信

```
processor data_mgr WSF_TRACK_PROCESSOR
   report_to commander via blue_comm
end_processor
```

示例：平台 jet_2 指定指挥官为 sam_1，此时 sam_1 成为 jet_2 的指挥官，故 sam_1 可以使用找寻下属的方式发送报告：

```
platform jet_2 jet_fighter
   position 26:28:21.147s 49:33:16.627w
   side blue
   altitude 30000 feet
   heading 45 degrees
   route
      position 26:28:21.15s 49:33:16.63w altitude 30000.00 ft agl
         speed 500 mph
      position 26:23:51.59s 48:53:40.95w
   end_route
   add comm blue_comm1 TEAM_DATALINK
      network_name blue_net1
      internal_link data_mgr
      internal_link task_mgr
   end_comm
   commander sam_1
end_platform
```

```
platform sam_1 SINGLE_LARGE_SAM
   position 26:21:21.53s 49:25:20.23w
   side blue
   heading 90 degrees

   processor data_mgr
      report_to subordinates via blue_comm1
   end_processor

   add comm blue_comm1 TEAM_DATALINK
      network_name blue_net1
      internal_link data_mgr
      internal_link task_mgr
   end_comm
end_platform
```

11.1.2. 指定指挥链指挥官

command_chain<command-chain-name> <commander-name>

为一个平台添加指挥链指挥官，即加入一个指挥链中并指定指挥官。

示例：平台 sam_1 使用 command_chain 方式创建并加入指挥链 PIEATE 并指定指挥官为自己，平台 jet_2 使用 command_chain 方式加入指挥链 PLEATE 并指定指挥官为 sam_1

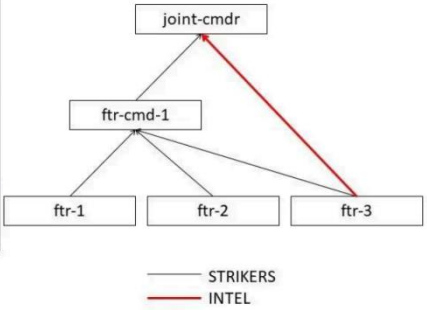
```
platform sam_1 SINGLE_LARGE_SAM
    position 26:21:21.53s 49:25:20.23w
    side blue
    heading 90 degrees

    processor data_mgr
       report_to command_chain PIEATE subordinates via blue_comm1
    end_processor

    add comm blue_comm1 TEAM_DATALINK
       network_name blue_net1
       internal_link data_mgr
       internal_link task_mgr
    end_comm
    command_chain PIEATE SELF
end_platform
```

```
platform jet_2 jet_fighter
    position 26:28:21.147s 49:33:16.627w
    side blue
    altitude 30000 feet
    heading 45 degrees
    route
       position 26:28:21.15s 49:33:16.63w altitude 30000.00 ft agl
          speed 500 mph
       position 26:23:51.59s 48:53:40.95w
    end_route

    add comm blue_comm1 TEAM_DATALINK
       network_name blue_net1
       internal_link data_mgr
       internal_link task_mgr
    end_comm
    command_chain PIEATE sam_1
end_platform
```

11.1.3. 多条指挥链

一个平台可以存在于多个指挥链中，一个平台可以有多个指挥官（在不同的链上）

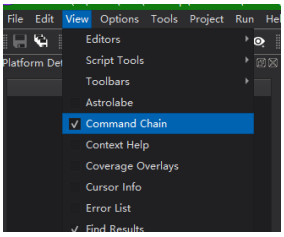
```
platform joint-cmdr COMMAND_POST
   command_chain INTEL SELF
   command_chain STRIKERS SELF
end_platform
platform ftr-cmd-1 COMMAND_POST
   command_chain STRIKERS joint-cmdr
end_platform
platform ftr-1 F-18
   command_chain STRIKERS ftr-cmd-1
end_platform
platform ftr-2 F-18
   command_chain STRIKERS ftr-cmd-1
end_platform
platform ftr-3 F-18
   command_chain STRIKERS ftr-cmd-1
   command_chain INTEL joint-cmdr
end_platform
```



图上指挥链为：joint-cmdr 创建两条指挥链 INIEL 和 STRIKERS 指挥官都为子级，ftr-cmd-1 在 STRIKERS 中指挥官为 joint-cmdr，ftr-1、ftr-2、ftr-3 均在 STRIKERS 中指挥官为 ftr-cmd-1，另外 ftr-3 还有另一条指挥链 INIEL 指挥官为 joint-cmdr。

11.2. 指挥链浏览器

打开指挥链浏览器（Command Chain Browser）：打开 View 下拉列表勾选 Command Chain 即可



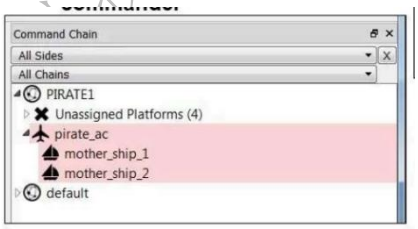
示例：

```
platform pirate_ac SOCATA_TB9
  side red
  command_chain PIRATE1 SELF
  icon CRW
```

```
platform mother_ship_1 MOTHER_SHIP
  side red
  command_chain PIRATE1 pirate_ac
  icon Luyang
```

```
platform mother_ship_2 MOTHER_SHIP
  side red
  command_chain PIRATE1 pirate_ac
  icon Luyang
```

以上指挥链三个平台都有相同的指挥链 PIRATE1，Pirate_ac 的指挥官是自己（SELF），Pirate_ac 是指挥链顶端，Mother_ship 隶属于 Pirate_ac 指挥官，他们在指挥链浏览器上显示如下：



11.3. 平台默认值

指挥官名称	SELF（此标识标识此平台）
指挥链名称	“default”

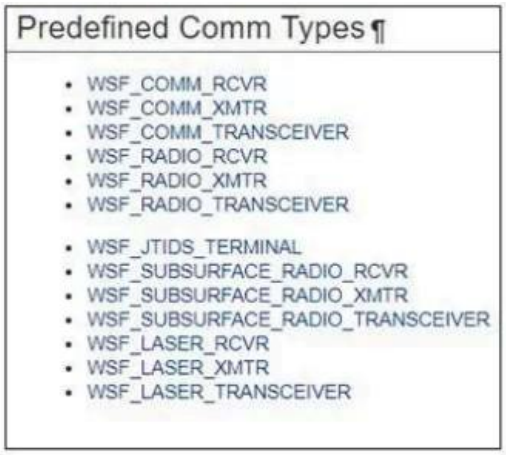
11.4. 通信

11.4.1. 预定义的通信设备

COMM	表示完美的（有线）通讯设备
JTIDS TERMINAL	是内置的 Link-16 设备

RADIO	是无线电设备，仅限于视距内通讯
SUBSURFACE	启用与地表以下设备的通讯
TRANSCEIVER	是双向的
RCVE	仅接收
XMTR	仅发送

11.4.2. 预定义的通信类型



11.4.3. 定义通信设备

定义通信设备示例：

首先定义一个通信设备 TEAM_DATALINK

```
comm TEAM_DATALINK WSF_COMM_TRANSCEIVER
   //link_protocol      csma
   transfer_rate      100 mbits/sec
end_comm
```

现有两个平台：single_large_sam 和 jet_fighter,他们都继承了 TEAM_DATALINK 定义了各自的通信设备 blue_comm1,并将 network_name 设置为 blue_net1

```
comm blue_comm1 TEAM_DATALINK
   network_name blue_net1
   internal_link data_mgr
   internal_link task_mgr
end_comm
```

```
comm blue_comm1 TEAM_DATALINK
   network_name blue_net1
   internal_link data_mgr
   internal_link task_mgr
end_comm
```

也可以通过以下方式定义通信：

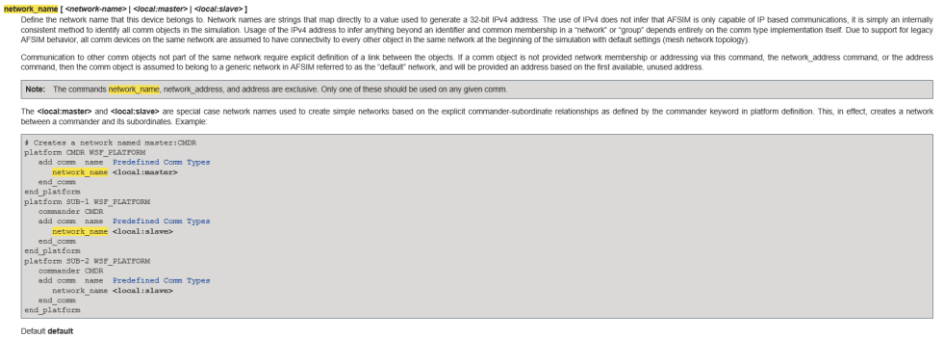
```
add comm blue_comm1 TEAM_DATALINK
   network_name blue_net1
   internal_link data_mgr
   internal_link task_mgr
end_comm
```

通讯设备是平台的组成部分。

相同的 network_name 表示这些通讯设备默认情况下可以相互通信。与平台上其他部件建立链接，否则消息无法传递

11.5. 简易指挥官从属网络

11.5.1 从属网络



network_name [<network-name> | <local:master> | <local:slave>]

Define the network name that this device belongs to. Network names are strings that map directly to a value used to generate a 32-bit IPv4 address. The use of IPv4 does not infer that AFSIM is only capable of IP based communications, it is simply an internally consistent method to identify all comm objects in the simulation. Usage of the IPv4 address to infer anything beyond an identifier and common membership in a "network" or "group" depends entirely on the comm type implementation itself. Due to support for legacy AFSIM behavior, all comm devices on the same network are assumed to have connectivity to every other object in the same network at the beginning of the simulation with default settings (mesh network topology).

Communication to other comm objects not part of the same network require explicit definition of a link between the objects. If a comm object is not provided network membership or addressing via this command, the network_address command, or the address command, then the comm object is assumed to belong to a generic network in AFSIM referred to as the "default" network, and will be provided an address based on the first available, unused address.

Note: The commands network_name, network_address, and address are exclusive. Only one of these should be used on any given comm.

The **<local:master>** and **<local:slave>** are special case network names used to create simple networks based on the explicit commander-subordinate relationships as defined by the commander keyword in platform definition. This, in effect, creates a network between a commander and its subordinates. Example:

```
# Creates a network named master:CMDR
platform CMDR WSF_PLATFORM
   add comm  name  Predefined Comm Types
      network_name <local:master>
   end_comm
end_platform
platform SUB-1 WSF_PLATFORM
   commander CMDR
   add comm  name  Predefined Comm Types
      network_name <local:slave>
   end_comm
end_platform
platform SUB-2 WSF_PLATFORM
   commander CMDR
   add comm  name  Predefined Comm Types
      network_name <local:slave>
   end_comm
end_platform
```

Default **default**

< local:master >和< local:slave >是一个特殊的案例网络名称,用于创建简单的网络,基于在平台定义中由指挥官关键字定义的显式指挥官从属关系。实际上,这创造了一个指挥官和它的下属之间的网络。

不要在同一个平台上使用默认的和“命名”的网络

11.5.2. 示例

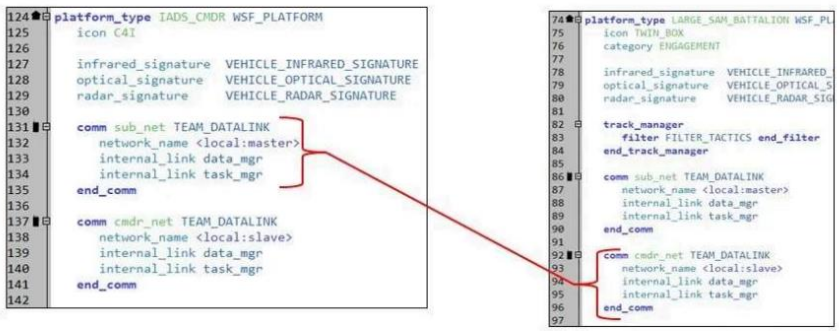
*使用<local:master>和<local:slave>用作 network_name 会创建一个依托于已定义指挥链的网络

*当通信设备包含在许多场景的文件中时，功能强大

*< local:master > = "我是这个数据链的指挥官"

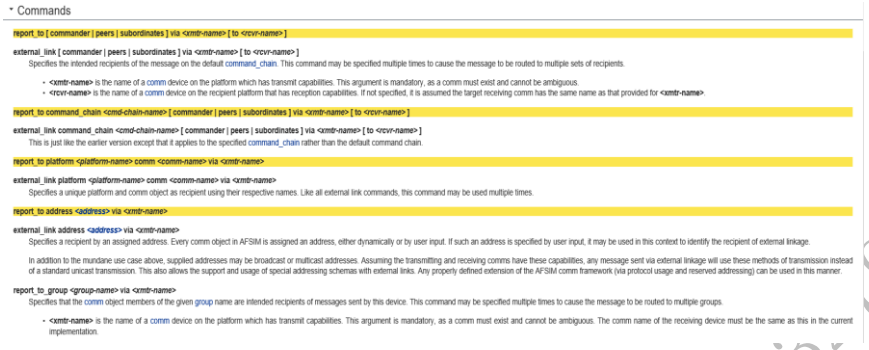
*< local:slave> = "我是这个数据链的下属"

*使用< local:master >和< local:slave >创建网络时，在网络上最多容纳 255 个



11.6. 报告指令

11.6.1. 报告命令的几种使用方式



report_to [commander | peers | subordinates] via <xmtr-name> [to <rcvr-name>]

指挥官（如果没有指定，“default”将被使用）

收件者（指挥官、同级或下属）[commander | peers | subordinates]

发送通信（期望接收通信的名称以匹配发送名称）<xmtr-name>

接收通讯（可选）[to <rcvr-name>]

需要脚本和跟踪处理器,但不需要任务处理器

< xmtr-name >是一个通信设备的名称，它有传输能力。此项要求必须指定。

< rcvr-name >是一个通信设备的名称，在接收平台上有接收能力。如果没有指定,则假定目标接收通讯与提供的< xmtr-name >相同的名称。

11.6.2. report_to

不通过指挥链进行报告指令的传输，通过通信将消息路由到指定的收件人

report_to [commander | peers | subordinates] via <xmtr-name> [to <rcvr-name>]

report_to <报告类型即向谁汇报> via <通信名称> to <接收方通信名称>

如果通信名称相同 report_to <报告类型即向谁汇报> via <通信名称>

示例：平台 sam_1 带有雷达传感器，可以捕获目标 track，然后通过通信将 track 传输给 jet_2 使得 jet_2 捕获到 track

示例解释（平台 jet_2 使用 commander<commander-name>方式指定指挥官为 sam_1 这样 sam_1 就成为 jet_2 的指挥官，sam_1 可以不通过指挥链进行通信，可以使用找寻下属的方式发送报告）

```
platform sam_1 SINGLE_LARGE_SAM
   position 26:21:21.53s 49:25:20.23w
   side blue
   heading 90 degrees

   processor data_mgr
      report_to subordinates via blue_comm1
   end_processor

   add comm blue_comm1 TEAM_DATALINK
      network_name blue_net1
      internal_link data_mgr
      internal_link task_mgr
   end_comm
end_platform
```

```
platform jet_2 jet_fighter
   position 26:28:21.147s 49:33:16.627w
   side blue
   altitude 30000 feet
   heading 45 degrees
   route
      position 26:28:21.15s 49:33:16.63w altitude 30000.00 ft agl
         speed 500 mph
      position 26:23:51.59s 48:53:40.95w
   end_route
   add comm blue_comm1 TEAM_DATALINK
      network_name blue_net1
      internal_link data_mgr
      internal_link task_mgr
   end_comm
   commander sam_1
end_platform
```

```
processor data_mgr
   report_to subordinates via blue_comm1
end_processor
```

11.6.3. report_to command_chain

通过指挥链进行报告指令的传输

report_to command_chain <cmd-chain-name> [commander | peers | subordinates] via <xmtr-name> [to <rcvr-name>]

report_to command_chain <指挥链名称> [commander | peers | subordinates] （三选一“接收平台对于本平台的身份”）via <发送平台通信>[接收平台通信]

示例：平台 sam_1 加入指挥链 PIEATE 并设置指挥官为自己（SELF），添加通信 blue_comm1，设置 network_name 为 blue_net1

```
platform sam_1 SINGLE_LARGE_SAM
   position 26:21:21.53s 49:25:20.23w
   side blue
   heading 90 degrees

   processor data_mgr
      report_to command_chain PIEATE subordinates via blue_comm1
   end_processor

   add comm blue_comm1 TEAM_DATALINK
      network_name blue_net1
      internal_link data_mgr
      internal_link task_mgr
   end_comm

   command_chain PIEATE SELF
end_platform
```

平台 jet_2 加入指挥链 PIEATE 并设置指挥官为 sam_1，添加通信 blue_comm1,设置 network_name 为 blue_net1

```
platform jet_2 jet_fighter
   position 26:28:21.147s 49:33:16.627w
   side blue
   altitude 30000 feet
   heading 45 degrees
   route
      position 26:28:21.15s 49:33:16.63w altitude 30000.00 ft agl
         speed 500 mph
      position 26:23:51.59s 48:53:40.95w
   end_route

   add comm blue_comm1 TEAM_DATALINK
      network_name blue_net1
      internal_link data_mgr
      internal_link task_mgr
   end_comm

   command_chain PIEATE sam_1
end_platform
```

所以两个平台可以通过 blue_comm1 进行通信，并且两者都在 PIEATE 这条指挥链中，并且 jet_2 是 sam_1 的下属，所以可以使用以下方式链接通信

```
processor data_mgr
   report_to command_chain PIEATE subordinates via blue_comm1
end_processor
```

```
processor data_mgr
   report_to command_chain PIEATE subordinates via blue_comm1 to blue_comm1
end_processor
```

blue_comm1 to blue_comm1 是指该信息是从 blue_comm1 发送到 blue_comm1，因为两个平台的通信都为同一名称，所以是 blue_comm1 to blue_comm1

如果两者名称不同，如下：

设置平台 sam_1 的通信名称为 blue_comm1

```
add comm blue_comm1 TEAM_DATALINK
   network_name blue_net1
   internal_link data_mgr
   internal_link task_mgr
end_comm
```

设置平台 jet_2 的通信名称为 blue_comm2

```
add comm blue_comm2 TEAM_DATALINK
   network_name blue_net1
   internal_link data_mgr
   internal_link task_mgr
end_comm
```

两者通信发送报告可以使用以下方式：

report_to command_chain <指挥链名称> <对方身份> via <本平台通信> to <接收方式通信>

```
processor data_mgr
   report_to command_chain PIEATE subordinates via blue_comm1 to blue_comm2
end_processor
```

11.6.4. report_to platform

使用平台名称和通信名称

report_to platform <platform-name> comm <comm-name> via <xmtr-name>

report_to platform <接收平台名称> comm <接收平台通信名称> via <本平台通信名称>

```
processor data_mgr
   report_to platform jet_2 comm blue_comm2 via blue_comm1
end_processor
```

12. 环境

12.1、概述

仿真环境中的全局环境设置,特别是在雷达和传感器仿真中用来定义影响传感器性能和目标检测的环境因素。这些设置可以帮助模拟不同的自然条件和地理特征,从而影响仿真中的对象和传感器的行为。

12.2、命令

12.2.1. land_cover (土地覆盖类型):

为接收机杂波模型指定土地覆盖

land_cover <land-cover-type>

指定地表覆盖类型，如城市、森林、沙漠等。不同类型的地表覆盖会影响雷达杂波的特性。

为接收机杂波模型指定地表覆盖类型。在雷达信号处理中，地表覆盖（land cover）指的是地面上的不同类型的物质，如土壤、植被、水体、人造结构等，它们对雷达波的反射特性各不相同。

指定地表覆盖类型对于建立准确的杂波模型至关重要,因为不同的地表覆盖会对雷达信号产生不同的影响，从而影响雷达对目标的检测和识别能力。例如：

森林地区可能会产生强烈的杂波。

城市地区由于建筑物的存在，可能会产生特定的回波模式。

水体可能对雷达波产生吸收或反射，取决于其表面状态和雷达波的极化方式。

在设计雷达系统或进行雷达信号处理算法开发时，了解和指定这些地表覆盖类型有助于：

优化雷达参数，如发射功率、波束宽度和频率。

改进信号处理算法，以区分目标信号和杂波。

提高目标检测的准确性和雷达系统的整体性能。

简而言之,要求用户为雷达系统中用于处理和过滤杂波的模型选择或定义合适的地表覆盖特征。

有效值：general(一般)、urban（城市）、agricultural（农业）、rangeland_herbaceous（草本牧场）、rangeland_shrub（灌木牧场）、forest_deciduous（落叶森林）、forest_coniferous（原始森林）、forest_mixed（混合森林）、forest_clear_cut（砍伐殆尽的森林）、forest_block_cut（块状砍伐的森林）、wetland_forested（湿地森林）、wetland_non_forested（非森林湿地）、desert（沙漠）

默认值：general(一般)

12.2.2. land_formation (地貌形态):

指定接收机杂波模型的地形

land_formation <land-formation-type>

指定地形类型，如水平、倾斜、起伏等。地形的形态会影响雷达波的传播和散射。

接收机杂波模型指定地形形态。在雷达系统中，"receiver clutter model"（接收机杂波模型）是一个用来模拟和预测雷达接收机接收到的来自地面、建筑物、植被等非目标物体的反射信号的模型。这些反射信号被称为杂波，它们可能会干扰对真实目标的检测。

因此，这句话通常出现在雷达信号处理或雷达系统设计中，要求用户或设计者定义或指定雷达系统在处理信号时所考虑的地形特征。这可能包括：

平原、山地、城市地区、森林、水体等

正确指定地形形态对于减少杂波干扰、提高雷达目标检测的准确性和效率非常重要。

有效值：level(水平)、inclined（倾斜）、undulating（起伏的）、rolling（滚动）、hummocky（圆丘）、ridged（山脊）、moderately_steep（适度陡峭的）、steep（陡峭的）、broken（破碎）

默认值：level(水平)

12.2.3. sea_state (海洋状态):

为接收机杂波模型指定海况

sea_state <value>

指定海洋表面的波涛状态，从无波到非常粗糙。海洋状态会影响舰船和潜艇的雷达信号。

有效值 0-6.

度	高度 （m）	描述
0	无波	平静（玻璃状）
1	0 - 0.10	平静（涟漪）
2	0.10 - 0.50	光滑
3	0.50 - 1.25	轻微
4	1.25 - 2.50	温和
5	2.50 - 4.00	粗糙
6	4.00 - 6.00	非常粗糙

默认值：0

12.2.4. wind_speed (风速)

wind_speed

指定风速，这些因素会影响大气传播的雷达波和空中目标的动态。

默认值：0 m/s

12.2.5. wind_direction (风向)

wind_direction

指定风向，这些因素会影响大气传播的雷达波和空中目标的动态。

默认值：0 度

PS:**WSF** 风向为风流向的方向;不是从哪里来的。

12.2.6. wind_table (风表):

允许按高度定义风速和风向，用于更复杂的大气模型。

```
wind_table
       0 ft    90 deg   20 kts
    5000 ft    95 deg   25 kts
   10000 ft   100 deg   30 kts
   15000 ft   100 deg   35 kts
   20000 ft   120 deg   40 kts
   25000 ft   125 deg   45 kts
   30000 ft   125 deg   50 kts
   35000 ft   130 deg   55 kts
   40000 ft   130 deg   60 kts
end_wind_table
```

每个条目使用 3 个参数（从最低高度到最高高度）指定风值数据集：

高度<高度值>风向<角度值>风速<速度值>。

Ps:如果没有 wind_table，则 WSF 默认赋值给 wind_speed 和 wind_direction。

12.2.7. cloud_altitude_limits (云层高度限制)

cloud_altitude_limits

指定云的最小和最大高度范围。这被一些衰减模型所使用。

默认值： 0 m 0 m（无云）

云层高度分类（仅供参考）

低云：600m 到 2000m，包括积云、层云和层积云（600m-900m）

中云：2000m 到 6000m,高积云、高层云和雨层云。高积云通常不会带来降水，高层云和雨层云能带来持续降雨或降雪。

高云：5000m 到 14000m,包括卷云、卷积云、和卷层云。通常不会带来降水。

12.2.8. cloud_water_density (云中水密度):

cloud_water_density

指定云中水的密度。这被一些衰减模型所使用。

定义云层的高度范围和云中水的密度，影响雷达波的衰减和散射。

默认值： 0 kg/m^3

（仅供参考）

小雨：云中液态水含量可能在 0.1 g/m^3 到 1g/m^3 之间。

中雨：云中液态水含量可能在 1g/m^3 到几克每立方米。

大雨：云中液态水含量可能超过 3g/m^3,有时甚至达到 5g/m^3 或更高。

12.2.9. rain_altitude_limit (降雨高度限制)

rain_altitude_limit

指定存在降雨的高度。这被一些衰减模型使用。

默认值： 0 m

12.2.10. rain_rate (降雨率):

rain_rate

指定降雨量累积率。这被一些衰减模型使用。

定义降雨的高度和降雨量累积率，这些因素会影响雷达信号的衰减。

默认值： 0 m/s

降雨率标准分类（仅供参考）

小雨：降雨率低于 2.5mm/h

中雨：降雨率在 2.5mm/h 至 7.6mm/h 之间

大雨：降雨率在 7.6mm/h 至 50mm/h 之间

暴雨：降雨率在 50mm/h 之上

注：降雨率越大对雷达信号的衰减力度越大

12.2.11. central_body (中心体):

central_body.....end_central_body

指定仿真中使用的中心天体，如地球、月球、太阳等，以及相关的椭球体模型。

```
central_body <central-body-type>
   polar_offset_angles
end_central_body
```

<中心体类型>的选项如下：

earth_wgs72 (Earth World Geodetic System 1972):中心体椭球体根据 WGS-72 标准定义。

earth_wgs84 (Earth World Geodetic System 1984):中心体椭球体根据 WGS-84 标准定义。

earth_egm96 (Earth Gravity Model 1996):中心体椭球体是根据 EGM-96 标准定义的。

moon:中心体椭球体是根据公布的月球参数定义的。

sun:中心体椭球体是根据公布的太阳参数定义的。

Jupiter：中心天体椭球体是根据公布的木星参数定义的。

默认值：earth_wgs84

polar_offset_angles <角度值>

polar_offset_angles

指定中心天体极点相对于世界坐标系的偏移角度，用于精确的坐标转换。

指定天体中间极点 （CIP） 相对于 WCS （ITRS） 坐标系的 central_body 极偏移角（x_p分别为 y_p和 ）。提供这些值（十分之一角秒量级）可以在 ECI 和 WCS 坐标之间实现非常精确的转换。

默认 0.0 rad 0.0 rad

12.3、使用方式

AFSIM 环境是全局环境，在使用时可以在全局进行定义

以下在 Main 中定义为例：

```
global_environment
   land_cover desert
   land_formation hummocky
   sea_state 0
end_global_environment
```

如上图为接收机杂波模型指定土地覆盖，为（desert）沙漠,指定接收机杂波模型的地形为(hummocky)小丘，为接收机杂波模型指定海况为 0（平静）海洋状况默认值也为 0

这些设定的环境参数在杂波模型使用 surface_clutter 杂波模型被使用，用来确定陆地地形、陆地覆盖范围和海洋状态。详见 surface_clutter 杂波模型说明。

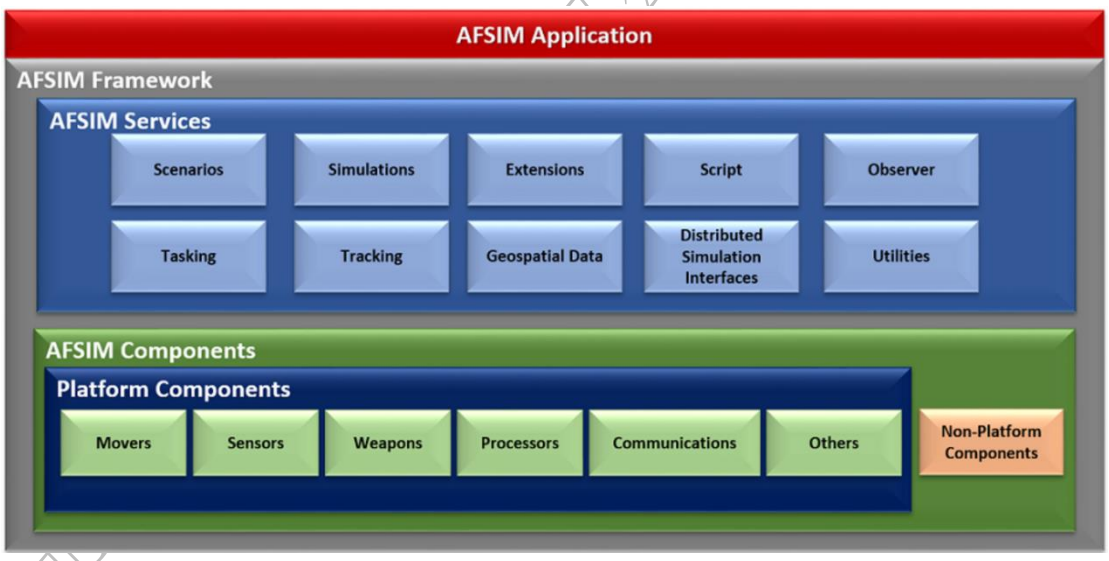
附录一（AFSIM 仿真系统-架构概览）

AP1.1. 介绍

本文档从最终用户的角度描述了 AFSIM 架构，旨在帮助最终用户深入了解 AFSIM 的操作概念。

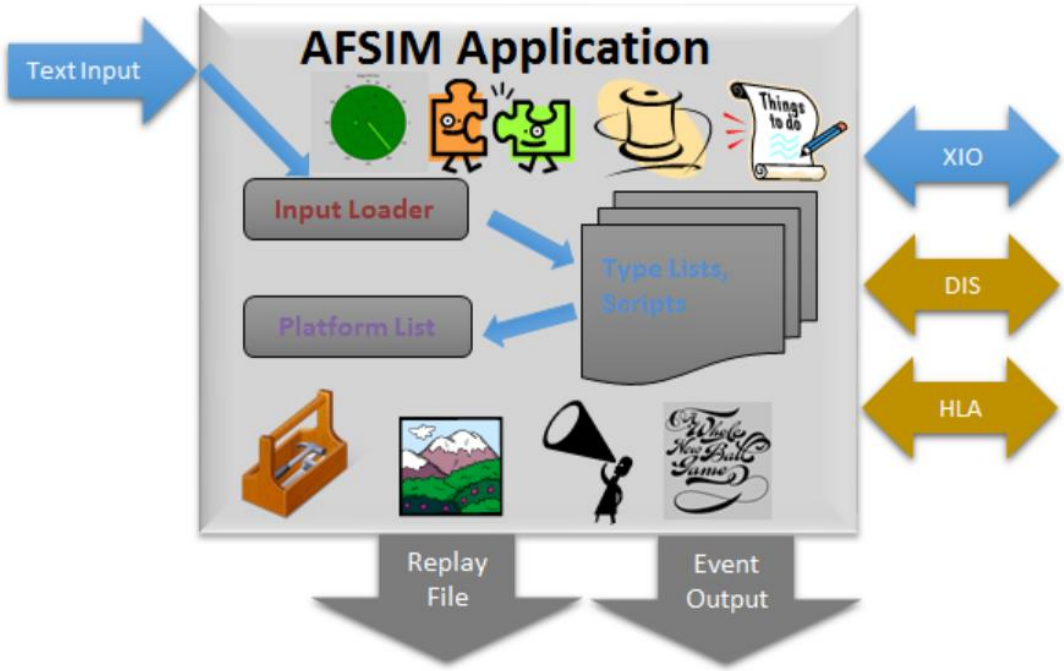
AP1.2. 核心架构

AFSIM 基于面向对象的 C++架构，提供了一种可扩展和模块化的架构，使得许多附加功能能够轻松集成。AFSIM 允许新的组件模型（如传感器、通信、移动器等）以及全新的组件类型被插入并在框架中使用。扩展和插件是框架扩展以集成新平台组件模型、新扩展平台功能以及新扩展仿真服务的主要机制。插件功能是一种扩展形式，允许用户在不重新编译 AFSIM 核心代码的情况下添加新功能。使用插件可以更容易地分发扩展功能，并提供为特定分析选择使用哪些扩展功能的能力。以下图表展示了 AFSIM 提供的主要框架组件和服务，这些组件和服务可以进行扩展。

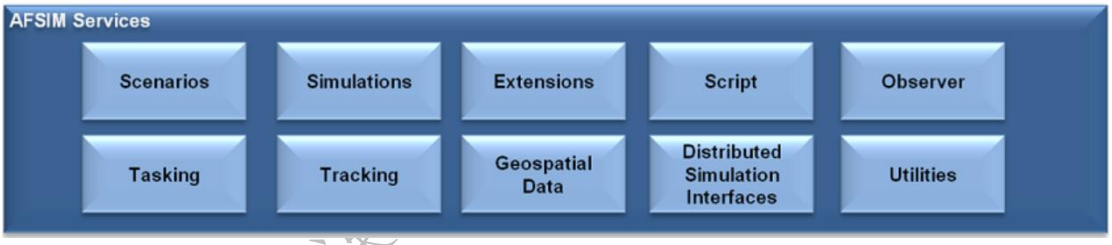


AP1.3. 核心应用

基于 AFSIM 的可执行文件通常由一个利用 AFSIM 服务的 AFSIM“应用程序”组成。这个应用程序维护着脚本类型、扩展和插件管理器以及应用程序配置数据。该应用程序由一个或多个场景组成，这些场景拥有类型工厂和列表、用户输入以及脚本。场景由一个或多个模拟组成，具体取决于应用程序。模拟包含了类型实例、接口（如 DIS、XIO、observer（观察者）、terrain（地形））以及运行时数据，包括事件管理和线程处理。



AP1.4. 核心服务



AFSIM 提供了处理和支持仿真执行以及其他常规计算和基本功能的能力。

- **Scenarios(场景)** - 提供场景输入处理、类型列表和脚本。
- **Simulations(仿真)** - 提供基于时间的事件处理和维护平台列表。
- **Thread Management(线程管理)** - 提供线程和多线程管理能力。
- **Extensions and Plug-Ins(扩展和插件)** - 提供添加新服务和组件的通用方法。
- **Script(脚本)** - 提供实现和扩展 AFSIM 脚本语言的基础结构。
- **Observer（观察者）** - 提供一种通用的发布-订阅服务，用于从仿真中提取数据。
- **Tasking(任务分配)** - 允许跨平台的任务分配和行为建模。
- **Tracking(跟踪)** - 允许根据传感器测量结果形成轨迹以及轨迹关联和融合。
- **Geospatial(地理空间)** - 提供地形和视线数据。
- **Distributed Simulation Interfaces(分布式仿真接口)** - 仿真接口应用接口标准以实现仿真的互操作性（IEEE 1278 & 1516）。
- **Utilities(实用工具)** -提供地球模型、坐标系、数学例程、人工智能构造等。

AP1.5. 场景（Scenario）

Scenario Input Loader（场景输入加载器）提供了从输入文件加载模拟的机制。场景类型列表和脚本为 AFSIM 组件和脚本提供用户输入的内部表示。在一个应用程序中可能有多个场景，尽管这在某种程度上不太规范。

AP1.6. 仿真（Simulations）

每个仿真都维护一个平台列表。仿真从*scenario（场景）*中的场景类型列表中实例化。仿真使用时间管理来以实时或比实时更快（即构造性，尽可能快）的模式推进仿真的状态。仿真拥有一个事件管理器，负责按时间顺序处理事件。尽管 AFSIM 是基于事件的，但事件是抽象的，因此分析师不必担心它们。每个场景中可能包含多个仿真。

AP1.7. 线程管理（Thread Management）

模拟线程管理使得模拟执行更加迅速且流畅，特别是在虚拟实时环境中。通过线程化，可以利用独立的执行线程并行处理一些任务，例如传感器和移动器更新、模拟界面以及地理空间检查。

AP1.8. 扩展和插件(Extensions and Plug-Ins)

AP1.8.1. 扩展(Extensions)

应用程序、场景和模拟都可以“扩展”。应用程序扩展表示可以添加到应用程序的可选功能。方案扩展用于注册新的组件类型并提供对输入加载器的访问。Simulation Extensions（模拟扩展）提供特定于模拟的可选功能，并允许访问观察者服务。

AP1.8.2. 插件(Plug-Ins)

扩展插件管理允许在不更改交付的框架代码的情况下开发扩展的 AFSIM 功能。

AP1.9. 实用工具(AFSIM Utilities)

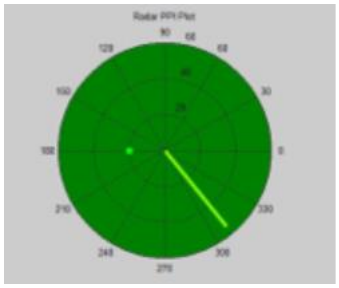
AFSIM Utilities 提供各种各样的软件工具：

- 专用数据类型
- 人工智能构造
- 数学类和算法
- 输入、输出和文件管理例程
- 时间管理
- 地球坐标参考系和大气数据
- 观察者服务使用的发布/订阅类
- 以及其他许多工具……

AP1.10. 任务分配（Tasking）

任务分配是一种服务，用于发送和接收与轨迹或感知相关的*任务分配。它允许用户利用有限状态机的概念对轨迹进行分类。用户定义一组转换规则，这些规则定义了从一个状态转换到另一个状态的条件。在 AFSIM 任务分配中，每条轨迹都维护着自己的状态。

AP1.11. 跟踪(Tracking)



提供轨迹关联、轨迹滤波和轨迹融合功能。AFSIM 提供了原生的跟踪算法，同时也可集成第三方跟踪算法。目前，AFSIM 支持：

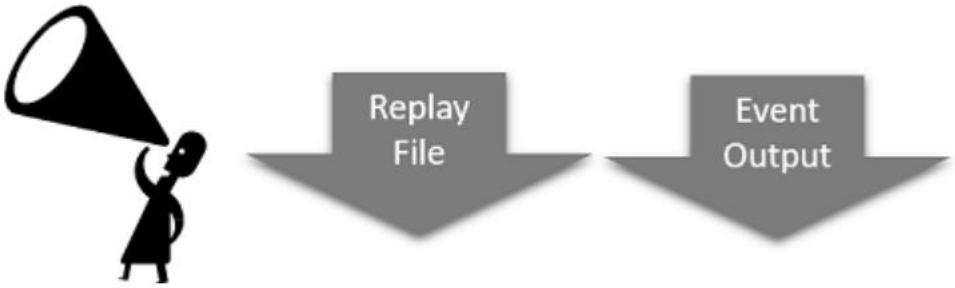
- 完美和不完美的轨迹相关选项。
- 多种轨迹滤波器选择。
- 协方差矩阵提供检测和跟踪概率区域。

AP1.12. 地理空间数据(Geospatial Data)

地形（地理空间数据）管理：为 DTED 和“浮点网格”（与 ESRI-GIS 兼容）数据库提供优化的地形高程查找功能。

视线管理：提供目标可见性计算服务。

AP1.13. 观察者(Observer)



观察者允许在不更改框架的情况下提取数据。用户可以轻松创建“脚本观察者”，以脚本类型输出方式提取数据，而无需对软件进行修改。观察者可用于标准和自定义输出、回放文

件和分布式接口。

AP1.14. 脚本(Script)

脚本管理器理解脚本类型并启用脚本语言。脚本类型易于扩展，以适应新集成的模型和服务。

AP1.15. 分布式仿真接口(Distributed Simulation Interfaces)

AP1.15.1. DIS & HLA

AFSIM 集成了行业标准接口，这些接口允许平台与其他模拟中的实体进行交互。

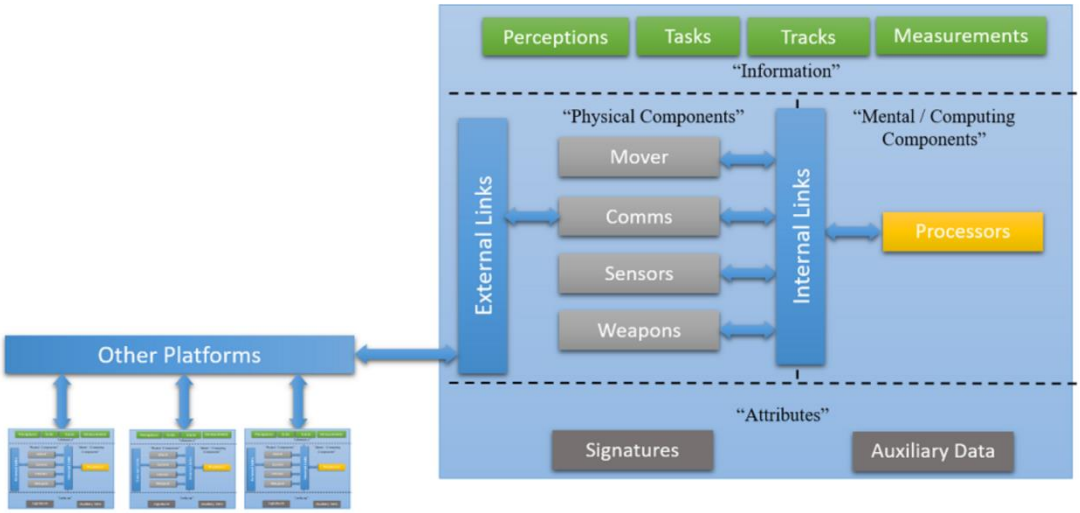
AP1.15.2. XIO

XIO 是 AFSIM 的一个专用接口，它允许模拟分布在多台计算机上，并通过图形用户界面进行模拟控制。

AP1.16. 核心组件

AP1.16.1. 平台(Platforms)

平台是其构成组件的“容器”。平台由以下部分组成：*物理组件*心理/计算组件*信息*属性*链接。



AP1.16.2. 移动器(movers)

移动器维护着它所附着的平台的动力学状态（位置、方向、速度、加速度等）。移动器有多种选项可供选择，范围从地下到空间动力学模型。

AP1.16.3. 通信(Communications)

通信设备通过外部链路在平台之间发送和接收消息。AFSIM 支持有线或无线设备，使用发射器、接收器和天线来捕捉通信系统的全部物理方面。

AP1.16.4. 传感器(Sensors)

传感器创建测量值并通过链路在轨迹消息中传输它们。在 AFSIM 中，传感器经常使用发射器、接收器和天线。AFSIM 提供了雷达传播、衰减、杂波和误差的多种选项。

AP1.16.5. 武器(Weapons)

武器是指旨在阻止其他物体运行（永久或暂时）的装置。在 AFSIM 中，大多数武器是显式武器，即对象被明确建模为平台（如导弹和炸弹），与隐式武器相比，后者在模拟中不作为平台表示（如干扰机或激光器）。

AP1.16.6 处理器(processor)

处理器定义行为或计算算法，类似于人脑或计算机。大多数处理器由用户使用 AFSIM 脚本语言定义，但 AFSIM 也提供了许多专用处理器。

AP1.17. 术语

- AFSIM - 高级模拟、集成和建模框架

- COMMS - 通信
- DIS - 分布式交互模拟
- HLA - 高级体系结构
- WSF - 世界模拟框架
- XIO - 外部输入/输出

附录二（AFSIM 概要介绍）

AP2.1. 引言

AFSIM 的开发理念是一个通用的仿真建模框架，在具有业务需求的共同环境中使用通用模型。为了鼓励政府和行业购买，AFRL 不仅构建了一个强大的产品，还向批准用户免费提供软件、源代码和培训。迄今为止，美空军研究室（AFRL）已将 AFSIM 授权给 275 个政府、行业和学术组织，并为 1200 多名用户提供培训。AFRL 的软件开发、分发、支持和治理的整体方法可以作为鼓励未来免费软件产品广泛采用的模型。

AP2.1.1. 开发背景

数字孪生技术获美国国会关注，标志其广泛应用的到来。2020 财年国防授权法案要求国防部报告 F-35 项目中数字孪生的实施情况。为此，国防部需构建高效、经济、易用的建模框架，并培养接受数字模型结果的文化。AFRL 开发的 AFSIM 框架，前身为波音公司投资 3500 万美元研发的 AFNES，已成为国防建模与仿真社区的通用工具。AFSIM 不仅限于空军，其多领域特性支持海陆空天及网络代理的综合模拟，拥有超过 1200 名来自全球各界的用户，为武器系统评估和决策提供有力支持。

AP2.1.2 持续发展

AFRL 在俄亥俄州代顿总部为普通用户和代码开发人员提供免费培训，降低用户门槛。AFSIM 框架及其支持工具的源代码也免费开放，旨在提升透明度与可调试性，减少依赖昂贵商业软件的必要。AFRL 鼓励社区参与，通过领域工作组模式促进自主治理与专家共识，推动 AFSIM 的持续发展。

AFSIM 作为技术成熟仿真平台，特别在飞行器自主性测试方面表现卓越，为自主算法的开发与测试提供统一环境。其成功吸引了 DARPA、JHU APL 等多家机构采用，成为武器系统概念分析的首选工具。AFRL 与行业紧密合作，如洛克希德-马丁公司的投资，进一步彰显了 AFSIM 的广泛认可与前景。波音公司作为 AFSIM 前身的开发者，持续支持并见证其成长。

AP2.2. 平台特点

AP2.2.1. 开放性

为了克服框架的扩展和兼容性限制，利用“即插即用”模块化思想，AFSIM 被设计为一个开放系统。这种模块化方法允许建模者，而不是 AFSIM 程序员，为模拟中使用的模型确

定适当的保真度（即模拟基础物理的程度）。同样用户可以调整每个平台的保真度，以满足其特定的模拟需求。例如，飞机模型的保真度可以在沿预定向量移动的空间点之间变化，也可以根据虚拟座舱控制器的位移，改变速度、方向、高度等的全六自由度模型。模块化方法还允许在不改变核心 AFSIM 代码的情况下，重用或修改各种平台的现有模型。

AFSIM 的模块化结构使 AFRL 能够以两个安全分类级别分发代码，用户可以通过添加额外的软件模块来适应其特定的安全要求。AFRL 提供代码的非保密和保密（美国机密）版本。两个可用版本之间的主要区别仅仅是所包含模型的数量、类型和保真度。保密版本还配备了国家航空航天情报中心（NASIC）批准的许多威胁系统模型，以及国家地理空间情报局（NGA）数字地形高程数据（DTED）模型。最终用户可以添加自己的模块，以包含其他感兴趣平台的模型，以用于其专业用途。

AP2.2.2. 适应性

AFSIM 涵盖了广泛的军事模拟，包括工程、交战、任务和“轻型战役”级（通过兵棋推演和实验进行）。

仿真级别	复杂性维度	时间维度
战役级	all VS all	天
任务级	n VS n	小时
交战级	1 VS 1	分钟
工程级	子系统交互	秒

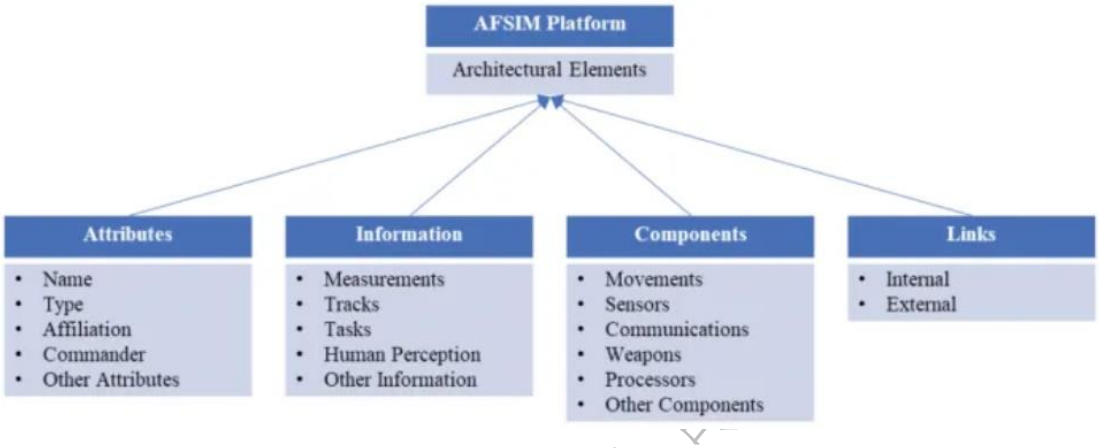
如上表所示，工程级包括与其他子系统进行短期交互。其中一个例子是射频（RF）发射器与接收器交互，以识别子系统级能力和限制。交战级包括“手动”战斗，即 AFSIM 术语中两个实体或平台之间的简短数据交换。例如，蓝（友军）和红（敌军）飞机之间的导弹交换将构成交战级模拟。下一个复杂程度将是任务级，例如模拟多架红蓝飞机在一次出击或任务期间（通常为几个小时）进行的一系列战斗数据交换。这些模拟可以包含多达数千个实体。战役级交战进一步扩展了这一点，可能包括在给定区域在延长的时间段内（即几天甚至几个月）的所有红色和蓝色平台。AFSIM 开发的重点主要在于交战和任务层面，并持续的发展通 “战役级”能力。当需要更全面地探索工程或完整的战役建模时，可以利用其他 M&S 工具，例如合成战区作战研究模型（STORM），当前主要由空军研究所、分析和评估办公室（AF/A9）使用。

AFSIM 使用户能够将场景缩放到适当的仿真级别，以研究感兴趣的项目。每个后续级别在逻辑上建立在较低级别的基础上，以创建更复杂的仿真，以识别在较简单的仿真中可能不明显的系统体系特性。例如，耗尽弹药和燃料储备的作战效果在交战或任务级别可能不明显，但在战役级别仿真中则完全改变了游戏规则。AFSIM 仿真的规模和复杂性的限制因素

是主机平台的存储、内存和计算能力，以及运行仿真所需的相关时钟时间。AFSIM 允许用户通过调整与平台相关的各种参数和行为，在处理时间和输出保真度之间设置所需的平衡。

AP2.2.3. 灵活性

为了实现上述平台类型、保真度、仿真类型的灵活性要求，AFSIM 使用四个架构元素（属性、元素、组件和信息链接）来描述仿真中的每个平台，如下图所示。



属性包括平台名称、类型和隶属关系等标准数据。该子元素可以扩展到包括任务特有的信息，如雷达、光学和红外特征数据，以确定飞机被敌方传感器探测到的脆弱性。信息元素包括平台上的数据，以及接收这些数据的人如何感知这些数据的详细信息。对于飞机，这将包括向飞行员显示的数据类型（即高度、速度、航向、雷达指示等），以及驱动这些显示的无数原始数据。组件元素由直接控制平台行为的各种模型组成。这些模型描述了平台如何在时空内移动、感知周围环境、处理收集到的信息、与其他平台通信、使用动能和非动能武器对抗敌方平台，以及执行各种其他任务。最后，数据链接元素协调平台各子系统之间的数据交换，以及与其他平台的通信。

AP2.2.4. 易用性

对于实时仿真，利用其 Warlock 图形用户界面（GUI），AFSIM 同样允许“操作员在回路中”执行，以促进分析兵棋推演。具体而言，Warlock 使操作员能够触发各种场景事件，控制各个平台，甚至从平台内部体验任务，就像飞行模拟器一样。Warlock 还促进了操作员以“阵营”的形式创建，例如，友方平台操作员的蓝色阵营和敌方平台操作员的红色阵营，每个阵营只能访问该阵营平台收集的虚拟信息。换句话说，红色阵营和蓝色阵营对彼此都有不完整的信息。AFSIM 可以通过降低同一阵营成员之间的信息流来进一步增加兵棋推演的真实性。Warlock 还支持创建白组（裁判）他们拥有所有平台的完美信息，利用这些信息来控制整个战争游戏的流程。

然而，在执行任何类型的 AFSIM 模拟之前，用户必须定义各种平台和组件模型，然后制作兵棋推演场景。为了促进这一过程，AFRL 分发了作为支持工具（如 Warlock）的 AFSIM 集成开发环境（IDE）Wizard。就像现代软件开发 IDE 一样，Wizard 作为一个单一的应

用程序来编辑场景文件：编写 AFSIM 脚本，图形化操纵场景布局，运行软件可执行文件（即在 AFSIM 中运行场景），查看结果输出或错误消息。它还突出显示文件语法，标记未知命令，并提供上下文相关的文档。Wizard 甚至附带了一个自动完成功能和一个脚本调试器，以最大限度地减少开发和调试模型和场景所需的时间。

AP2.2.5. 易集成

AFSIM 对虚拟和构造性模拟的支持力度也在逐步增大。在构造性模拟中，模拟操作员控制模拟系统，例如军事战斗，其中红色和蓝色玩家都是计算机控制的。在虚拟模拟中，你有真实操作员控制模拟系统，例如飞行员驾驶飞行模拟器。AFSIM 可以用于构造性地开展军事能力的大型任务空间探索，可能涉及数以万计的独特测试点，以非实时方式执行。然后，可以利用这种构造性模拟活动的结果来定义和进行实时运行的虚拟模拟，以调查更窄的任务空间（由构造性模拟提供），以便在飞行员参与的情况下进行更集中的评估。这使得在构造性模拟和虚拟模拟中都可以使用相同的底层模拟模型，从而在两个环境中提供更一致的建模和分析。此外，AFSIM 还可以与其他仿真或其他模拟器/仿真器连接，以提供真正的实时虚拟构造（LVC）仿真能力。使用分布式交互仿真（DIS）或其他支持的通信协议，AFSIM 可以与其他仿真或实时实验交互，以提供额外的实体（虚拟和构造）、系统和子系统模型、威胁系统或其他潜在的仿真能力。这使得 AFSIM 可以根据需要，通过额外的功能来增强和补充更大的仿真或实验环境，以最好地实现任何给定的测试和分析目标。

AP2.3. 核心功能

AP2.3.1. 集成开发环境

集成开发环境是一种 AFSim 的程序性工具，它通过提供各种工具（如工程浏览、类型浏览、路径编辑、脚本编辑器以及地图显示等多种功能）来简化创建和编辑想定的过程，并将上述介绍的五种仿真引擎和三种工具更具交互性的集成到 Wizard 中，便于用户快速构建仿真场景，快速调用仿真工具。

（1） 作战单元部署/任务规划；

（2） 想定开发/模型开发；

（3） 命令和脚本双重控制；

（4） 高精度三维场景构设（支持高精度地形数据）；

（5） 脚本编辑/调试/执行环境；

（6） 仿真模式（五大仿真引擎）管理；

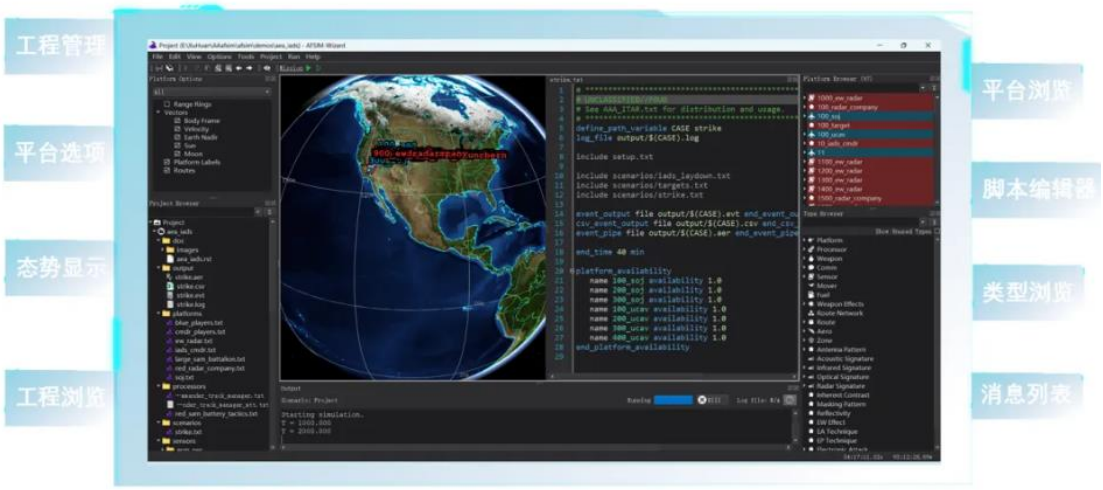
（7） 支持飞行手柄与摇杆实现座舱飞行接口；

（8） 提供人在回路的导调控制手段；

（9） 支持图形化自定义控件和显示；

（10） 利用脚本，可以在地图上灵活绘制需要的效果，过集成开发环境脚本系统添加

到工程中。



AP2.3.2. 超实时仿真计算引擎

Mission 是 AFSim 中的超实时仿真引擎。超实时仿真引擎读取工程中的脚本文件，并执行模拟，输出所有仿真数据后，生成可视化回放文件.aer，可通过结果可视化回放工具对仿真过程进行展示。

（1） 支持读取脚本编辑器中的文本文件；

（2） 执行模拟、推进时间、移动平台、做出决策和执行对象之间的交互；

（3） 支持事件步进或帧步进；

（4）支持输出二进制事件报告文件，可以使用结果可视化回放工具中的应用模块查看，由输入命令 event_pipe 控制；

（5） 支持使用结果可视化的二进制“回放文件”，通过指定 dis_interface 块中的 record 命令。

AP2.3.3. 人在回路实时仿真引擎

人在回路实时仿真引擎是一个提供图形用户界面环境的应用程序，可在其中查看和控制场景，译为 Warlock。主要目标是在 AFSIM 框架内支持（OITL）战争游戏和实验，提供的功能涵盖交战、任务、作战和战略层面的模拟和分析领域。人在回路实时仿真引擎提供了可扩展的架构和相应的接口来支持自定义控件和显示。

（1） 控制 AFSIM 人在回路实时仿真；

（2） 涵盖交战、任务、作战和战略层面的模拟；

（3） 支持控制 AFSIM 实体进行战争演习，支持多屏幕的人在回路实时导调；

（4） 提供可扩展的架构和相应的接口，支持自定义控件和显示；

（5） 支持显示包括控制平台、传感器、武器、通信等的对话框；支持查看整个模拟环境；

（6） 支持使用通用驾驶舱功能控制单个 AFSIM 实体，并使用操纵杆飞行实体；

（7） 可扩展体系结构支持自定义控件和自定义显示；

（8）支持个性化配置数据显示，支持保存显示器、键盘快捷键、首选项等的当前配置。

AP2.3.4. 电磁计算与绘制引擎

sensor_plot 是 AFSim 的传感器仿真显示绘制引擎，用于评估传感器特性和与用户指定的几何体的交互作用。它具有创建生成图形的文件的功能。

（1） 支持绘制传感器垂直覆盖及垂直图；

（2） 支持绘制传感器水平覆盖及水平图；

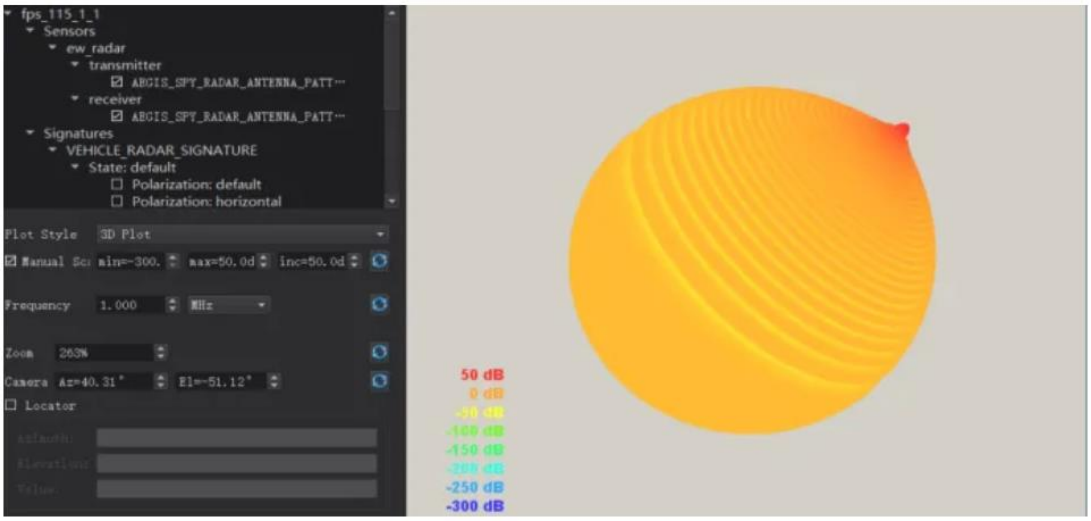
（3） 支持分析传感器球形覆盖范围；

（4） 支持目标飞行路径分析；

（5） 支持雷达杂波处理表；

（6） 支持构建传感器集合的防御区域；

（7） 支持天线方向图绘制。



AP2.3.5. 武器交战计算引擎

武器交战范围计算引擎是一个可选的仿真引擎，可以合并到应用程序中，也可以作为独立的可执行文件分发。武器交战范围计算引擎在各种不同的接合条件下反复触发预定义的武器交战计算模块。捕获命中/未命中结果以量化该定义武器类型的交战能力，从而可以成功地对目标轨道使用武器。

（1） 捕获命中/未命中结果以量化该定义武器类型的交战能力；

（2） 根据交战能力计算武器攻击结果；

（3） 空对空发射计算机；

（4） 空对地通过发射可接受区域和发射计算机计算；

（5） 弹道发射计算机；

（6） 弹道导弹发射计算机；

（7） 轨道发射计算机；

（8） 地对空导弹发射计算机。

AP2.3.6. 交战对抗计算引擎

ENGAGE 为交战对抗计算引擎，用于评估地面武器的有效性（反之，也用于评估目标的脆弱性）。

（1） 支持指定运行的启动器、跟踪器平台；

（2） 支持指定平台的处理器定义；

（3） 支持指定站点坐标、站点航向、站点速度等；

（4） 支持指定站点网格配置；

（5） 支持目标网格配置、简单路径配置、飞行路径配置等；

（6） 支持分阶段、分事件、分批次仿真数据输出；

（7） 支持生成 PK 表，设置 PK 表目录层级结构，设置 PK 表生成路径。

AP2.3.7. 仿真模型库框架

（1） 提供了多个领域，高保真度的仿真模型，模型设计科学规范，建模粒度细，逼真度高；

（2） 采用组件化、参数化、面向对象建模思想；

（3） 覆盖陆、海、空、天、网电等领域；

（4） 支撑联合、战役、战术等作战层次；

（5） 具备作战行动、指挥控制、战场环境、分析评估等能力；

（6） 模型支持任意继承和派生，可以生成现有或在研的所有武器系统模板。

（7） 使其他平台失效的效果：

在构建武器系统时，可以指定武器的效能模型以及类型指标，主要类型包括卡尔顿致死率模型、大气外杀伤力模型、分级杀伤力模型、机动性火力杀伤力模型、球形致死率模型、高能激光杀伤力模型、武器交战杀伤概率表模型。

（8） 武器效能模型分类：

采用物理和概率两种分辨率，对目标的影响可以是概率的、物理的或两者的组合武器效能建模既包括硬毁伤，也包括干扰等软杀伤。

AP2.3.8. 仿真模型库框架

机动平台构建工具是一款 AFSIM 软件工具，提供了一种方便易用的基于 GUI 的应用程序，可帮助用户为 P6DOF 移动器创建 AFSIM 输入文件。使用自定义的基于图形的车辆定义，用户可以模拟飞机、武器和发动机，以供 AFSIM 使用。Mover Creator 界面允许用户从现有的模板模型（由 Mover Creator 提供）创建新的自定义模型，或者用户可以编辑他们以前创建的模型。

（1） 设计平台、武器、引擎等运动模型；

（2） 提供引擎设计、驾驶设计、外形设计、空气动力学设计、性能设计、自动驾驶、平台测试等；

（3） 通过参数化设计与快速气动计算，链接作战仿真与气动性能仿真；

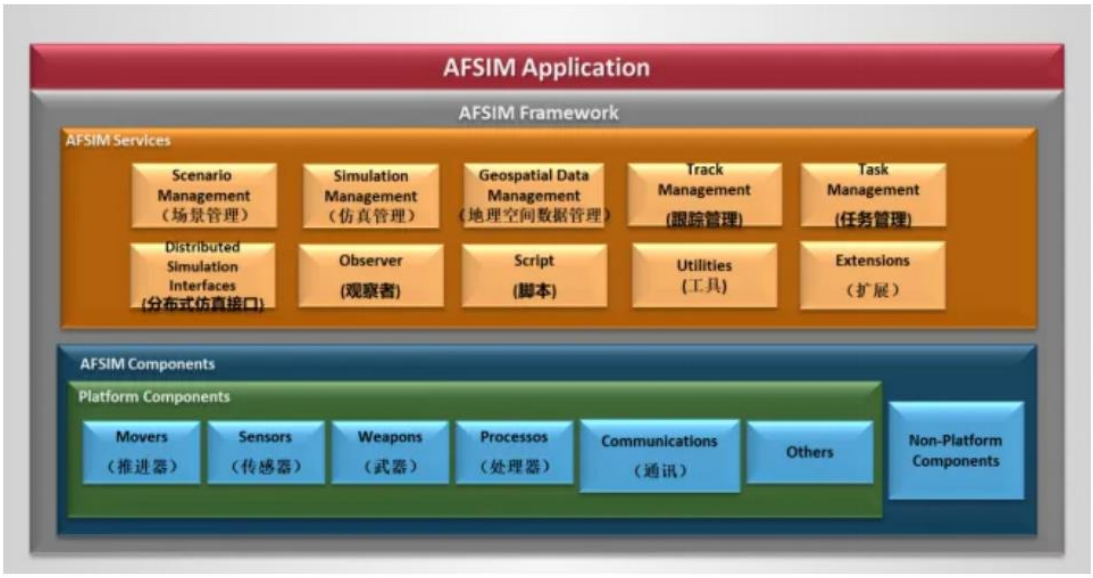
（4） 参数化快速设计环境。

AP2.4. 核心架构

AFSIM 基于 OSGi 提供了一个可扩展的模块化架构，采用面向对象的语言实现，在遵循开放接口和服务协议的基础之上，允许包含许多额外的功能，以便轻松集成。其将系统通过模块之间的依赖关系、应用等级和接口类型等形成层级组织结构。

AFSIM 允许在框架中插入和使用新的组件模型（例如传感器、通信、移动器等）以及全新的组件类型。扩展和插件是扩展框架以集成新的平台组件模型、新的和扩展的平台功能以及新的和扩展的仿真服务的主要机制。插件功能是一种扩展形式，允许在不重新编译核心 AFSIM 代码的情况下添加功能。使用插件可以更轻松地分发扩展功能，并提供选择要用于给定分析的扩展功能的能力。

AFSim 提供主要业务应用层、核心能力服务层和基础组件支持层层，同时在每一层都提供了业务扩展接口。下图显示了已提供且可能扩展的 AFSIM 主要框架组件和服务。



AFSim 的体系结构设计基于支撑特种任务仿真以及装备系统性能验证等多业务领域，具有如下的设计特点：

（1）仿真模型库模板采用组件化设计，可自主装配；

（2） 系统采用插件化模块结构设计，支持灵活扩展；

（3） 提供基于 DIS 网络通信的分布式仿真架构设计；

（4） 采用脚本和 UI 交互结合的仿真场景构建方式；

（5） 提供跨平台架构设计，支持国产化平台运行支持。

AP2.5. 业务应用

AP2.5.1. 电子对抗模拟

电磁域成为继陆域、空域、天域、海域之外的又一个重要的作战领域，同时延伸到了其他四个典型域，针对电磁域作战的研究越来越重要。平台支持复杂电磁环境下的作战模拟，通过各种电磁环境模型、传感器模型、武器模型、电子战模型等构建一个高逼真的复杂电磁环境，支持电子对抗分析、电子防护分析、网络攻击和网络防护分析、多种干扰模拟分析等在电磁域的典型应用。平台拥有多种电磁相关高级算法，包括有干扰功率计算、干扰效应结构、信号效应结构、跟踪效果变量结构、消息效果变量结构、聚合类型等。

通过以上模型的构建，为电子对抗作战模拟提供了能力支持，AFSim 脚本系统提供电子对抗战术指挥的支持，能够编制任务计划、协同对抗、战斗编组以及组织保障。

AP2.5.2. 通信网络模拟

AFSIM 支持平台多种通信方式模拟，支持静态组网和动态组网，在静态组网中设定各个对象之间的通信关系和通信方式，在动态组网中可以根据通信组网策略实现动态的加入网络和退出网络，并能够根据加入/删除网络节点实时、动态的构建调整网络拓扑关系。

通信对象使用5层网络模型（应用、传输、网络、链路和物理）进行组织，以实现通信组网，每个通信对象都包含一个调用协议栈的对象，该对象包含多个“层”对象，这些对象处理从通信对象发送和接收的消息。

网络通信组网节点可以搭载在多类型平台之上，如地面站、指挥中心、地面装备、战斗机、预警机、卫星等，构建多样化、立体式、分层次的通信网络，支持航天器遥测、卫星数据上下行传输、情报共享通信、指挥控制等多种通信应用场景。

AP2.5.3. 统合防空模拟

综合防空（IADS）综合防空系统是实施对敌方进攻拒止的活动，用来抵抗敌人的空中打击，避免敌人空中力量穿透其控制空域的结构、设备、人员、程序和武器。IADS可以执行三大功能——空中监视，战斗管理和武器控制。

IADS体系采用了OODA过程并和C3I节点结合，它不是一个简单的武器或者武器系统，而是所有元素的有机组合，旨在将空域中的被敌人突破的威胁降至最低，实现了对目标观察（Observation）、调整（Orientation）、决策（Decision）、行动（Action）四个在高风险情况下做出有效决策的过程，步骤包括收集相关信息，认识到潜在的偏差，决定是否执行拒止行为以及何种方案，并采取行动，然后用新的信息重复这个“观察-调整-决策-行动”的过程。

整个防空网络的构建主要是以多个C3I指挥控制节点为核心，同时对指挥节点实行分层分级管理，每一个C3I节点统一管理其情报侦察、战斗决策管理和武器打击系统节点，支持作战集中式指令指挥、分散式委托指挥及指导式指挥。

AP2.5.4. 飞行对抗仿真

AFSim 提供六自由度飞行推进器，结合飞行平台的属性参数，通过设定参数和条件可以输出6自由度的运动和姿态数据，除了位置还具有逼真的偏航、俯仰、横滚以及攻角（alpha），这些都是通过基于真实力和力矩计算得出的物理/空气动力学模型，能够支持机动飞行科目、任务执行和对抗演练应用等。

六自由度飞行推进器提供多种机动飞行科目模拟，同时允许用户使用六自由度移动器手动控制平台中的飞行模拟对象。该工具提供了一个窗外(OTW)视图，其中包括一个平视显示器（HUD）覆盖图。它还包括一些实验性功能，例如多功能显示器（MFD）。

六自由度飞行推进器除了在固定翼飞机上使用，还可以在旋翼机、导弹（包括空对空（AAM）、地对空（SAM）、空对地（AGM） 和反弹道导弹（ABM））、弹道导弹、太空运载火箭（包括典型的多级线性助推器，以及平行级配置和有翼再入飞行器）和航天器/卫星上使用。

AP2.5.5. 航天任务仿真

针对航天领域仿真，AFSim 具备非常丰富的仿真模型和应用支撑，仿真模型包括太空环境模型、航天器运动模型、传感器模型、网络模型以及武器模型，基于这些模型，提供全

流程航天任务仿真模型，主要有航天发射、在轨运行、姿态控制、轨道机动、卫星组网、空间环境，支持对地观测、星际航行、深空探测、目标躲避等业务。

AFSim 提供专门用于航天的推进器模型，用于支持航天发射、在轨运行、轨道转移等典型的航天任务。其底层集成主流的轨道传播算法模型（NORAD SGP、SGP4、SGP8、SDP4/SDP8）和用于轨道传播的椭球体坐标模型。同时提供了轨道机动模型，能够支持轨道参数更改（改变偏心率、倾角、RAAN、长半轴、高度），霍曼转移，目标拦截，匹配速度，交汇会合，切线飞行，目标追踪等。

AP2.5.6. 人工智能框架

AFSim提供RIPR人工智能框架,允许场景开发人员快速创建具有复杂行为的灵活代理。RIPR 代理脚本通常使用行为树技术构建。指挥官/下属代理使用作业板技术进行任务分配，通过人工智能框架处理器基类和人工智能框架任务类向场景开发人员公开。

RIPR 为仿真环境中的对象提供来自于外部的控制和对象感知态势（属性、状态、位置、探测对象、跟踪目标及打击目标等）通道，同时为人工智能算法库提供决策指令和态势信息的通道，其作为桥梁完成人工智能决策系统和 AFSim 仿真系统之间的联系，同时为不同类型的人工智能决策系统集成提供了扩展而又不增加 AFSim 的复杂性。

人工智能决策系统通过接收态势信息输出决策信息,这些决策信息通过行为树的形式组织，为仿真对象提供任务控制，使用了这种代理模式，相对于仿真对象外部决策控制和本地决策控制是等同的，都是由对象的任务和当前态势属性所决定。

AP2.5.7. LVC 仿真系统

LVC 仿真是指在仿真系统中同时具有实况仿真、虚拟仿真、构造仿真等三种类型的仿真，为了使开发出来的装备仿真测试系统能够真正服务于装备的系统论证、方案设计、关键技术验证、系统集成试验、系统训练等全寿命周期，所研制的仿真系统必须要能够根据不同的阶段重组其系统组成，这些组成部件根据其真实程度可以是虚拟仿真(V)、构造仿真(C)或实况仿真(L)三种类型中的任何一种或多种类型的组合。

(L：Live)实况仿真，是指真实的人使用实际装备在实际战场的假象行动，主要使用于试验与训练领域；(V：Virtua1)虚拟仿真，真是指系统和军队在合成战场上模拟作战，往往表现为真人操纵模拟系统；(C：Constructive)构造仿真，是一种战争演练和分析工具，通常由模拟的人操纵模拟的系统。

AFSIM 通过标准的 DIS（XIO）协议和可扩展的 DDS、ZMQ 等数据交换接口实现与其他异构仿真系统的数据交换。提供仿真对象代理模式动态导调模块，实现仿真对象属性、状态数据同步。

附录三（AFSIM2.9 更新一览）

AP3.1. 引言

AFSim2.9 在其旧版问的基础上，在模型种类、软件功能、脚本方法和 Demo 示例等多方面进行更新优化，使用户可以获得更加高效便捷、专业全面的仿真体验。

AP3.2. 模型更新

2.9 新增共 42 个模型，而且模型覆盖面较广，填补了传感器覆盖计算模型、网格模型、任务目标评估模型、多分辨率模型容器、空间轨道传播模型、uci 通信组件模型的空白，以及在 2.6 中的伪六自由度推进计算模型提升到真六自由度，大气环境模型也更加丰富。

模型类别	模型名称
大气模型	WSF_JACCHIA_ROBERTS_ATMOSPHERE-Jacchia_Roberts 大气模型
	WSF_PIECEWISE_EXPONENTIAL_ATMOSPHERE 大气密度模型
传感器覆盖计算模型	WSF_SENSOR_COVERAGE 传感器覆盖计算模型
赛博攻击模型	WSF_CYBER_DETONATE_EFFECT 赛博指定攻击模型
燃料模型	WSF_SIX_DOF_FUEL 6 自由度燃料模型
网格模型	WSF_LAT_LON_GRID 纬度和经度的规则矩形网格
	WSF_ZONE_BASED_GRID 纬度和经度的规则网格，但点经过过滤后位于指定区域内
	WSF_DISTANCE_STEPPED_GRID 一个规则的矩形网格，其分隔由距离给出
	WSF_EXISTING_PLATFORM_GRID 由现有模拟平台形成的网格
	WSF_COMPOSITE_GRID 由子网格组成而成的网格
任务目标评估模型	WSF_ACCESS_DURATION_MOE 衡量每个空闲资源在被访问时的持续时间

	WSF_COVERAGE_TIME_MOE 衡量一个特定的网格点可以被覆盖的时间长度
	WSF_NUMBER_OF_ACCESSES_MOE 计算一个网格点访问空闲资源的次数
	WSF_NUMBER_OF_GAPS_MOE 衡量特定网格点所经历的覆盖间隙的数量
	WSF_N_ASSET_COVERAGE_MOE 衡量在特定时刻，网格点能够访问到的空闲资产的数量
	WSF_REVISIT_TIME_MOE 衡量网格点的回访时间
	WSF_SIMPLE_COVERAGE_MOE 衡量网格资产是否与覆盖区指定的某个空闲资产有任何交互
	WSF_TIME_AVERAGE_GAP_MOE 评估覆盖间隙的平均长度
推进器模型	WSF_RIGID_BODYSIX_DOF_MOVER 刚体 6 自由度模型
	WSF_POINT_MASS_SIX_DOF_MOVER 质点 6 自由度模型
	WSF_ARGO8_MOVER-TMAP ARGO 导弹模型对象推进器
多分辨率模型容器	WSF_MULTIRESOLUTION_ACOUSTIC_SIGNATURE 声学特性多分辨率容器
	WSF_MULTIRESOLUTION_COMM 通信多分辨率容器
	WSF_MULTIRESOLUTION_FUEL 燃料多分辨率容器
	WSF_MULTIRESOLUTION_INFRARED_SIGNATURE 红外特性多分辨率容器
	WSF_MULTIRESOLUTION_MOVER 推进器多分辨率容器
	WSF_MULTIRESOLUTION_OPTICAL_SIGNATURE 光学特性多分辨率容器
	WSF_MULTIRESOLUTION_PROCESSOR 处理器多分辨率容器
	WSF_MULTIRESOLUTION_RADAR_SIGNATURE 雷达特性多分辨率容器

	WSF_MULTIRESOLUTION_SENSOR 传感器多分辨率容器
处理器模型	WSF_SA_PROCESSOR 态势感知处理器
轨道传播模型	WSF_KEPLERIAN_PROPAGATOR 开普勒轨道传播模型
	WSF_J2_PERTURBATION_PROPAGATOR J2 轨道传播模型
	WSF_NORAD_PROPAGATOR-两行根数轨道传播模型
	WSF_INTEGRATING_PROPAGATOR-集成器轨道传播模型
	AMTI-雷达传感器 UCI 组件 COMPUTER-UCI 计算组件 ESM-ESM-传感器 UCI 组件 IRST-IRST-传感器 UCI 组件 WEAPON-武器 UCI 组件
武器模型	WSF_P6DOF_EXPLICIT_WEAPON 伪 6 自由度武器模型
	WSF_SIX_DOF_EXPLICIT_WEAPON 6 自由度武器模型

AP3.3. 功能更新

2.9 相比 2.6 增加的功能较多，包括添加 8 条主要功能，对原有功能不足之处的修复，以及细节上的功能增加及优化。下面将简述一些功能更新。

AP3.3.1. ACES 空战视图

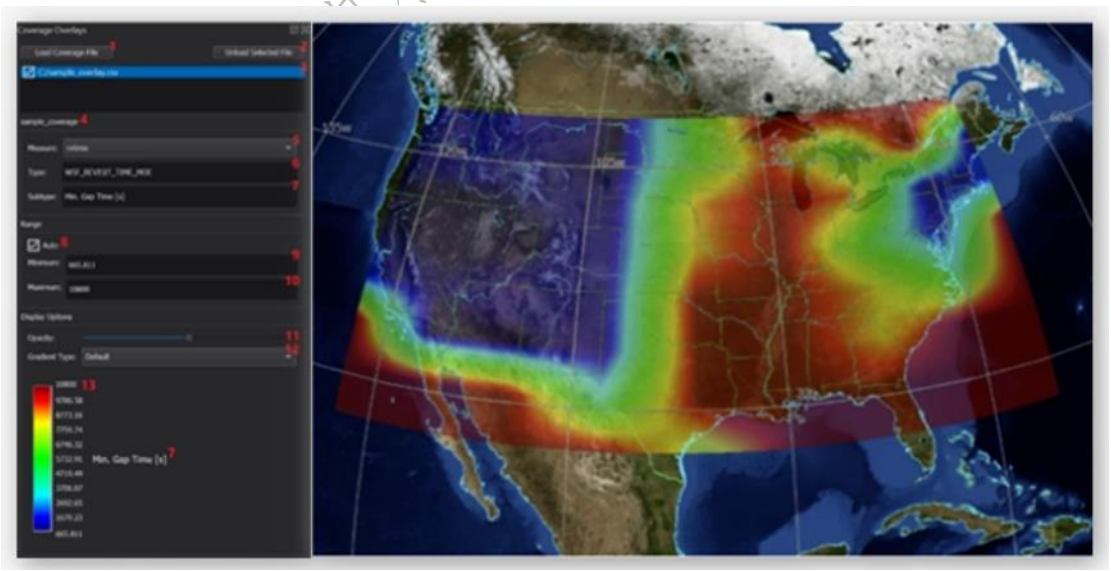
空战交战摘要(ACES)显示器旨在提供一个集成显示器，为平台提供多组空战数据。在单个可重新配置的显示器中包含一个 WSF_SA_PROCESSOR。

AP3.3.2. 态势感知视图

SA Display(态势感知显示)为用户提供平台战术感知的图形摘要。

AP3.3.3. 覆盖叠加工具

覆盖叠加工具在地图显示中以视觉方式显示覆盖数据。该工具适用于覆盖叠加文件，其中可能包含多个数据字段。通过一系列可用的渐变方式，数据值被映射成不同的颜色。

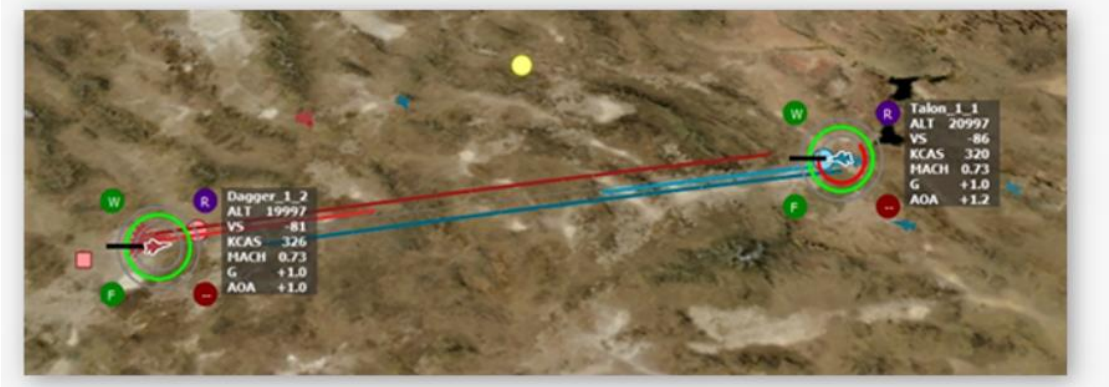


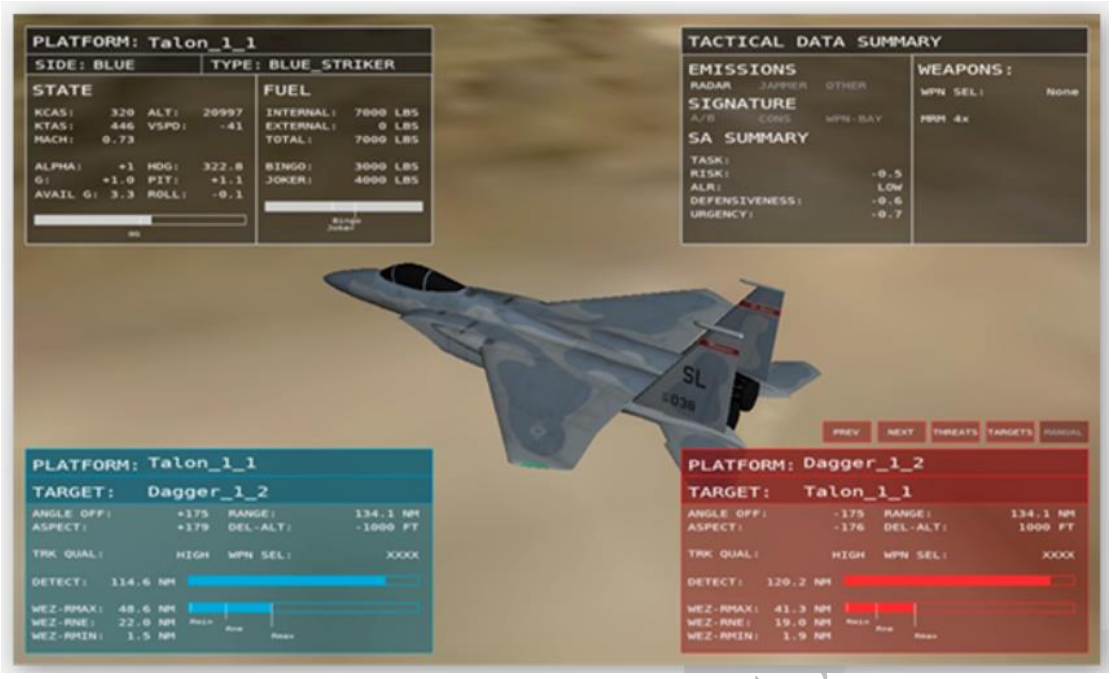
AP3.3.4. 通信可视化

该工具为用户提供了一种基于 GUI 的方法可视化通信/网络配置。

AP3.3.5. 空战可视化

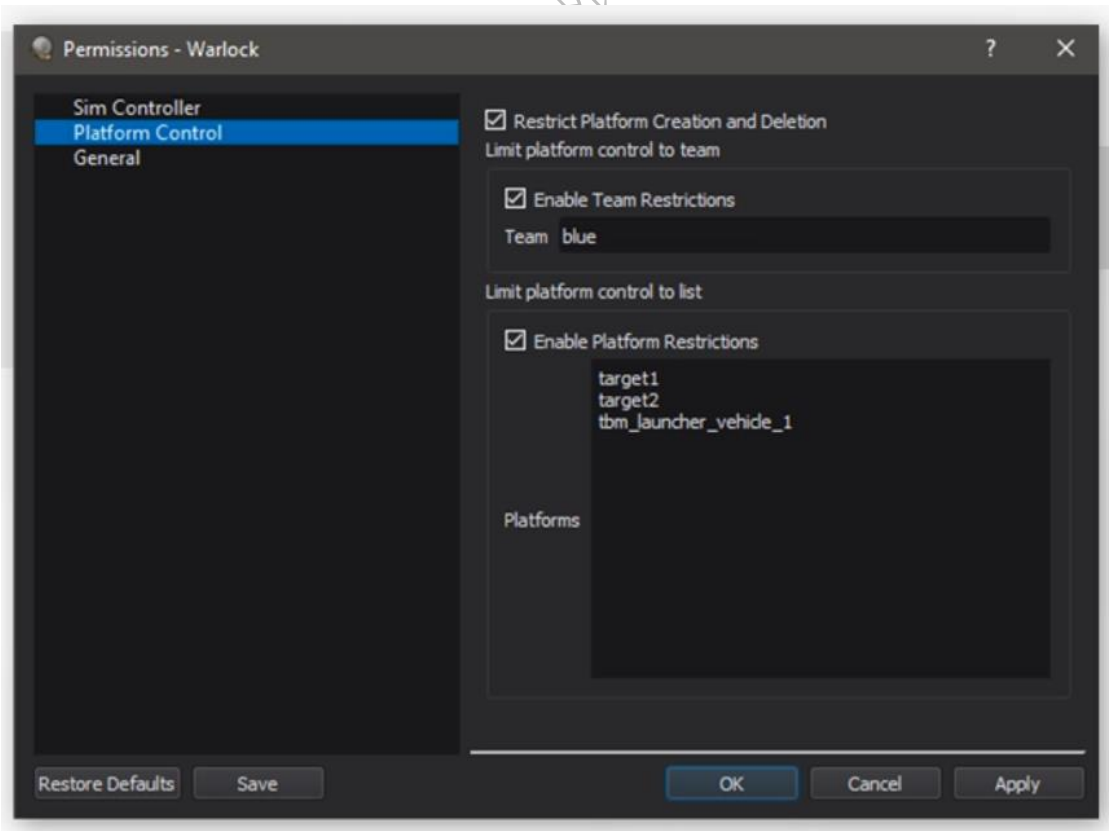
“空中战斗可视化”插件提供了多种可视化与空中战斗相关数据的方法，包括:数据环、数据修饰、状态数据、侦测/WEZ 线、空中战斗叠加层。当选择相关平台时，数据环、数据修饰、状态数据和侦测/WEZ 线将显示在地图显示中。如果平台具有 WSF_SA_PROCESSOR，则被视为相关平台。





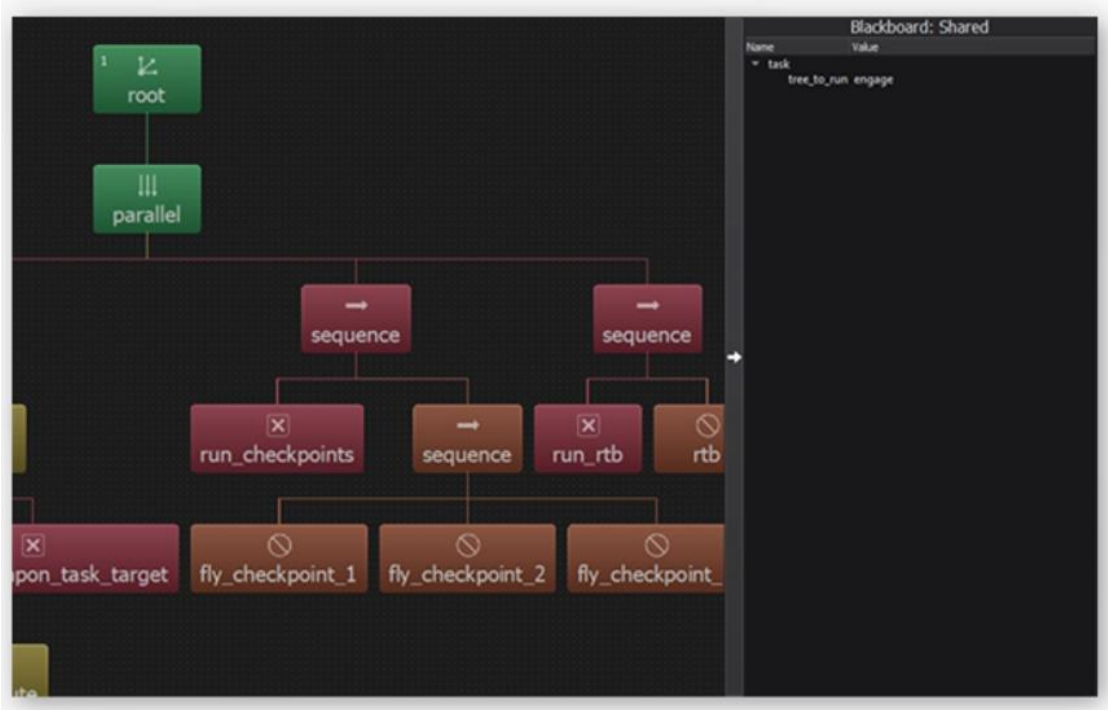
AP3.3.6. 权限管理器

“权限管理器”对话框显示有关用户修改 warlock 和仿真的能力的信息。允许哪些团队可见兵棋推演线索控制，哪些平台是用户可以控制的。



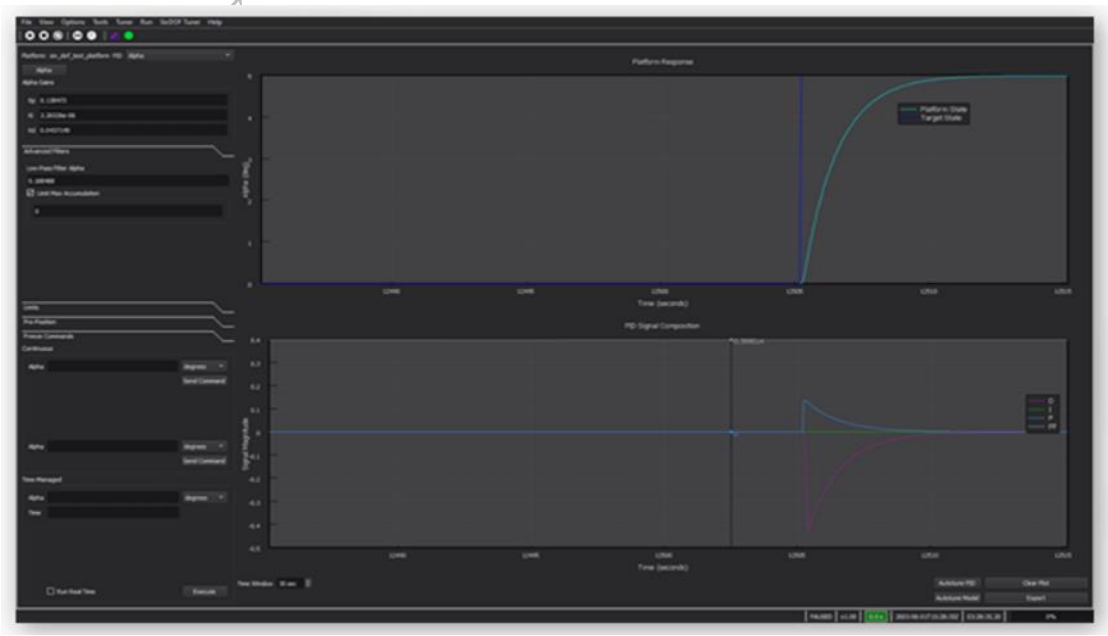
AP3.3.7. 高级行为树

添加了右键单击节点并最小化子节点的功能，以及查看节点是否为树、是否隐藏了其子节点以及节点当前在行为树视图中有多少个黑板变量的方法。



AP3.3.8. 六自由度调谐器

P6Dof 调谐器工具可帮助用户调整 P6Dof 车辆自动驾驶仪使用的比例、积分、微分 (PID)控制器。SixDOF 调谐器工具可帮助用户调整 SixDOF 导航自动舵使用的比例、积分、微分 (PID)控制器。



AP3.3.9. 3D 模型

添加了 J-20、AH-64、UH-60、MH-60、M-1、M-2、悍马、通用卡车以及 S-300 地空导弹系统(SA-10、SA-12、SA-20)的各种雷达、TEL 和指挥车的 3D 模型。

AP3.4. 脚本更新

- 添加和修改了各种脚本方法，并扩展了对高级行为树的黑板变量支持。
- 添加了 WsfPlatform.RadarCrossSection 的一个版本允许用户使用接收机和发射机纵横比角度查询平台的双基地 RCS。
- 向 Filel0 添加了其他功能，供用户查询打开文件的模式和路径信息。
- 添加了新的脚本方法 KeySet 以访问 Map 的完整密钥集添加了用于查询 WsfGeoPoint 的平均太阳时的方法 WsfGeoPoint.ApparentTimeNow 和 WsfGeoPoint.ApparentTime.
- 添加了使用新的视场类层次结构动态更改传感器上的视场的功能 WsfFieldOfView、WsfCircularFieldOfView、WsfRectangularFieldOfView、WsfPolygonalFieldofView 和 WsfEquatorialFieldOfView。
- 添加了绝对误差模型，该模型定义了围绕传感器检测目标的两维或三维的一标准差绝对位置误差测量。
- 添加了 Sun.Elevation 和 Sun.Azimuth，用于查询有关太阳位置的信息。
- 添加了对通过 OnSensorDetectionAttempt 方法的脚本化传感器检测约束的支持。传感器现在还可以利用通用脚本接口的功能。
- 添加了新的脚本方法 WsfEM_XmtrRcvr.Index 和别名 WsfEM_Xmtr.BeamNumber.
- 添加了脚本方法 WsfAntennaPattern.AzimuthBeamwidth 和 WsfAntennaPattern.Elevation

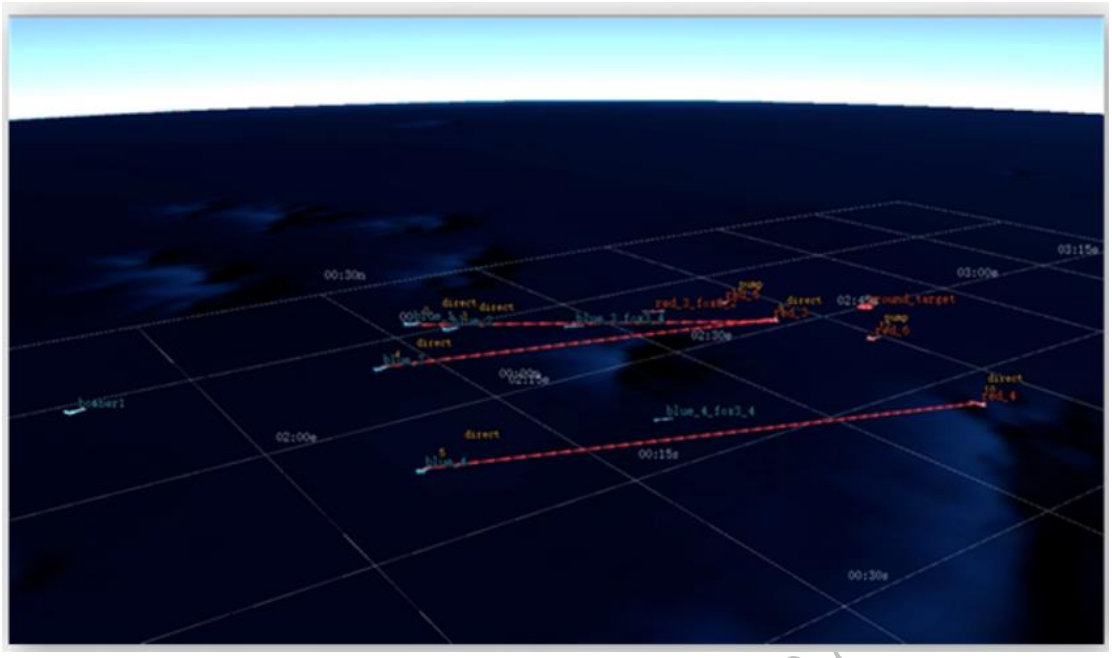
Beamwidth 的变体，以考虑电子波束控制。

- 添加了克隆 script_struct 的功能。
- 添加了新的传感器调度器旋转，以模拟旋转雷达。
- 添加了向覆盖率计算添加访问间隔持续时间约束的功能。
- 添加了新的效果 WSF_CYBER_DETONATE_EFFECT，可以引爆平台上给定名称或类型的所有武器。
- ·为网络攻击添加了称为归因的新阶段。当随机抽奖成功时，平台在名义上已经追踪了攻击的来源。
- 向 WSF_CYBER_MANIN_THE_MIDDLE_EFFECT 添加了 exfiltrate 命令，以允许将通信消息泄露回攻击者。
- 添加了具有相应类 WsfDriftManeuver 的 Drift 轨道机动。
- 添加了具有相应类 WsfTeardropManeuver 的 Teardrop 轨道机动。
- 添加了使用轨道传播的空间轨迹位置和速度外推。
- 向 WsfSpaceMover 添加了用于计算与空间任务相关的常见几何量的方法。
- 添加了在指定机动目标时和通过 WsfTargetPoint 定位特定运动学状态的功能。
- 添加了 WsfSpaceMover.InitialHeading，它从卫星的初始速度状态计算卫星的航向。
- 增加了对 SA(态势感知)处理器的支持，以感知导弹/武器，改善整体 SA 并提供区分飞机威胁和导弹威胁的方法。
- 为 WSF SA PROCESSOR 实施了新的感知延迟设置，通过延迟新轨道向感知实体状态的过渡来模拟人类认知。
- 添加了命令 align_heading_with_velocity，启用后，将 WSF GUIDED_MOVER 或 WSF UNGUIDEDMOVER 航向设置为匹配所属平台的速度矢量，而不是方向。

AP3.5. 示例更新

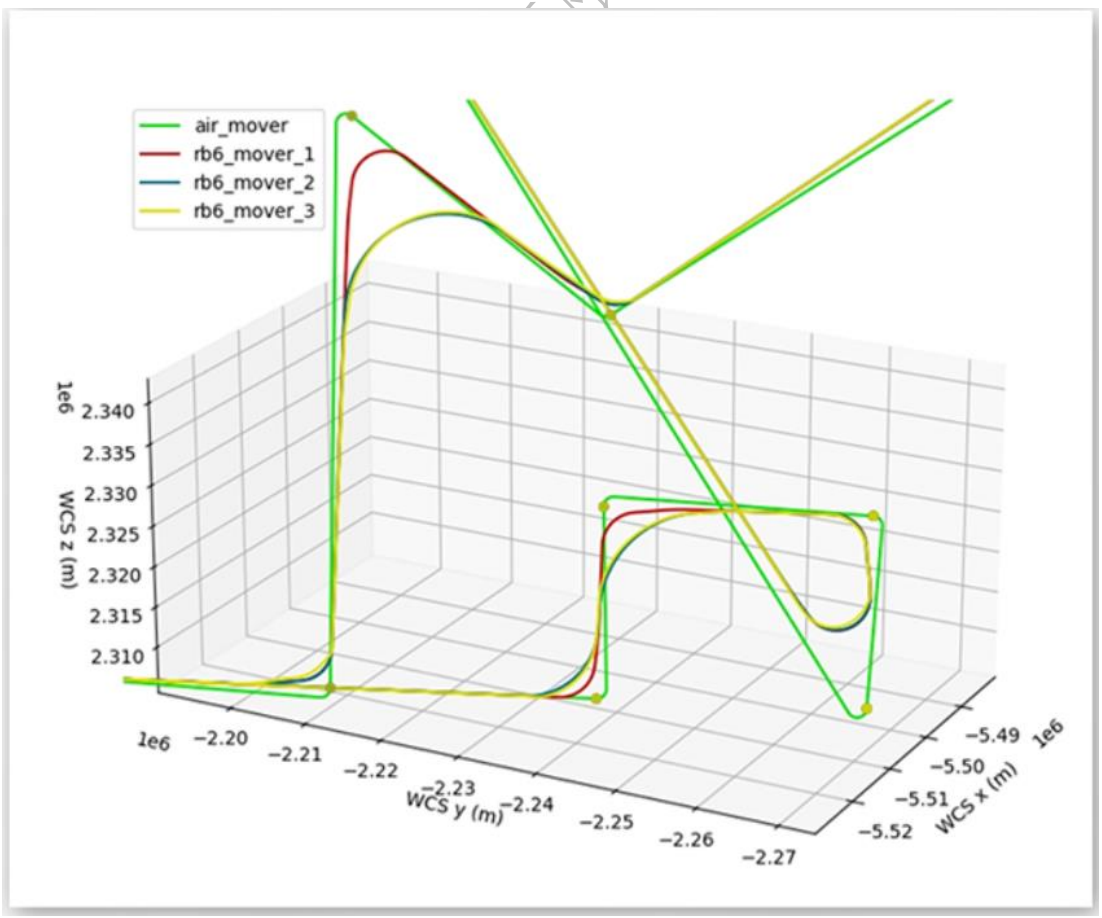
AP3.5.1. 空对空战斗

空对空交战示例集合，包括一对一交战、二对二交战（使用不同推进器）、高价值机载资产（HVAA）保护/防御性反空（DCA）任务、轰炸机护航任务。



AP3.5.2. 多分辨率组件

包括演示多分辨率移动器、传感器、通信、处理器、燃料、光学特征和雷达特征的示例。每个方案都相当简单，可以演示如何设置多分辨率组件。这些场景旨在说明多分辨率的灵活性。



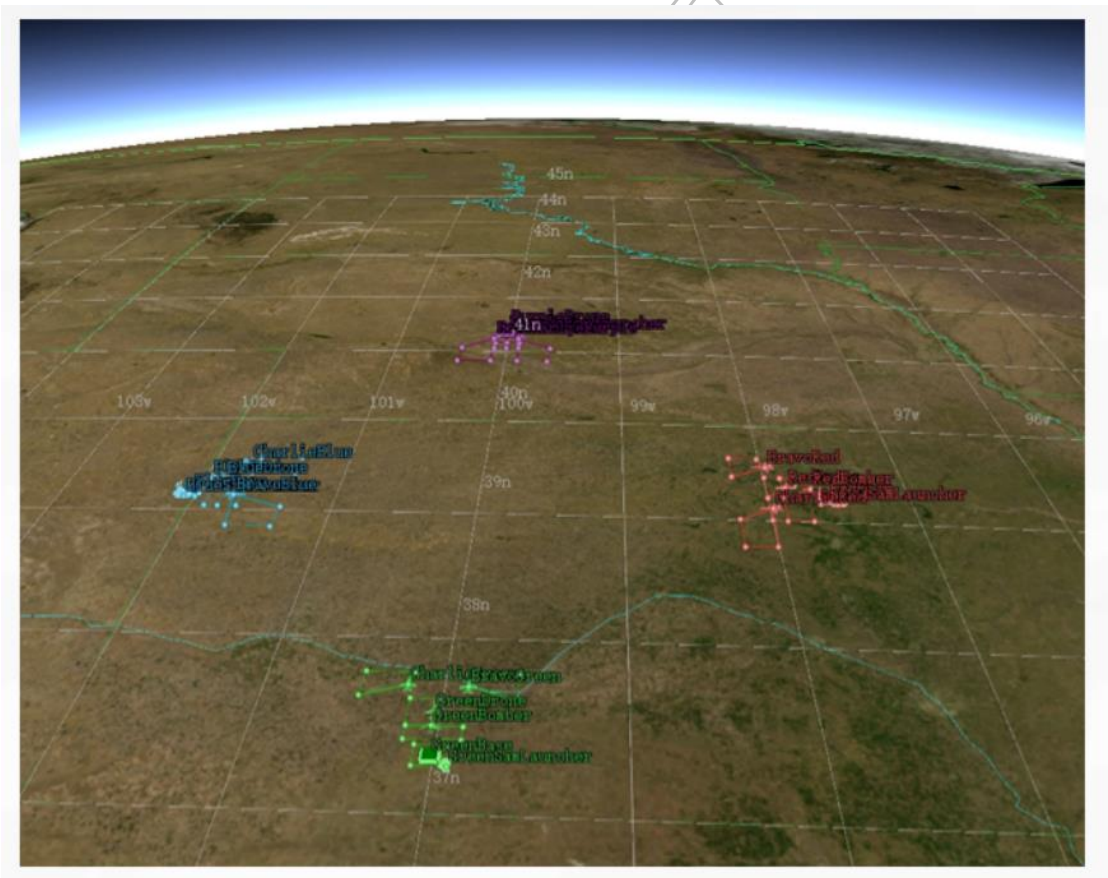
AP3.5.3. 视觉组件及分布式

展示了 WSF_VISUAL_PART 的使用，以及将关节部件发布到 DIS 的过程。有两辆带有可视部件的坦克，一个脚本处理器负责更新这些部件，是他们指向一架飞机，这是通过真实数据实现的。dis_interface 模块允许可视部件（或任何关节部件）通过 DIS 实体状态 PDU 进行发布。

AP3.5.4. 多席位战争游戏

这个场景最适合多名玩家参与，分别控制四个可玩团队。这四个团队虽然颜色不同，但拥有相同的资源。在启动时，资源将遵循预设的初始路线，直到收到其他指令。每个玩家可用的资源包括:两架战斗机、一架地对空导弹发射器、一架轰炸机、一架侦察无人机以及一个基地。

玩家的目标是尽可能利用自己的资源，摧毁对手的基地，同时保卫自己的基地。一旦基地被摧毁，玩家将失去该场景中剩余的所有平台，并收到“游戏结束”的提示。最后保留基地完好无损的玩家获胜!



附录四（推进器）

AP4.1. Route 类型

这些移动器是简化版的，可以通过定义带有航点的路线来模拟它们在仿真过程中从位置到位置的移动。在平台上定义的限制会约束它们的移动。平台的移动是基于数学而不是平台的空气动力学或质量特性。

⋆WSF_GROUND_MOVER 地面车辆推进器

1. 概述

WSF_GROUND_MOVER 是一种专门用于模拟地面车辆或实体移动的移动器（mover）类型。这种移动器适用于模拟坦克、装甲车、步兵战车、运输车辆等地面军事资产或民用车辆的行为。

2. 典型用途和特点

地面机动：模拟车辆在各种地形上的移动，包括公路、越野、城市环境等。

战术行为：模拟车辆执行战术机动，如快速部署、战术撤退、巡逻路线等。

编队控制：在车队或装甲纵队中协调多辆车辆的移动，保持特定的队形和间距。

交战模拟：模拟车辆在战斗中的移动，包括向敌方目标移动、规避敌方火力等。

障碍物规避：模拟车辆在遇到障碍物时的规避行为，如绕行、穿越或摧毁障碍物。

地形适应：根据地形的变化调整车辆的速度和路线，以模拟不同地形对机动性的影响。

载具性能：模拟车辆的物理性能，如最大速度、加速度、转向半径和越野能力。

后勤支持：模拟车辆在后勤任务中的角色，如补给运输、伤员撤离等。

3. 推进器基本指令

update_interval <A>

参数列表	参数类型	参数含义
A	时间	更新时间间隔

模拟流程周期性调用此推进器的间隔时间，为 0 时只有在需要确认平台位置时才会调用。

默认值：0 s

update_time_tolerance <A>

参数列表	参数类型	参数含义

A	时间	最小更新时间间隔

模拟流程允许请求推进器更新位置的最小间隔时间。

默认值：推进器以标准速度行驶 1m 所需的时间。

4. 航路相关推进器指令

altitude_offset A>

参数列表	参数类型	参数含义
A	长度值	

at_end_of_path [extrapolate | stop | remove]

指定当移动器到达其定义路径的终点时要采取的操作。

参数列表	可选参数	参数含义
A	extrapolate	继续前进
	stop	停止
	remove	移除平台

默认值：extrapolate

draw_route <A>

参数列表	参数类型	参数含义
A	布尔值	是否启用功能

将会绘制出平台所在的航路。

on_turn_failure <A>

参数列表	可选参数	参数含义
A	best_effort	尽力而为以达到可能达到的最近接近点
	ignore_point	忽略该航路点
	reverse_turn	反向转向以便能够抵达该点

定义了在推进器由于转向半径的限制无法抵达某个航路点时的行为。

默认值：best_effort

pathfinder <A>

参数列表	参数类型	参数含义
A	字符串	路径查找器名称

定义物体使用的路径查找器名称。

print_route <A>

参数列表	参数类型	参数含义
A	布尔值	是否启用该功能

在航路被修改时输出航路数据，并且绘制航路的可视化图像。

start_at <A>

参数列表	参数类型	参数含义
A	字符串	航路点标签

平台将从指定的航路点开始航路。

start_time <A>

参数列表	参数类型	参数含义
A	时间	移动开始时间

平台将在指定的时间开始移动。

switch_on_approach

在平台接近当前目标航路点的一个转弯半径范围内时，推进器会自动切换到下一个路径点。

switch_on_passing

推进器只在平台与当前目标点并排（到达或超过）时切换到下一个路径点。

turn_failure_threshold <A>

参数列表	参数类型	参数含义
A	浮点数	失败阈值比率

定义了 on_turn_failure 行为触发的阈值，该阈值取决于转弯半径的比例。例如，如果转弯半径为 1000 米，turn_failure_threshold 为 0.01，则当航路点的偏差超过 10 米时，on_turn_failure 逻辑就会被触发。

默认值：0.01

use_route <A>

参数列表	参数类型	参数含义
A	字符串	航路名称

为推进器指定航路。

5. 路径点相关推进器指令

angle_of_attack_table

格式：

```
angle_of_attack_table
    altitude <A1>
        speed <B1> angle <C1> speed <B2> angle <C2>
    altitude <A2>
        angle <C3>
end_angle_of_attack_table
```

参数列表	参数类型	参数含义
A	长度	高度值
B	速度	速度值
C	角度	旋转角度

定义了一个关于迎角（Angle of Attach，AoA）的表格，用于指定在不同高度和速度下有效的迎角值。

altitude <A>

参数列表	参数类型	参数含义

A	<altitude-value>	海拔高度

指定后续数据有效的海拔高度。海拔块必须按数值递增顺序排列。使用海拔块的线性插值。

speed A>

参数列表	参数类型	参数含义
A	<speed-value>	速度

指定列出的迎角数据有效的速度。速度条目必须按数值递增的顺序排列。将使用速度数据的线性插值来计算迎角。

angle A>

参数列表	参数类型	参数含义
A	<angle-value>	迎角角度

迎角。迎角条目必须按数值递增的顺序排列。

bank_angle_limit <A>

参数列表	参数类型	参数含义
A	<angle-value>	滚转角限制

滚转角限制。值必须在 0 度到 85 度之间。用于计算最大径向加速度。

默认值：0

body_g_limit A>

参数列表	参数类型	参数含义
A	<acceleration-value>	加速度限制值

实体 g 限制。值必须大于地球引力的加速度。

heading_pursuit_gain <A>

参数列表	参数类型	参数含义
A	<double-value>	航向追踪增益

默认值:5

maximum_climb_rate A>

参数列表	参数类型	参数含义
A	<speed-value>	改变高度时使用的最大爬升率和俯冲率。

指定在改变高度时使用的最大爬升率和俯冲率。航点或脚本指定的其他爬升率受此值的限制。

注意：移动器的实际爬升率也会受到 maximum_flight_path_angle 的影响。

maximum_flight_path_angle A>

参数列表	参数类型	参数含义
A	<angle-value>	指定最大飞行航迹角(爬升/俯冲角)

指定最大飞行航迹角（爬升/俯冲角）。值必须大于或等于 0。

注意:移动器的实际爬升率也会受到 maximum_climb_rate 的影响。

默认值:0

maximum_linear_acceleration A>

参数列表	参数类型	参数含义
A	<acceleration-value>	指定在需要加速时使用的最大线性加速度。如果航点没有包含线性加速度规范,则使用此值。

默认值；6 g's

maximum_radial_acceleration A>

参数列表	参数类型	参数含义
A	<acceleration-value>	指定在转弯时使用的最大径向加速度。如果航点没有包含径向加速度规范,则使用此值。

默认值；6 g's

maximum_altitude <A>

参数列表	参数类型	参数含义
A	<altitude-value>	最大高度限制

minimum_altitude <A>

参数列表	参数类型	参数含义
A	<altitude-value>	最小高度限制。

maximum_speed A>

参数列表	参数类型	参数含义

A	<speed-value>	最大速度限制。值必须大于 0。

minimum_speed A>

参数列表	参数类型	参数含义
A	<speed-value>	最小速度限制。值必须大于或等于 0。

默认值：0.0

path_variance_radius A>

参数列表	参数类型	参数含义
A	<length-value>	

这个值会在给定的半径内随机变化航点的位置。在选择随机方位角和距离后，应用于计算路径时的下一个航点。

roll_rate_limit A>

参数列表	参数类型	参数含义
A	<angle-rate-value>	滚转率限制

滚转率限制。值必须大于 0。注意：当应用于 WSF_AIR_MOVER 或其他类型的航点移动器时，滚转率不会影响平台沿路线的运动。要影响路线跟随行为，请使用最大径向加速度。

speed_variance_percent <A>

参数列表	参数类型	参数含义
A	<percent-value>	这个值会在给定的百分比范围内随机变化移动器在每个航点的速度。

这个值会在给定的百分比范围内随机变化移动器在每个航点的速度。值必须大于 0。这意味着移动器在通过每个航点时，其速度会在这个百分比的正负范围内随机变化，从而为仿真增加一定的随机性和不可预测性。

turn_rate_limit A>

参数列表	参数类型	参数含义
A	<angle-rate-value>	转弯率限制。值必须大于 0。

pitch_disable

用于禁用飞行器的俯仰（pitch）控制。当这个设置被激活时，飞行器将不会响应任何改变其俯仰角度的命令或输入，这意味着它将保持当前的俯仰姿态，即使在执行某些机动或遇

到外部扰动时也是如此。

no_pitch

限制移动器使平台俯仰。这对运动学没有影响。

on_road

限制移动器使平台翻滚。这对运动学没有影响。

off_road

关闭“on_road”选项。这允许平台进行翻滚。

path_compute_timestep

航点移动器会预先计算沿路径的移动。这种行为强制转弯具有恒定的半径。如果将 path_compute_timestep 指定为正值，那么如果速度有任何变化，转弯率将在这个时间间隔上更新。

⋆WSF_AIR_MOVER 简易飞行器推进器

1. 应用环境

·不需要为平台提供质量属性、空气动力学或推进力

·运动效果直接基于为平台设置的最大限制

·只适用于水平面内的连续移动，垂直过渡会是不连续的

2. 推进器基本指令

update_interval <A>

参数列表	参数类型	参数含义
A	时间	更新时间间隔

模拟流程周期性调用此推进器的间隔时间，为 0 时只有在需要确认平台位置时才会调用。

默认值：0 s

update_time_tolerance <A>

参数列表	参数类型	参数含义
A	时间	最小更新时间间隔

模拟流程允许请求推进器更新位置的最小间隔时间。

默认值：推进器以标准速度行驶 1m 所需的时间。

3. 航路相关推进器指令

use_route <A>

参数列表	参数类型	参数含义
A	字符串	航路名称

为推进器指定航路。

altitude_offset <A>

参数列表	参数类型	参数含义
A	长度	高度偏移量

平台相对于航路点的高度偏移量。

start_at <A>

参数列表	参数类型	参数含义
A	字符串	航路点标签

平台将从指定的航路点开始航路。

start_time <A>

参数列表	参数类型	参数含义
A	时间	移动开始时间

平台将在指定的时间开始移动。

pathfinder <A>

参数列表	参数类型	参数含义
A	字符串	路径查找器名称

定义物体使用的路径查找器名称。

print_route <A>

参数列表	参数类型	参数含义
A	布尔值	是否启用该功能

在航路被修改时输出航路数据，并且绘制航路的可视化图像。

draw_route <A>

参数列表	参数类型	参数含义
A	布尔值	是否启用功能

将会绘制出平台所在的航路。

at_end_of_path <A>

参数列表	可选参数	参数含义
A	extrapolate	继续前进
	stop	停止
	remove	移除平台

设置平台抵达航路末尾时的行为。

默认值：extrapolate

switch_on_approach

在平台接近当前目标航路点的一个转弯半径范围内时，推进器会自动切换到下一个路径点。

switch_on_passing

推进器只在平台与当前目标点并排（到达或超过）时切换到下一个路径点。

turn_failure_threshold <A>

参数列表	参数类型	参数含义
A	浮点数	失败阈值比率

定义了 on_turn_failure 行为触发的阈值，该阈值取决于转弯半径的比例。例如，如果转弯半径为 1000 米，turn_failure_threshold 为 0.01，则当航路点的偏差超过 10 米时，on_turn_failure 逻辑就会被触发。

on_turn_failure <A>

参数列表	可选参数	参数含义
A	best_effort	尽力而为以达到可能达到的最近接近点
	ignore_point	忽略该航路点
	reverse_turn	反向转向以便能够抵达该点

定义了在推进器由于转向半径的限制无法抵达某个航路点时的行为。

默认值：best_effort

4. 路径点相关推进器指令

指令与⋆WSF_GROUND_MOVER 地面车辆推进器指令相同。

5. 飞行器相关推进器指令

maximum_impact_speed <A>

参数列表	参数类型	参数含义
A	速度值	与平台相关的最大速度

指定与平台相关的最大速度，超过这个速度时，平台与地形相交并被认为是“坠毁到地面”。坠毁的平台将通过 WsfSimulationObserver::CrashedIntoGround() 方法通知观察者，并从仿真中移除自己。如果撞击速度低于这个最大值，则平台被认为是“着陆”。默认行为是总是着陆，而不是坠毁。

⭐WSF_KINEMATIC_MOVER（可用于鱼雷）

1. 概述

WSF_KINEMATIC_MOVER 是一种移动器（mover）类型，用于模拟平台的预定或关键帧动画运动。这种移动器特别适用于那些运动路径已经预先定义好，且不需要实时物理计算的场景。

2. 典型用途和特点

预定义路径：允许平台沿着一系列预定义的点或路径移动，这些点可以通过关键帧或路径数据来指定。

动画控制：可以用于控制平台的动画序列，使其按照特定的时间表或触发条件进行。

非物理驱动：与物理驱动的移动器不同，WSF_KINEMATIC_MOVER 不依赖于物理引擎来计算运动，而是根据预设的路径和时间点来移动平台。

简单快捷：对于不需要复杂物理交互的场景，如某些训练模拟或演示，这种移动器提供了一种简单快捷的解决方案。

易于控制：用户可以轻松地控制平台的运动，包括速度、方向和加速度，而不必担心物理碰撞或动力学效应。

脚本化：可以通过脚本精确地控制平台的运动，使其在特定事件或条件下执行预定的移动。

适用于演示和训练：特别适合于需要展示特定操作或程序的演示和训练场景。

2. 指令

detailed_debug A>详细调试

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否启用

启用向标准输出的 debug 调试输出。

prefer_canopy_up A>优先保持顶盖朝上

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否启用

使平台始终向局部垂直方向翻滚。与 bank_to_turn 互斥。

bank_to_turn A>倾斜转弯

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否启用

使得平台在不进行转弯加速时，向加速向量方向翻滚，但仍然优先保持垂直姿态。与 prefe r_canopy_up 互斥。

broach_at_sea_level A>在海平面上浮现

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否启用

这个移动器最初是为了定义一个鱼雷移动器而开发的。然而，该模型的功能比较强大，以至于它可以在海平面以上使用。这个标志确保一旦进入水下，运动必须保持在本地海平面以下。

target_speed A>目标速度

参数列表	参数类型	参数含义
A	速度值<speed-value>	目标速度

一旦提供，平台将加速或减速以匹配目标（期望）速度，但要受到最大线性加速度限制的约束。

initial_speed A>初始线速度

参数列表	参数类型	参数含义
A	速度值<speed-value>	初始线性速度

初始线性速度。初始化后，速度会变化以保持目标速度。

initial_flight_path_angle A>初始飞行路径角

参数列表	参数类型	参数含义
A	角度值<angle-value>	初始飞行路径角

初始飞行路径角。初始化后，飞行路径角将变化以引导至定义路线中的期望航点。

maximum_linear_acceleration A>最大线性加速度限制

参数列表	参数类型	参数含义
A	加速度值<acceleration-value>	最大线性加速度限制

定义最大线性加速度限制。

默认值：0.25G

maximum_radial_acceleration A>径向加速度限制约束

参数列表	参数类型	参数含义
A	加速度值<acceleration-value>	径向加速度限制约束

定义了径向（垂直于速度方向）加速度限制约束。这个约束与最大机体转向率同时施加，采用其中最严格的。

默认值：8.0G

maximum_body_roll_rate <A>最大机体滚转率

参数列表	参数类型	参数含义
A	角速率值<angular-rate-value>	平台滚转机动最大角速度

定义了平台尝试捕获期望目标倾斜角的最大速率。平台在进行滚转机动时所能达到的最大角速度。

默认值：180 deg/sec

maximum_body_turn_rate <A>最大机体转向率

参数列表	参数类型	参数含义
A	角速率值<angular-rate-value>	最大机体转向率

定义了速度矢量在三维空间中旋转的最大速率。这个约束与最大径向加速度同时施加，采用其中最严格的。

默认值：45 deg/sec

velocity_pursuit_gain <A>速度追踪增益

参数列表	参数类型	参数含义
A	非负值<non-negative-value>	速度追踪增益值

定义了目标方位角和仰角（以弧度为单位）与应用的横向或垂直加速度（以米每秒平方为单位）之间的比例因子，以将速度矢量对准目标。

默认值：4.0

waypoint_switch_on_ground_turning_radius A>地面转弯半径航点切换

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否设置

如果此输入值设置为 true，则在决定是否前进到路线中的下一个航点时，只考虑水平偏移；垂直误差值将被忽略。如果设置为 false，则在决定一个航点是否足够接近以被视为“命中”时，会考虑三维斜向偏移。

默认值：true

proportional_navigation_gain <A>比例导航增益

参数列表	参数类型	参数含义
A	非负值<non-negative-value>	比例导航增益值

定义了目标视线速率（以弧度/秒计）与用于将速度矢量抵消至拦截目标未来位置的横向或垂直加速度（以米/秒平方计）之间的比例因子。在决定时，同时考虑最大径向加速度，采用其中最严格的限制。

默认值：40.0

use_route <A>

参数列表	参数类型	参数含义
A	字符串	航路名称

为推进器指定航路。

WSF_ROAD_MOVER

1. 概述

WSF_ROAD_MOVER 是 WSF_GROUND_MOVER 的一个特殊化版本，它在道路网络上移动。它计算起始点和终点之间的最短路径，并将其作为其航点路径。可以设置一个暂停时间来偏移移动器的开始时间。此外，可以设置一个标志（use_closest_waypoint），使得移动器根据用户指定的起始和结束位置计算基于最近航点的最短路径。

⭐WSF_ROTORCRAFT_MOVER 旋翼飞行器路线移动器

1. 概述

WSF_ROTORCRAFT_MOVER 是一种路线移动器，用于模拟旋翼飞行器的运动特性。它允许将期望的平台朝向（即机体指向的方向）与速度向量分离。在低速时，平台可以采取任何期望的朝向，无论其平移方向如何；但在高速时，朝向会与飞行方向一致。对期望朝向或航向（机体指向方向）的脚本命令被视为请求，但可能不会立即执行。移动器将维持一个期望的航向角，在速度降低到风向标速度以下时，将再次调整至期望的航向。当平台在横向加速或转弯时，其旋翼平面在 North-East-Down（NED）坐标系中的朝向会倾斜以指向加速度向量，但在巡航状态下则保持水平。

WSF_ROTORCRAFT_MOVER 是首个实现移动器“模式”的 WSF 移动器类型，参见下面的模式命令。当移动器模式改变时，无论是通过脚本还是通过航点，所有运动属性都会变为该运动模式下定义的属性。例如，脚本命令 ‘PLATFORM.Mover().SetMode(“CRUISE”)’ 会立即采用在“CRUISE”运动模式下预定义的速度、爬升率和加速度值。可以为旋翼飞行器移动器定义适合“悬停”、“盘旋”、“冲刺”或任何其他期望的移动器模式的模式。如果移动器没有定义该模式，则该命令将被忽略。

WSF_ROTORCRAFT_MOVER 类型中的移动器模式是一种允许用户定义旋翼飞行器在不同飞行状态下的行为特征的方法。这些模式可以包括悬停（HOVER）、盘旋（LOITER）、冲刺（DASH）等，每种模式都有其特定的速度、爬升率、加速度和其他运动属性。理解移动器模式的关键在于以下几个方面：

模式定义：用户可以根据旋翼飞行器的预期用途定义不同的飞行模式。每种模式都有一组预设的运动参数，如速度、加速度、爬升率等。

模式切换：在飞行过程中，可以通过脚本命令或到达特定的航点来切换移动器模式。例如，当旋翼飞行器到达一个需要悬停的点时，可以切换到“悬停”模式。

属性变化：当模式切换发生时，旋翼飞行器的所有运动属性会立即更新为新模式下定义的属性。这意味着飞行器的行为会根据当前模式进行调整。

命令执行：在不同的模式下，对移动器的命令（如改变航向或速度）可能会有不同的响应。例如，在“悬停”模式下，飞行器可能会更快速地响应航向改变的命令。

默认行为：如果移动器接收到一个未定义的模式命令，它将忽略该命令并继续执行当前模式下的行为。

灵活性：移动器模式提供了一种灵活的方式来模拟旋翼飞行器在不同飞行条件下的行为，使得模拟更加真实和可控。

2. 旋翼机移动器命令

desired_heading A>

参数列表	参数类型	参数含义
A	角度值<angle-value>	航向

为移动者设置所需的航向，该航向将在低于 “weathercocking speed” 的速度下生效。

position_hold_capture_radius A>

参数列表	参数类型	参数含义
A	长度值<length-value>	

在 AFSIM 仿真系统中，旋翼飞行器（Rotorcraft）在从一个位置转移到另一个位置时，通常会使用速率引导机制。当接近目标点时，引导机制会根本性地改变，以便捕获一个位置值，并且必须提前减速，以准备捕获和保持目标位置。这个值决定了从一种引导类型到另一种引导类型的转换点。默认值设定为 200 米，这应该足以满足大多数旋翼飞行器模型的需求。

这种机制确保了旋翼飞行器在到达目标点之前能够平稳地减速，以实现精确的位置保持。在 AFSIM 中，这种转换点的设置是为了模拟现实中旋翼飞行器在执行任务时的行为，例如在接近目标地点时的飞行控制和导航策略。通过调整这个值，仿真专家可以模拟不同情况下的飞行器行为，以评估其性能和操作效率。

start_mode <A>

参数列表	参数类型	参数含义
A	字符串<mode-name>	模式名称

在 AFSIM 仿真系统中，平台的初始移动器模式可以在仿真开始时通过脚本设置指定。随着仿真的进行，移动器模式可以通过外部脚本或在航点转换时进行更改。移动器模式控制平台的运动行为，例如飞行、驾驶或步行等。这些模式是 AFSIM 中预定义的。

ned_filter_time_constant A>

参数列表	参数类型	参数含义

A	时间值<time-value>	滤波时间常数

指定一个滤波时间常数，用于施加于受控加速度值，以平滑瞬态响应。较大的值会对指令施加更多的平滑效果。该值必须大于零。默认值为 1 秒，对大多数情况而言已经足够。

altitude_error_to_rate_of_climb_gain <A>

参数列表	参数类型	参数含义
A	浮点值<floating-point-value>	增益值大小

指定一个增益值，用于将高度误差转换为垂直加速度值。默认值为 1，对大多数用途来说已经足够。

vertical_acceleration_rate_pid … end_vertical_acceleration_rate_pid

vertical_acceleration_value_pid … end_vertical_acceleration_value_pid

lateral_acceleration_rate_pid … end_lateral_acceleration_rate_pid

lateral_acceleration_value_pid … end_lateral_acceleration_value_pid

上述四个输入块中的每一个都可能包含用于 Proportional Integral Derivative 比例积分导数（PID）控制器的调整参数，这些控制器决定了驱动平台运动状态的加速度值。除非需要特定的动态响应，否则最终用户没有必要修改默认增益。调整 PID 控制器超出了本文档的范围，但有许多其他资源可以提供帮助。对于大多数目的而言，PID 控制器的响应应该是足够的。

有效命令，参阅 PID 控制器命令。

3. 模式命令

mode<name> … end_mode 模式块

weathercock_speed <A>

参数列表	参数类型	参数含义
A	速度值<speed-value>	风标速度

当平台的地速超过设定值时，它将假设一个与航线角度相匹配的航向。在这个速度以下，平台将假设一个由脚本命令设定的理想航向角度。这意味着平台在低速时会保持一个预设的航向，而当达到一定的速度阈值时，它会调整自己的航向以跟随既定的路径或航线。这种机制通常用于确保平台在起飞、着陆或低速机动时能够保持控制，而在高速巡航时则能够自动对准预期的航线。

maximum_attitude_rate <A>

参数列表	参数类型	参数含义
A	角速率值<angular-rate-value>	身体姿态变化的最大角速率

身体姿态变化的最大角速率。这主要与 weathercock_speed 配合使用 确定平台进入和退出 Weathercocking 模式的速度。

maximum_total_acceleration <A>

参数列表	参数类型	参数含义
A	加速度值<acceleration-value>	最大线性加速度值

最大线性加速度。在剧烈横向加速时设定上限。

minimum_upward_acceleration <A>

参数列表	参数类型	参数含义
A	加速度值<acceleration-value>	最小垂直加速度的限制

最小向上加速度值。当飞机正在上升，然后接到快速下降的指令时，这个值设定了最小垂直加速度的限制。请注意，许多设计上不允许带有摇摆旋翼头的旋翼机承受负重力。

body_rates_gain <A>

参数列表	参数类型	参数含义
A	浮点值<floating-point-value>	增益值

将角航向误差应用的增益，以获得消除航向误差的速率。主要用于在快速前进飞行时，消除航向误差，尤其是在超过风标速度的情况下。

当飞机从慢速飞行过渡到快速前进飞行时，飞机的气动特性会发生变化，这可能导致航向控制上的误差。为了纠正这种航向误差，飞行员或自动驾驶系统需要应用一个增益（可以理解为调整系数或力度），这个增益会作用于航向误差信号上，以产生一个控制力矩或控制指令，从而以适当的速率消除这个误差。这个增益的大小和方向取决于飞机的气动特性和飞行条件，以及所需的控制效果。在超过风标速度（即飞机的前进速度超过风速）时，这种控制尤为重要，因为这时飞机的稳定性和操控性可能会受到更大的挑战。

maximum_ground_speed <A>

参数列表	参数类型	参数含义
A	速度值<speed-value>	此运动模式下的最大行驶速度

maximum_rate_of_climb <A>

参数列表	参数类型	参数含义
A	爬升速率<speed-value>	此运动模式下允许的最大爬升速率

maximum_rate_of_descent <A>

参数列表	参数类型	参数含义
A	爬升率<speed-value>	此运动模式下允许的最大爬升率

此运动模式下允许的最大下降速率的大小。值必须大于零，但应用时会取反。

你可以为特定的运动模式设置一个正值的最大下降速率，但当应用这个参数时，它会被取反，因为下降速率是一个负值。这通常用于为特定模式设置一个现实的下降速率。

4. PID 控制器命令

proportional_gain <A>比例增益

参数列表	参数类型	参数含义
A	浮点值<floating-point-value>	增益值

用于当前误差的增益，旨在将误差消除至零。较大的值会更强力地尝试消除误差，但容易导致输出产生振荡。

derivative_gain <A>微分增益

参数列表	参数类型	参数含义
A	浮点值<floating-point-value>	增益值

用于当前误差变化率的增益，旨在防止误差超过期望目标值。较大的值会抵消输出的任何变化，导致漂移。

integral_gain <A>积分增益

参数列表	参数类型	参数含义
A	浮点值<floating-point-value>	增益值

用于抵消稳态偏差的增益，旨在将偏差减少到零。

input_threshold <A>

参数列表	参数类型	参数含义
A	浮点值<floating-point-value>	增益值

用于抵消稳态偏差的增益，旨在将偏差减少到零。

⋆WSF_SURFACE_MOVER 水面移动器

1. 概述

WSF_SURFACE_MOVER 为受约束沿水面移动的平台（例如 船舶）。此 mover 类似于 WSF_GROUND_MOVER;但是，默认情况下，Pitch 和 Roll 设置为零。

2. 指令

该移动器为 Route Mover，其指令与⋆WSF_GROUND_MOVER 地面车辆推进器指令相同。

WSF_TSPI_MOVER

1. 概述

WSF_TSPI_MOVER 是 AFSIM（Advanced Framework for Simulation, Integration, and Modeling）中的一个移动器类型，它根据从文本文件中读取的时间空间位置信息（Time Space Position Information，简称 TSPI）数据来更新平台的位置。这种移动器类型适用于需要根据实际飞行测试数据或历史轨迹数据来模拟平台运动的情况。

TSPI 数据通常包含一系列时间戳和相应的位置信息（例如，纬度、经度和高度），这些数据可以用来精确地重现平台的飞行路径或移动轨迹。

AP4.2. Follower 类型

这些移动器将“附加”到路由类型移动器，并用于让平台实例“跟随”其他平台。

WSF_HYBRID_MOVER

1. 概述

一个特殊的移动器，它整合了 WsfFollower 移动器和 WsfWaypointMover 移动器的功能。

WSF_OFFSET_MOVER

1. 概述

WSF_OFFSET_MOVER 实现了一个移动器，它强制平台保持在指定领路平台的预定偏移位置。平台可以被“刚性”或“系留”连接。

WSF_OFFSET_MOVER 是 AFSIM 仿真框架中的一个移动器类型，它用于模拟一个平台相对于另一个指定的领路平台（leader platform）保持特定的偏移位置。这种移动器可以用于多种场景，例如模拟战斗机在编队飞行中相对于长机的位置，或者在某些战术模拟中，一个平台需要保持在另一个平台的特定位置上。

在 AFSIM 中，使用 WSF_OFFSET_MOVER 可以定义平台的偏移移动行为，例如：

平台可以被“刚性”或“系留”地附着在领路平台上，这意味着它会保持与领路平台的相对位置不变。

可以指定偏移量，包括距离、高度和方向。

可以通过脚本或仿真输入文件来动态调整偏移量和领路平台。

这种移动器的实现通常涉及到计算领路平台的位置和速度，然后根据预设的偏移参数来更新跟随平台的位置。WSF_OFFSET_MOVER 允许在仿真中创建复杂的编队和队形，同时保持队形的稳定性和一致性。

在实际配置中，用户需要指定领路平台的名称或标识，并设置跟随平台相对于领路平台的偏移参数。这些设置可以通过 AFSIM 的输入文件或通过脚本在仿真运行时进行调整。

AP4.3. SixDof 类型

⋆WSF_SIX_DOF_MOVER 六自由度移动器基类

1. 概述

WSF_SIX_DOF_MOVER 是一个用于三维平移和三维旋转自由度的基于物理的时间步进移动器的基类。这些移动器能够产生比其他移动器更复杂的动力学效果。有两种六自由度（SixDOF）移动器类型可供选择：WSF_RIGID_BODY_SIX_DOF_MOVER 和 WSF_POINT_MASS_SIX_DOF_MOVER。

WSF_RIGID_BODY_SIX_DOF_MOVER 是 WSF_P6DOF_MOVER 的直接衍生类，并且几乎保留了它的所有功能。它是一个完整的六自由度移动器，包括三个旋转惯量的轴。而 WSF_POINT_MASS_SIX_DOF_MOVER 旨在提供一个更简化的模型，同时保留了子对象和序列器等有用特性。它在实用性上需要的调整较少，但仍然可以进行精细调整以符合特定的需求。

2.六自由度车俩类型指令

vehicle_type A>

参数列表	参数类型	参数含义
A	字符串<string>	对象类型名称

这将定义移动器使用的对象类型。请参阅 vehicle_type 和 vehicle_type。

autopilot_no_control A>自动驾驶无控制

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否启用自动驾驶无控制

这将命令自动驾驶仪将所有控件“归零”，这将使摇杆和方向舵居中，并将油门拉回零（怠速）。它类似于 enable_controls 但是他命令的是自动驾驶仪，而不是控件本身。

enable_controls A>启用控制

参数列表	参数类型	参数含义

A	布尔值<boolean-value>	

这将启用/禁用来自任何来源（自动驾驶仪、外部手动导航仪等）的控制输入。默认情况下，控件处于启用状态，因此此命令通常用于在启动时禁用控件。这通常用于允许武器在从运载飞机中释放时以弩炮方式投放，并且没有控制输入，然后在武器安全离开飞机后调用 true 脚本方法 WsfSixDOF_Mover.EnableControls 以建立控制输入。

enable_thrust_vectoring A>启用推力矢量控制

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否启用推力矢量

指示是否启用推力矢量。

engines_on A>发动机启动

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	发动机是否启动

指示在场景开始时发动机应该开启还是关闭。

follow_vertical_track 跟随垂直轨迹

通常，当改变高度时，自动驾驶仪会尽可能快地爬升/俯冲（在当前限制范围内 - 参见 WsfSixDOF_Mover.SetVerticalSpeedMax 和 WsfSixDOF_Mover.SetVerticalSpeedMin），但当设置“follow_vertical_track”时，自动驾驶仪将改用垂直速度，允许物体沿着航路点之间的直线垂直轨迹平滑/缓慢地改变高度。

尽管这是一个可选设置，并且默认不使用 follow_vertical_track（跟随垂直轨迹），但它经常会被使用，因为许多用户更喜欢由此产生的“渐进式”垂直飞行路径，而不是航点之间默认的“快速”高度变化。

ignore_all_crashes A>忽略所有坠毁

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	指示移动器是否忽略所有形式的坠毁。这通常用于测试

six_dof_alt A>移动器初始高度

参数列表	参数类型	参数含义
A	长度值<length-value>	此命令设置移动器的初始高度

six_dof_ned_heading A>移动器初始航向

参数列表	参数类型	参数含义
A	角度值<angle-value>	此命令设置移动器的初始航向。

six_dof_ned_pitch A>移动器初始俯仰角

参数列表	参数类型	参数含义
A	角度值<angle-value>	此命令设置移动器的初始俯仰角

six_dof_ned_roll A>移动器初始滚动角度

参数列表	参数类型	参数含义
A	角度值<angle-value>	此命令设置移动器的初始滚动角度

six_dof_position A> B>移动器的初始位置

参数列表	参数类型	参数含义
A	实数值<real-value>	纬度

B	实数值<real-value>	经度

此命令设置移动器的初始位置 （使用十进制纬度经度）。

six_dof_set_velocity_ned_fps A> B> C> NED 设置坐标系中移动器的初始速度

参数列表	参数类型	参数含义
A	实数值<real-value>	N
B	实数值<real-value>	E
C	实数值<real-value>	D

这个命令使用英尺/秒(ft/sec)为单位，在（NED）坐标系中设置移动器的初始速度。

在 AFSIM 或其他仿真平台中，设置移动器的初始速度是一个常见的需求，尤其是在进行飞行器或车辆动态仿真时。NED 坐标系是一种常用的导航坐标系，其中：

N 代表北向（North），表示沿着地球表面向北的方向。

E 代表东向（East），表示沿着地球表面向东的方向。

D 代表下向（Down），通常用于表示高度或深度的变化，向下为正。

produces_launch_smoke A>产生发射烟雾

参数列表	参数类型	参数含义
A	时间值<time-value>	持续时间值

这个命令设置了飞行器发射时产生发射烟雾/闪光效果（仅外观）的持续时间。如果不需要发射效果，则根本不应定义此命令。

throttle_afterburner <A>加力燃烧室油门

参数列表	参数类型	参数含义

A	布尔值<boolean-value>	将油门设置为 afterburner

这个命令在方案开始时将油门设置为 afterburner。

这个命令用于在仿真开始时将飞行器的油门设置为加力燃烧室状态，以便模拟飞行器在需要极快速度或爬升时使用的加力燃烧室功能。加力燃烧室可以提供额外的推力，通常在军事飞行器中用于空中战斗或逃避敌方。在 AFSIM 仿真环境中，这个命令可以帮助模拟飞行器在执行高速机动或需要额外推力时的行为。

throttle_full <A>油门全开

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	是否油门全开

这个命令在场景开始时将油门设置为军用功率。

这个命令用于在仿真开始时将飞行器的油门设置为最大推力状态，通常用于模拟起飞或需要最大推力的飞行操作。通过设置为“油门全开”，飞行器可以在仿真中以最高的动力输出进行飞行，适合执行高性能的飞行任务。

throttle_idle A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	在场景开始时将油门设置为怠速

这个命令在场景开始时将油门设置为怠速。

在 AFSIM 或类似的飞行仿真平台中，"idle" 指的是发动机的非工作状态，即油门被设置到最低，发动机仅维持最基本的运转水平，不提供额外的推力。这个设置通常用于模拟飞行器在地面等待起飞指令或在滑行时的状态。将油门设置为怠速可以确保飞行器在仿真开始时不会立即开始移动，而是处于一种准备就绪的状态，等待进一步的指令或操作。这对于进行起飞前的检查、系统测试或模拟飞行器在机场地面的运行情况非常有用。

wash_in_conditions A>

参数列表	参数类型	参数含义

A	布尔值<boolean-value>	

这个命令或设置可能用于指定在仿真中模拟飞行器或平台在遭遇水面或潮湿环境时的特定条件，例如在进行海上作战、水上飞机操作或水下潜航器的仿真时。这种设置有助于评估飞行器在这些特定环境下的性能和行为。

3.航路相关指令

WSF_SIX_DOF_MOVER 支持以下 route 命令

注意：WSF_SIX_DOF_MOVER 并不支持所有 route 命令。

route … end_route 块

position

参数列表	参数类型	参数含义
A	纬度值<latitude-value>	航点纬度
B	经度值<longitude-value>	航点经度

altitude A>

参数列表	参数类型	参数含义
A	长度值<length-value>	航点高度

speed A>

参数列表	参数类型	参数含义
A	速度值<speed-value>	航点处的速度

label <A>

参数列表	参数类型	参数含义
A	string	标签名称

将标签与紧接着的航点定义关联。这可以用作 goto 命令的目标。

goto <A>

参数列表	参数类型	参数含义
A	string <label>	标签名称

前往当前路线中带有指定标签的航路点。

bank_angle_limit A>

参数列表	参数类型	参数含义
A	角度值 <angle-value>	倾斜角

指定在从此航点开始的路线段上进行航向变化时转弯所使用最大倾斜角。这有效地将径向加速度设置为$g * tan(bank_angle_limit)$.

在 AFSIM 或类似的飞行仿真平台中，这个设置用于限制飞行器在沿着预定路径进行航向变化时的最大倾斜角度。倾斜角是飞行器在转弯时与水平面的夹角，这个角度的大小直接影响飞行器的转弯半径和所需的向心加速度。

最大倾斜角：设置最大倾斜角可以防止飞行器在转弯时超过其物理或性能限制，确保飞行器的稳定性和安全性。

径向加速度：在转弯时，飞行器需要向圆心施加一定的加速度，即向心加速度。这个加速度与飞行器的倾斜角有关，可以通过公式 g * tan（倾斜角限制）来计算，其中 g 是重力加速度。

航向变化：在飞行路径规划中，航向变化是指飞行器从当前航向转向新航向的过程。在 AFSIM 中，通过设置最大倾斜角，可以模拟飞行器在执行航向变化时的真实物理行为。

radial_acceleration A>

参数列表	参数类型	参数含义
A	加速度值 <acceleration-value>	径向加速度值

指定在从此航点开始的路线段上进行航向变化时转弯所使用的径向加速度。
在 AFSIM 或类似的飞行仿真系统中，这个设置用于定义飞行器在执行航向改变时，转弯过程中所施加的径向加速度。径向加速度是指飞行器在转弯过程中，沿着转弯半径向圆心方向的加速度，它决定了转弯的紧凑程度和所需的向心力。

转弯半径：径向加速度与转弯半径成反比。加速度越大，所需的转弯半径越小，飞行器可以更紧密地转弯。

航向变化：在飞行器的飞行路径中，航向变化是指飞行器从当前航向转向新航向的过程。通过设置适当的径向加速度，可以模拟飞行器在执行航向变化时的动态响应。

仿真精度：在仿真中准确设置径向加速度对于确保飞行器的转弯行为符合物理规律和实际操作经验至关重要。

turn_g_limit A>

参数列表	参数类型	参数含义
A	加速度值 <acceleration-value>	最大转弯过载

指定在从此航点开始的路线段上进行航向变化时转弯所使用的最大转弯过载。这有效地将径向加速度设置为 $sqrt(turn\_g\_limit^2 - g^2)$。

在 AFSIM 仿真平台中，这个设置用于定义飞行器在执行转弯动作时所允许的最大过载（G 力）。过载是指飞行器在机动飞行中所承受的加速度与重力加速度的比值。这个设置确保飞行器在转弯时不会因为过载过大而超出结构或乘员的承受能力。

switch_on_passing

switch_on_approach

定义移动器何时应该声明它已经到达这个航点，并应该开始向下一个航点移动的条件。switch_on_passing 有时被称为“飞越点”，当平台飞越或飞过航点时触发切换。switch_on_approach 有时被称为“提前转弯”，在平台到达航点之前触发切换。

注意：WSF_SIX_DOF_MOVER 默认为 switch_on_approach，这与大多数其他移动器不同。

use_route <A>

参数列表	参数类型	参数含义
A	字符串	航路名称

为推进器指定航路。

4.WSF_SIX_DOF_MOVER 路线示例

示例路由定义如下所示。最简单的路线包括使用 position、altitude 和 speed 命令的多条线路，如下所示：

```
route
   position 21.325n 158.000w   altitude  9000.0 ft   speed 600.0 kts
   position 21.325n 157.941w   altitude  9000.0 ft   speed 600.0 kts
   position 21.250n 157.800w   altitude  9000.0 ft   speed 600.0 kts
   position 21.260n 157.700w   altitude  9000.0 ft   speed 600.0 kts
   position 21.400n 157.700w   altitude  9000.0 ft   speed 600.0 kts
   position 21.700n 157.900w   altitude 13000.0 ft   speed 600.0 kts
   position 21.900n 158.300w   altitude  9000.0 ft   speed 600.0 kts
   position 21.600n 158.200w   altitude  9000.0 ft   speed 600.0 kts
   position 21.550n 158.120w   altitude  9000.0 ft   speed 600.0 kts
   position 21.325n 157.941w   altitude  9000.0 ft   speed 600.0 kts
   position 21.325n 157.900w   altitude  9000.0 ft   speed 600.0 kts
end_route
```

⋆WSF_POINT_MASS_SIX_DOF_MOVER 质点六自由度移动器

1. 概述

WSF_POINT_MASS_SIX_DOF_MOVER 是 AFSIM 仿真框架中的一个移动器类型，它提供了一个简化的六自由度（6DOF）模型，用于模拟平台的运动。这个移动器适用于需要考虑基本物理运动，但不需要复杂刚体动力学的场景。

WSF_POINT_MASS_SIX_DOF_MOVER 是一种适用于简化动力学模拟的移动器，它通过预先定义的表格数据来模拟旋转和推力矢量的影响。这种移动器适用于那些不需要详细模拟质量分布和惯性特性的场景，例如在进行初步设计评估或概念验证时。尽管它在模拟复杂动力学方面有所限制，但它提供了一种计算效率更高、更容易实现的方法。

2. PM6 车辆类型指令

WSF_POINT_MASS_SIX_DOF_MOVER 最重要的命令是 vehicle_type，它定义对象的性能特征。vehicle_type 定义为 six_dof_object_types 块中的 point_mass_vehicle_type。必须先定义 vehicle_type，然后才能在 mover WSF_POINT_MASS_SIX_DOF_MOVER 块中引用它。

⋆WSF_RIGID_BODY_SIX_DOF_MOVER 刚体六自由度移动器

1. 概述

WSF_RIGID_BODY_SIX_DOF_MOVER 是 WSF_SIX_DOF_MOVER 的一种特定类型，它使用一阶和二阶力和力矩系数来实现旋转和平移。它是 WSF_P6DOF_MOVER 的直接衍生产品，几乎采用了它的所有功能。

构建一个成功的 WSF_RIGID_BODY_SIX_DOF_MOVER 所需的细节和数据量是非平凡的，但结果是能够自然捕捉到其他移动器（包括 WSF_POINT_MASS_SIX_DOF_MOVER）无法捕捉的现象和动力学。

WSF_RIGID_BODY_SIX_DOF_MOVER 仅使用惯性张量的主对角元素。这种简化有助于加快执行速度并减少一些复杂性，但可能会削弱交叉耦合效应的阻尼。净力和力矩是从空气动力学、推进力和重力源聚合而来，并直接用于运动方程中。

所有自由度的空气动力学系数都是可用的，包括导数系数。并非所有系数都是必需的，但每个使用的表格都可以增加模型的保真度。阻尼，特别是在气动阻尼方面，对于模拟真实

世界的飞行器行为至关重要，因为它影响着飞行器对控制输入的响应以及在受到扰动后的稳定和恢复能力。

这种移动器适用于需要高保真度动力学模拟的场景，如高级飞行器设计分析、复杂的交战模拟或科学研究。由于其复杂性，使用 WSF_RIGID_BODY_SIX_DOF_MOVER 可能需要专业的知识和经验来正确配置和使用。

2. RB6 车辆类型指令

对于 WSF_RIGID_BODY_SIX_DOF_MOVER 来说，最重要的命令是 vehicle_type，它定义了对象的性能特征。刚体 vehicle_type 被定义为 six_dof_object_types 块中的一个命令：rigid_body_vehicle_type。在 mover WSF_RIGID_BODY_SIX_DOF_MOVER 块中引用之前，必须先定义 six_dof_vehicle_type。

vehicle_type A>

参数列表	参数类型	参数含义
A	string	使用的对象类型名称

这将定义移动器使用的对象类型。

最简单的 WSF_RIGID_BODY_SIX_DOF_MOVER 定义是这样的：

```
mover WSF_RIGID_BODY_SIX_DOF_MOVER
   vehicle_type  F-15C
end_mover
```

landing_gear_down <A>

参数列表	参数类型	参数含义
A	布尔值	起落架是否放下

指示在场景开始时起落架是否应该放下。

在 AFSIM 或类似的飞行仿真平台中，这个设置用于控制飞行器在仿真开始时的起落架状态。起落架是飞行器用于在地面上移动和停放的装置，通常在起飞前收起以减少空气阻力，在着陆前放下以提供支撑和缓冲。

起落架放下：如果设置为“是”或“真”，则在仿真开始时，飞行器的起落架将处于放下状态，模拟飞行器准备着陆或已经着陆的情况。

起落架收起：如果设置为“否”或“假”，则在仿真开始时，飞行器的起落架将处于收起状态，模拟飞行器在空中飞行或准备起飞的情况。

nws_enabled <A>

参数列表	参数类型	参数含义
A	布尔值	是否启用前轮转向

指示是否应该启用前轮转向。前轮转向通常在滑行时使用，但在起飞滑跑前应该取消。

在 AFSIM 或类似的飞行仿真平台中，前轮转向（nose-wheel steering）是一个重要的功能，它允许飞行器在地面滑行时进行转向操作。这个设置通常用于模拟飞行器在机场地面滑行时的操控性，特别是在转弯或导航至跑道时。

启用前轮转向：如果设置为“是”或“真”，则在仿真开始时，飞行器的前轮可以进行转向，以便在地面上进行精确的操控。

禁用前轮转向：如果设置为“否”或“假”，则在仿真开始时，飞行器的前轮将被锁定，无法进行转向，这通常用于模拟起飞滑跑阶段，因为在此阶段，为了安全和性能考虑，通常需要保持前轮直行

parking_brake_on A>

参数列表	参数类型	参数含义
A	布尔值	是否启用驻车制动器

此命令设置在场景开始时应打开还是关闭驻车制动器。

taxi_mode_enabled A>

参数列表	参数类型	参数含义
A	布尔值	在场景开始时自动驾驶仪是否应该处于滑行模式

这个命令设置在场景开始时自动驾驶仪是否应该处于滑行模式。这只在平台在地面上时使用。

AP4.4. P-6DOF 类型

WSF_P6DOF_MOVER 伪六自由度推进器（已弃用）

1. 概述

WSF_P6DOF_MOVER 是一个高保真的伪六自由度（6DOF，six degrees of freedom）移动器。P6DOF 提供了六个自由度（允许姿态/角运动学以及平移运动学），提供了比基于 3DOF 模型更高的保真度模拟能力，它包含了一个基于物理的模型（使用力和力矩积累），包括马赫数和高度效应。P6DOF 从一开始就设计为与 AFSIM 集成，简化了其在框架中的使用。

WSF_P6DOF_MOVER 是一个伪 6DOF 模型，意味着与其说是“硬核”6DOF 模型，它在姿态计算中包含了一些简化。这些包括一些角加速度和惯性张量简化，但模型在其运动计算中保留了完整的六个自由度——因此，P6DOF 是真正的 6DOF（与其他被称为伪 6DOF 但实际上只是 5DOF 模型的模型不同）。P6DOF 中使用的姿态简化实际上对姿态运动学的影响很小。然而，这些“伪”设计特征的结果是，P6DOF 所需的数据更少，需要的计算更少，比完整的 6DOF 模拟更容易使用，但它仍然提供了包括偏航、俯仰和滚转以及迎角（alpha）和侧滑角（beta）效应的真实建模。

WSF_P6DOF_MOVER 旨在支持广泛的移动器车辆，包括固定翼飞机、旋翼机、导弹（包括空对空（AAM）、地对空（SAM）、空对地（AGM）和反弹道导弹（ABM））、弹道导弹（包括短程弹道导弹（SRBM）、中程弹道导弹（MRBM）、中远程弹道导弹（IRBM）和洲际弹道导弹（ICBM））、太空发射载具（包括典型的多级线性级联助推器，以及并联级联配置，和有翼再入飞行器）、以及航天器/卫星。最初的支持重点是固定翼飞机，但将来会支持/扩展其他类型。

WSF_P6DOF_MOVER 使用“框架不可知”的 P6DOF 核心软件，允许在需要时将其用于 AFSIM 之外的其他仿真框架。

使用 WSF_P6DOF_MOVER 而不是 WSF_AIR_MOVER、WSF_GUIDED_MOVER、WSF_HYBRID_MOVER、WSF_TBM_MOVER 或 WSF_KINEMATIC_MOVER 的优势在于，平台的姿态被更准确地建模，具有通过基于力和力矩的真实物理/空气动力学模型计算的真实偏航、俯仰和滚转以及迎角（alpha）。使用 WSF_P6DOF_MOVER 的劣势是，其增加的真实性需要更多的计算和计算机处理时间。

AP4.5. ARGO8 类型

WSF_ARGO8_MOVER

1. 概述

WSF_ARGO8_MOVER 是一个特定的移动器（mover）类型，它为 TMAP ARGO 导弹模型对象提供了移动能力。这意味着如果你在 AFSIM 中模拟导弹或其他需要精确移动控制的物体，你可以使用 WSF_ARGO8_MOVER 来定义其在仿真环境中的运动特性。

3. 应用环境

- 需要更加真实的导弹飞行动态
- 根据需要调整移动器参数，以模拟不同的导弹行为或策略

AP4.6. 卫星/轨道类型

WSF_NORAD_SPACE_MOVER

1. 概述

WSF_NORAD_SPACE_MOVER 实现了一个用于模拟地球轨道上平台移动的移动器。它适用于那些有两行元素（TLE）数据集的卫星模型，包括大多数运行中的地球轨道卫星、非运行中的卫星以及被积极追踪的轨道碎片。

它用于模拟地球轨道上的平台，如卫星。这种移动器对于模拟概念性卫星和卫星星座非常有用，同时也提供了对现有卫星的理想化表示。它可以通过多种方式进行初始化，包括提供初始位置、经典的轨道元素、位置和速度状态信息，或者两行元素（TLE）集。此外，它还可以通过脚本使用 WsfSpaceMover 方法进行初始化。

WSF_SPACE_MOVER 卫星

1. 概述

WSF_SPACE_MOVER 是一个用于模拟地球轨道上平台（例如卫星）的移动器。它对于构建概念性卫星模型和卫星群，以及为现有卫星提供理想化的表现非常有用。

默认情况下，这个移动器模拟卫星围绕一个质点的轨道运动。如果启用了“oblate_earth (地球扁率)”选项，WSF_SPACE_MOVER 还会包括由于地球的扁率而产生的平均一阶引力摄动效应。

WSF_SPACE_MOVER 可以通过多种方式进行初始化。一种方法是提供初始位置。或者，也可以提供经典的轨道元素、位置和速度状态信息，或者两行轨道元素。它还可以通过脚本进行初始化，使用 WsfSpaceMover 提供的方法。

WSF_SPACE_MOVER 能够执行各种机动操作。这些机动可以通过任务序列输入来指定，或者通过 WsfSpaceMover 脚本对象和移动器的脚本接口进行脚本化编程。可配置的机动模型决定了在执行机动的任务序列中如何分配 delta-V（速度变化量）。

移动器的朝向是通过可配置的姿态控制器模型来指定和控制的。提供了多种标准朝向选项，同时也可以动态地通过脚本进行朝向的调整。

WSF_INTEGRATING_SPACE_MOVER

1. 概述

WSF_INTEGRATING_SPACE_MOVER 是 AFSIM 仿真框架中用于模拟太空领域内平台移动的移动器。与 WSF_SPACE_MOVER 或 WSF_NORAD_SPACE_MOVER 不同，这两种移动器都使用分析模型来提供平台的未来状态，而 WSF_INTEGRATING_SPACE_MOVER 则使用数值积分方法。通过使用用户可指定的动力学模型，这种移动器不局限于那两种移动器所关注的案例，即地球在动力学中起主导作用的传播。此外，WSF_INTEGRATING_SPACE_MOVER 支持双曲线传播，因此也可以用来表示无约束轨道。

AP4.7. Brawler 类型

WSF_BRAWLER_MOVER

1. 概述

它通常用于模拟近距空战（Within-Visual-Range, WVR）的战术飞行行为。

2. 应用环境

近距离空战：当平台进入目标的视觉范围内时，使用 WsfBrawlerProcessor 提供的移动器方法来执行空战战术，如机动、瞄准和发射武器。

常规导航：当平台在视距外（Outside of Visual Range, OVR）时，使用 WsfPlatform 提供的位置和导航方法来执行常规飞行任务，如按照预定航线飞行、前往特定位置或转向特定的航向。

AP4.8. 军事类型

WSF_FIRES_MOVER

1. 概述

WSF_FIRES_MOVER 是一种特定的移动器（mover）类型，用于模拟火炮或类似武器系统的射击行为。这种移动器通常用于地面车辆或其他平台，它们装备有火炮或自动武器系统，能够在静止或移动中对目标进行射击。

WSF_FORMATION_FLYER

1. 概述

WSF_FORMATION_FLYER 是一种移动器（mover）类型，专门用于模拟执行编队飞行的飞行器。这种移动器适用于需要模拟多个飞机或无人机以特定队形飞行的场景，例如军事空中作战、空中表演或商业航空运输。

2. 典型用途和特点

编队飞行：模拟飞机按照预定的编队队形飞行，保持特定的相对位置和距离。

队形调整：在飞行过程中动态调整编队结构，以应对战术变化或适应不同的任务需求。

领导跟随：在编队中指定领飞机，其他飞机根据领飞机的位置和飞行路径自动调整自己的位置。

协调机动：执行协调的机动动作，如同步转弯、爬升、俯冲等，以保持编队的整体性和协同性。

间距和高度控制：精确控制编队中飞机之间的间距和高度差，以确保安全和有效的飞行。

战术模拟：在军事仿真中，模拟执行特定战术任务的编队飞行，如空中加油、护航或对地攻击。

视觉模拟：为了视觉效果或演示目的，模拟飞机以特定的队形和路径飞行。

WSF_GUIDED_MOVER

1. 概述

WSF_GUIDED_MOVER 是一种用于模拟制导武器或导弹的移动器（mover）类型。这种移动器专门设计来模拟那些能够通过制导系统调整飞行路径以击中目标的武器。

WSF_GUIDED_MOVER 在 AFSIM 仿真框架中实现了一种移动器，它能够以适度的保真度代表制导滑翔炸弹、单级或多级制导导弹或火箭。这种移动器设计用来模拟具有制导能力的武器系统，它们可以在发射后通过制导系统调整飞行路径以精确击中目标。

2. 典型用途和特点

制导飞行：模拟导弹或其他制导武器在发射后根据制导系统（如惯性导航、GPS、雷达制导、红外制导等）调整飞行路径。

目标追踪：模拟制导武器在飞行过程中追踪移动或静止目标的能力。

机动规避：模拟制导武器在面对敌方拦截或规避动作时的机动能力。

末端制导：模拟在接近目标时切换到更精确的制导模式，以提高命中率。

多模式制导：支持多种制导模式的武器，如初始阶段使用惯性导航，末段切换到红外或雷达制导。

弹道模拟：模拟制导武器的完整弹道，包括上升、巡航和俯冲阶段。

武器效应：模拟制导武器击中目标后的效果，如爆炸威力、穿透能力等。

对抗措施：模拟制导武器对抗敌方干扰和防御措施的能力，如电子对抗和诱饵。

WSF_PARABOLIC_MOVER

1. 概述

WSF_PARABOLIC_MOVER 是 AFSIM 仿真框架中的一个移动器类型，它用于定义平台的抛物线或弹道轨迹。这种移动器适用于需要模拟抛物线运动的平台，例如炮弹、炸弹或其他类型的投射物。它提供了一种简化的方式来模拟这些物体在重力作用下的运动，而不涉及到复杂的空气动力学或推进力模型。

WSF_STRAIGHT_LINE_MOVER 直线移动器

1. 概述

WSF_STRAIGHT_LINE_MOVER 是一种用于模拟沿直线轨迹移动的平台的移动器。它通常用于表示在仿真中的武器或飞行器的运动，这些武器或飞行器的运动轨迹可以被简化为直线。这种移动器不涉及复杂的空气动力学或质量特性，而是根据预设的最大速度、加速度等参数来模拟平台的连续运动。

WSF_STRAIGHT_LINE_MOVER 是一种用于 WSF_EXPLICIT_WEAPON 的移动器，它模拟武器从发射点到拦截目标轨迹的（或多或少）直线飞行。使用这种移动器，用户无需指定武器的质量特性、推进系统或空气动力学表格，但相应地会牺牲一定的仿真精度。目标轨迹必须由某种 WsfSensor 或 WsfProcessor 类型提供，这些类型通过 WsfPlatform 的 WsfTrackManager 填充当前目标。

WSF_STRAIGHT_LINE_MOVER 将实现以下功能：

以恒定的平均速度飞行，或者通过速度表（与飞行时间线性插值）来模拟武器的飞行。

可选地，用户可以指定一个最大的横向 g 限制，这将限制任何初始或最终机动加速度的大小。

这种移动器适用于以下情况：

当仿真的重点是武器的发射和命中，而不是其飞行过程中的详细动力学特性时。

当需要快速模拟武器的直线飞行轨迹，而不需要复杂的物理模型时。

WSF_STRAIGHT_LINE_MOVER 的使用简化了武器模型的创建过程，因为它不需要复杂的推进或空气动力学模型。然而，这种简化也意味着在模拟武器的飞行行为时，可能无法

准确反映真实世界的物理现象，如空气阻力、推力变化或复杂的机动。因此，在选择使用 WSF_STRAIGHT_LINE_MOVER 时，需要根据仿真的目的和所需的精度来权衡。

⋆WSF_SUBSURFACE_MOVER 水下航行器移动器

1. 概述

WSF_SUBSURFACE_MOVER 是一种用于模拟水下航行器的移动器，属于 Route Mover 的一种。它允许定义一系列的航点，以便在仿真中模拟水下航行器从一点移动到另一点的路径。这种移动器特别适用于模拟潜艇或其他类型的水下航行器。

2. 指令

该移动器为 Route Mover，其指令与⋆WSF_GROUND_MOVER 地面车辆推进器指令相同。

WSF_TBM_MOVER

1. 概述

WSF_TBM_MOVER 是 AFSIM（Advanced Framework for Simulation, Integration, and Modeling）中的一个预定义移动器类型，专门用于模拟战术弹道导弹的运动。这种移动器能够以简化的格式表示单级或多级弹道导弹的飞行特性。

WSF_TOWED_MOVER

1. 概述

WSF_TOWED_MOVER 是 AFSIM（Advanced Framework for Simulation, Integration, and Modeling）中的一个移动器类型，它用于模拟一个被牵引平台（towed platform）跟随牵引它的领先平台（lead platform）的移动。这种移动器类型特别适用于模拟拖曳式目标、拖曳式声纳或其他需要跟随另一平台移动的设备。

WSF_UNGUIDED_MOVER

1. 概述

WSF_UNGUIDED_MOVER 是 AFSIM（Advanced Framework for Simulation, Integration, and Modeling）中的一个移动器类型，用于模拟无制导的“哑”炸弹或单级或多级的无制导火箭。这种移动器是从 WSF_GUIDED_MOVER 派生出来的，因此它接受相同的命令集，但由于它模拟的是无制导武器，其“制导”能力默认是禁用的，且无法重新启用。

附录五（处理器）

WSF_AIR_TARGET_FUSE

1、概述

WSF_AIR_TARGET_FUSE 是一种特殊的 WSF_WEAPON_FUSE，它提供了适用于打击空中目标的武器的默认设置。以下为附加的默认值：

use_current_target（使用当前目标）

update_interval 0.5 sec

gross_proximity_range 1500 m

备注：此处理器为默认设置，其参数无需设置，参数详见：WSF_WEAPON_FUSE

WSF_ASSET_MANAGER

来源于：WSF_SCRIPT_PROCESSOR

脚本类：WsfAssetManager

1、概述

WSF_ASSET_MANAGER 是一个脚本基类，供基于 HELIOS 的资产管理模型继承。它不打算作为一个独立的处理器使用，相反，资产管理模型利用它来提供所有资产管理处理器中常见的脚本功能。

2、资源管理命令

filter_dead_tracks A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	过滤掉已失效的轨迹

如果设置为真（true），则会进行真实性呼叫，以确认目标是否存活，然后才考虑将其包含在威胁数组中。

默认值：false

max_track_grouping_distance A>

参数列表	参数类型	参数含义
A	长度值<length-value>	最大轨迹分组距离

设置用于确定轨迹强度的分组距离。请注意，轨迹在分组时不会被合并，仅仅是在强度计数上进行累加。

默认值：50 m

max_assignments A>

参数列表	参数类型	参数含义
A	integer	最大任务分配数

该命令设置了资产的最大活跃任务分配数。

注意：从属任务分配计入此总数。

默认值：0

assignment_delay A>

参数列表	参数类型	参数含义
A	时间值<time-value>	任务分配延迟

此命令设置 Asset Manager 的分配延迟。

注意：任务分配延迟发生在以下两个时刻：对于具有提交权限的处理器，在排队发送传出消息之前；对于没有提交权限的处理器，在处理接收到的消息之前。

默认值：0 秒

decision_update_delay <A>

参数列表	参数类型	参数含义
A	时间值<time-value>	决策更新延迟

该命令设置了任务分配的路由。

注意：这是一个定向消息，因此通常将被设置为“动态”。

默认值：0 秒

log_status A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	记录状态

这条命令用于开启或关闭资产管理器的状态日志记录。

默认值：true

3、状态设置命令

定义 HELIOS 资产管理器状态设置的区块。

subord_yellow_timeout A>

参数列表	参数类型	参数含义
A	时间值<time-value>	次级黄色状态超时

指定在将从属状态设置为黄色（表示不可用）之前，等待状态报告的时长。

默认值：15 秒

subord_red_timeout A>

参数列表	参数类型	参数含义
A	时间值<time-value>	次级红色状态超时

指定在将从属状态设置为红色（表示不可用）之前，等待状态报告的时间长度。

默认值：30 秒

report_position_every A>

参数列表	参数类型	参数含义
A	长度值<length-value>	每次位置报告间隔

当平台处于移动状态时，指定应多久发送一次位置信息。

默认值：100 m

or_every A>

参数列表	参数类型	参数含义
A	时间值<time-value>	

指定位置信息的传输频率。

默认值：300 s

report_status_every A>

参数列表	参数类型	参数含义
A	时间值<time-value>	传输状态消息的频率

指定传输状态消息的频率。

默认值：10 s

aggregate_unit_status <A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	聚合单元状态

在确定是否满足 GREEN 状态要求时，包括处于 GREEN 状态的下属单位。

默认值：false

stationary_opns_only <A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	仅在静止状态下操作

要求资产在静止状态下才能达到 GREEN 状态。

默认值：false

weapon_required A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	需要武器

要求配备弹药的武器才能达到 GREEN 状态。

默认值：false

require_all_weapons A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	需要武器

要求平台上所有武器都必须装填弹药才能达到 GREEN 状态。

默认值：false

ew_required A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	需要电子战设备

需要电子战雷达才能达到 GREEN 状态。

默认值：false

tar_required A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	需要目标获取雷达

需要目标获取雷达才能达到 GREEN 状态。

默认值：false

ttr_required A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	需要目标跟踪雷达

需要目标跟踪雷达才能达到 GREEN 状态。

默认值：false

4、资源管理器示例

```
status_settings
   subord_yellow_timeout 120 secs
   subord_red_timeout    180 secs
   report_position_every 100 meters
   or_every              300 secs
   report_status_every   10 secs
   aggregate_unit_status true
   weapon_required       true
   require_all_weapons   false
   ew_required           false
   tar_required          true
   ttr_required          true
   stationary_opns_only  false
end_status_settings

max_assignments        10
assignment_delay       0 secs
decision_update_delay  0 secs
log_status             true
```

WSF_BATTLE_MANAGER

来源于：WSF_SCRIPT_PROCESSOR

脚本类：WsfBattleManager

1、概述

WSF_BATTLE_MANAGER 是一个脚本基类，用于所有基于 HELIOS 的战斗管理器继承。它本身不作为处理器使用，而是其他战斗管理器利用它来实现所有战斗管理器共有的通用脚本功能。

2、战斗管理器指令

commit_authority <A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	提交权限

这条指令用于启用或禁用提交权限。如果关联的资产管理器未设置为过滤掉死亡轨迹（即无效轨迹），则可能会将过时或无效的轨迹分配给武器。此外，轨迹处理器上异常的高或低的清除间隔可能会导致异常的分配行为。

默认值：false

project_tracks_by_delays A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	按延迟投影轨迹

这条命令用于开启或关闭在创建分配时按分配延迟进行轨迹投影的功能。

默认值：true

project_targets_forward A>

参数列表	参数类型	参数含义
A	时间值<time-value>	向前投影目标时间

这条命令允许为武器评估目的配置目标的向前投影时间。请注意，HELIOS 会根据模型默认设置这些值，但它们可以通过此脚本设置进行覆盖。

默认值：60 s

by A>

参数列表	参数类型	参数含义
A	时间值<time-value>	

这条指令允许用户为武器评估配置目标的前向投影时间间隔。

默认值：10 s

3、参与 IFF 权限指令

"IFF" 指的是 "Identification Friend or Foe"（敌我识别），是一种用于军事目的的识别系统。

任何 IFF 类型，如果值为真（true），将被战斗管理器允许交战。值为假（false）的 IFF 类型将不会获得战斗管理器的交战许可。

unknowns A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	

默认值：false

neutrals A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	

默认值：false

friendlies A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	

默认值：false

hostiles A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	

默认值：false

4、战斗管理器示例

```
commit_authority         true
project_tracks_by_delays false

 engage_iff_permissions
    unknowns   false
    neutrals   false
    friendlies false
    hostiles   true
 end_engage_iff_permissions
```

WSF_BRAWLER_PROCESSOR

1、概述

WSF_BRAWLER_PROCESSOR “处理器类型”用于 BRAWLER 在目视范围内的空对空战斗机动。WSF_BRAWLER_PROCESSOR 使用 WSF_BRAWLER_MOVER、WSF_PERCEPTION_PROCESSOR 和 WSF_QUANTUM_TASKER_PROCESSOR，并通过脚本行为利用 WsfBrawlerProcessor 评估和控制功能。WSF_BRAWLER_PROCESSOR 依赖于接收“WEAPON”任务来知道其 1v1 空中格斗机动的目标是谁。

2、指令

mind_file A>

参数列表	参数类型	参数含义
A	文件名 file-name	

为 BRAWLER 心智模型及其意识事件更新指定心智文件。BRAWLER 非保密版本中的心智文件完全兼容。

默认值：无文件 - 使用默认值。

mover WSF_BRAWLER_MOVER mover-commands ...end_mover

定义 BRAWLER 平台要使用的 BRAWLER 移动器。

debug

如果指定，每个备选评估的调试信息将被输出到标准输出设备。

draw_alternatives

如果指定，每个向前投影的备选方案将以彩色编码的点在 DIS 回放文件中呈现。绿色表示该备选方案评估值接近迄今为止看到的最大值，红色表示相反，黄色/橙色表示介于两者之间。

draw_nominal_states

如果指定，目标的前向投影标准状态将以彩色编码的点呈现在 DIS 回放文件中。蓝色点代表我方，紫色点代表目标。BRAWLER 的标准状态投影是一种基于物理的前向投影，其中加速度被忽略，速度在投影过程中保持恒定。标准状态通常在评估函数中使用。

time_allowed_per_sector_search A>

参数列表	参数类型	参数含义
A	时间值<time-value>	每个扇区搜索允许的时间

意指代表用于搜索天空中一个扇区所允许的时间。这会影响意识事件的时间触发。

默认值：10 s

consciousness_event_update_time A>

参数列表	参数类型	参数含义
A	时间值<time-value>	意识事件更新时间

覆盖 BRAWLER 意识事件的定时。如果指定了，所有意识事件将在这个间隔发生。否则，将定期进行动态时间计算以确定适当的意识事件更新时间。

默认值：未指定

WSF_COHERENT_SENSOR_PROCESSOR

1、概述

WSF_COHERENT_SENSOR_PROCESSOR 提供了相干传感器功能，包括将多个传感器 WsfEM_Interaction 结果（即非本地结果）处理成单个交互结果（本地结果）以进行检测和跟踪报告的能力。该处理器将基于 result_processing_type 从非本地结果计算本地结果，并使用 fusion 从非本地结果计算本地结果位置（即测量）。

注意：如果指定了 update_interval，则传感器或目标结果将被排队，直到下一个更新间隔并批量处理。如果没有指定 update_interval，则传感器或目标结果将在每个传感器更新每个目标时立即处理。

2、指令

sensors … end_sensors

```
sensors
   sensor <sensor-name>
   platform_sensor <platform-name> <sensor-name>
end_sensors
```

sensor <*sensor-name*>

指定与处理器位于同一平台上的传感器名称 <sensor-name>。

platform_sensor <platform-name> <sensor-name>

指定外部平台上的传感器名称 <sensor-name>，该平台的名称为 <platform-name>。

detection_threshold <A>

参数列表	参数类型	参数含义
A	dbratio-value	

定义接收器检测阈值的另一种方法。为了提高输入文件的可读性，可以在这里输入该值。

默认值：3.0 db

use_target_result A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	使用目标结果

指定如果为“真”(true)，则应使用每个传感器的目标结果。如果为“假”(false)，则使用

每个传感器的结果，这是根据传感器能力，每种模式和波束的综合结果。

默认值：false

coast_time

参数列表	参数类型	参数含义
A	时间值<time-value>	"惯性滑行时间

指定在放弃轨迹之前，更新之间可能经过的最大时间量。

默认值：未指定默认值

result_processing_type [SNR_BASED | RSS_BASED]

指定用于计算从非本地传感器或目标结果到用于轨迹形成中的本地结果的信噪比（SNR）的结果处理类型。

请注意，无论哪种类型，融合处理都将进行。

- **SNR_BASED**
- 使用基于信噪比（SNR）的最佳非本地结果来形成本地结果，最佳非本地结果的 SNR 及其他数据将替代本地结果。
- **RSS_BASED**

使用每个有效信号和噪声功率测量的非本地结果的信号功率的平方和的平方根（RSS）与噪声功率的均方根（RMS）相结合，来计算本地结果的信号、噪声和信噪比（SNR）。

$$SNR_{\mathrm{RSS}} = \frac{\sqrt{\sum_{i=1}^{N} P_{signal\_i}^2}}{\sqrt{\sum_{i=1}^{N} P_{noise\_i}^2 / N}}$$

默认值：SNR_BASED

fusion_method [replacement | weighted_average | MTT(Multi-Target Tracker) mtt]

指定结果处理使用的融合算法。

融合算法将来自两个或多个来源的有关单个实体的信息组合成一个连贯的信息集例如，测量或跟踪。

- **replacement**

根据一组标准算法对相关测量值进行融合。本地测量位置将替换为非本地测量位置。

- **weighted_average**

相关测量值根据一组标准算法进行融合。使用局部和非局部测量的协方差矩阵将局部测量位置与非局部测量位置相结合。

注意：如果没有与非本地轨迹关联的协方差矩阵，则轨迹管理器将尝试使用测量协

方差矩阵，该矩阵由轨迹在范围、高程和方位角方面的测量误差生成。

- mtt

使用多目标跟踪器 （MTT） 融合相关的测量和轨迹。如果选择了此方法，则 correlation_method 也必须为 mtt。

默认值：replacement

message_length <A>

参数列表	参数类型	参数含义
A	数据值大小<data-size-value>	消息长度

指定从映像创建的轨迹消息所分配的逻辑长度。

默认值：0（使用从 message_table 派生的值）

message_priority <A>

参数列表	参数类型	参数含义
A	integer-priority	消息优先级

指定分配给从映像创建的跟踪消息的优先级。

默认值：0（使用从 message_table 派生的值）

WSF_DELAY_PROCESSOR

1、概述

WSF_DELAY_PROCESSOR 用于模拟处理信息需要有限的时间。当处理器接收到消息（例如，图像、轨迹）时，它将保留该消息以模拟处理时间，然后通过其内部和外部链接转发该消息。

2、指令

queuing_method [first_in_first_out | last_in_first_out | none]

指定如果所有服务器都忙碌时，传入消息将如何排队。值为 none 表示消息将被丢弃。

默认值：first_in_first_out

number_of_servers [| infinite]

指定任何给定瞬间可以“正在处理”的消息的最大数量。如果收到新消息且所有服务器都忙碌，该消息将根据 queuing_method 进行排队。

如果指定为“无限”（默认值），接收到的消息只会在转发前延迟所需的时间量。

默认值：infinite

time_distribution constant time A>

参数列表	参数类型	参数含义
A	时间值<time-value>	消息优先级

定义模拟处理时间为指定时间的恒定值。

默认值：默认 time_distribution 为“恒定时间 0 秒”

time_distribution uniform minimum_time <A> maximum_time <B>

参数列表	参数类型	参数含义
A	时间值<time-value>	最小时间值
B	时间值<time-value>	最大时间值

定义消息的模拟处理时间将从指定范围的均匀分布中抽取。

time_distribution gaussian mean_time <A> sigma_time <B>

参数列表	参数类型	参数含义
A	时间值<time-value>	平均时间值

B	时间值<time-value>	标准差时间值

定义消息的模拟时间将从具有指定均值和标准差的高斯分布中抽取。

time_distribution log_normal mean_time <A> sigma_time <B>

参数列表	参数类型	参数含义
A	时间值<time-value>	平均时间值
B	时间值<time-value>	标准差时间值

定义消息的模拟时间将从具有指定均值和标准差的对数正态分布中抽取。

WSF_DIRECTION_FINDER_PROCESSOR

1、概述

WSF_DIRECTION_FINDER_PROCESSOR 是一种专用处理器，用于融合多个仅方位角轨迹（例如，来自一个或多个被动系统）。此功能与标准的 WSF “默认” 融合有所不同。通常，轨迹数据首先被过滤，然后进行融合。在这种情况下，原始的方位角轨迹数据必须先被融合以形成 Location 和 Location Error 数据，然后才进行过滤。方向查找处理器的产物是经过融合过滤的轨迹，具有有效、三角测量的目标位置。

这种过滤是必要的，以便将三角测量误差降低到可操作的值。当一对报告通过三角测量算法运行时，位置误差随着累积对数的平方根而减小。

通常，这个处理器将作为仅方位角传感器的轨迹报告的消费者。因此，传感器将如下所示与方向查找处理器相连：

```
processor direction-finder WSF_DIRECTION_FINDER_PROCESSOR
end_processor

sensor passive_sensor WSF_PASSIVE_SENSOR
   internal_link direction-finder
   ...
end_sensor
```

方向查找处理器的输出是包含有效目标位置数据的轨迹。通常，它将与 WSF_TRACK_PROCESSOR 或与外部融合节点的通信链路相连。

```
processor direction-finder WSF_DIRECTION_FINDER_PROCESSOR
   processor track
   processor share
end_processor

processor track WSF_TRACK_PROCESSOR
end_processor

processor share WSF_LINKED_PROCESSOR
   report_to commander via datalink
end_processor

comm datalink WSF_COMM_TRANSCEIVER
end_comm
```

2、指令

fuse_all_measurements

融合所有测量

fuse_all_collects <A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	融合所有收集的数据

如果启用，对于给定目标的所有测量值都将用于测向。否则，如果在使用这对测量值之后，位置误差的预期值大于当前值，那么这对测量值将不会被融合。

默认值：disable

measurement_replacement_interval

测量替换间隔

collect_replacement_interval A>

参数列表	参数类型	参数含义
A	时间值<time-value>	收集替换间隔

指定一个时间间隔，在此间隔之后，如果现有的测量数据尚未用于方向查找，则将用更近期的测量数据进行更新。

默认值：1.0e+12（Infinite）

maximum_expected_error A>

参数列表	参数类型	参数含义
A	长度值<length-value>	最大预期误差

指定一个在单一维度上的最大值，预期的方向查找误差值不应超过此值。

默认值：100000 m

use_truth_altitude

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	使用真实高度

指定在报告空中目标的位置时应使用实际的高度。

默认值：disabled

filter … end_filter

将过滤器类型与此处理器关联。所有没有滤波器的输入轨道都将分配此 类型。如果未指示此命令，则将使用 WsfKalmanFilter 类型。

默认值：WsfKalmanFilter 带零过程噪声

filter_bypass <flag>

参数列表	参数类型	参数含义
A	string (flag)	旁路过滤器

允许旁路过滤器，这样输出将是“测量”结果，即成功三角测量后的交点

默认值：filter enabled

minimum_baseline_distance A>

参数列表	参数类型	参数含义
A	长度值<length-value>	最小基线距离

指定正在融合的两组测量的起始位置之间的最小距离。

默认值：10 km

maximum_time_difference A>

参数列表	参数类型	参数含义
A	时间值<time-value>	最大时间差异

指定测量之间允许的最大时间差异。超过这个时间差异的测量将不会被融合。这有助于在目标移动时减少误差。如果未包括此字段，则没有最大时间差异的限制——所有对都将被融合，假设其他所有条件都相等。

默认值：1.0e+12 (无限)

test A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	最大时间差异

启用诊断消息的输出和每个测向解决方案的基于 WsfDraw 的表示形式。 当候选测向解决方案由于以下原因而失败时，将输出诊断消息：

- 角度阈值测试失败：两个候选目标矢量形成的内角小于两次测量的最大预期方位角误差的 5 倍。该测试拒绝了范围不确定性非常大的解决方案。
- 基线距离测试失败：两个测量位置的原点距离小于 minimum_baseline_distance 命令指定的距离。
- 最大预期范围误差失败：计算的预期范围误差超过了 maximum_expected_error 输入指定的值。
- 方位线发散失败：两次测量的方位线在目标方向上发散（即，它们在与目标相反的方向上相交）。

基于 WsfDraw 的解决方案可视化显示：沿着轨迹方向的绿色线条，用红色线条表示

预期方位误差的范围，用蓝色线条和点来显示计算出的 3D 解决方案误差的范围，并用一个点来标示计算出的目标位置。

3、使用示例

```
processor <name> WSF_DIRECTION_FINDER_PROCESSOR
   ... WSF_LINKED_PROCESSOR Commands ...
   fuse_all_measurements | fuse_all_collects
   measurement_replacement_interval | collect_replacement_interval
   maximum_expected_error
   use_truth_altitude
   filter
   filter_bypass
   maximum_time_difference
   minimum_baseline_distance
   test
end_processor
```

WSF_DISSEMINATE_C2

来源于：WSF_SCRIPT_PROCESSOR

脚本类：WsfBMDisseminateC2

1、概述

WSF_DISSEMINATE_C2 是一个脚本基类，用于 HELIOS 基础的 C^2（指挥与控制）传播模型继承。它本身不作为处理器使用，而是传播 C^2 模型使用它来提供所有传播处理器中常见的通用脚本功能。

2、传播 C2 指令

routing_style [next_unit | next_c2 | direct]

用于确定 IADS 网络中的下一跃点的消息路由样式。

next_unit：消息将在到达目的地的途中发送到下一个单元。

next_c2：消息将在到达目的地的途中发送到下一个被指定为支持 C^2 的单元。

注意：C^2 能力通常在 IADS（集成防空系统）初始化时设置，并对于在战斗管理器中初始化的任何资产，如果已在 WsfBMAssetRecord 上调用了 WsfBMAssetRecord.SetC2Capable 方法，则该能力将为真。

direct：消息将直接发送到消息目标。

默认值：**next_c2**

4、路由表指令

定义 HELIOS C^2 传播消息路由表的区块。所有值可能具有以下值之一：[peer | commander | dynamic | none]。

peer：将消息发送给该资源的同行。

commander：将消息发送给该资产的指挥官。

dynamic：将消息发送到消息目的地的路线上（用于像分配和分配状态这样的定向消息）。

none：不发送消息。

track_updates [peer | commander | dynamic | none]

这条命令设置了轨迹消息的路由。

注意：在 AFSIM 中，轨迹消息的处理通常由轨迹处理系统负责，而不是战斗管理器组件。

默认值：none

assign_track_updates [peer | commander | dynamic | none]

这条命令设置了分配轨迹的路由。

注意：这是一个定向消息，因此通常设置为“dynamic”（动态）。

默认值：dynamic

assignments [peer | commander | dynamic | none]

这条命令设置了分配的路由。

注意：这是一个定向消息，因此通常设置为“dynamic”。这意味着路由将根据消息的目的地动态确定，以确保信息能够沿着最佳或指定的路径传递。

默认值：dynamic

assignment_status [peer | commander | dynamic | none]

这条命令设置了分配状态的路由。

注意：这是一个定向消息，因此通常设置为“dynamic”。这意味着路由将基于消息的目的地来动态确定，以便消息能够按照预期的路径发送到正确的接收者。

默认值：dynamic

assignment_cancel [peer | commander | dynamic | none]

这条命令设置了分配取消的路由。

注意：这是一个定向消息，因此通常设置为“dynamic”。这意味着消息的路由将根据其目的地动态确定，以确保取消指令能够准确快速地传达给相关接收者。

默认值：dynamic

sensor_cue [peer | commander | dynamic | none]

这条命令设置了传感器提示的路由。

注意：这是一个定向消息，因此通常设置为“dynamic”。这意味着传感器提示的路由将根据其预定的接收者动态确定，确保信息能够直接发送到需要这些提示的实体。

默认值：dynamic

status [peer | commander | dynamic | none]

这条命令设置了资产状态的路由。

注意：这通常会被设置为“commander”（指挥官）。这意味着资产状态信息默认会被发送给资产的指挥官，以便进行监督和管理。

默认值：none

5、使用示例

```
processor <name> WSF_DISSEMINATE_C2
   WSF_SCRIPT_PROCESSOR Commands ...

   routing_style [next_unit|next_c2|direct]

   routing_table
      track_updates          [peer|commander|dynamic|none]
      assign_track_updates   [peer|commander|dynamic|none]
      assignments            [peer|commander|dynamic|none]
      assignment_status      [peer|commander|dynamic|none]
      assignment_cancel      [peer|commander|dynamic|none]
      sensor_cue             [peer|commander|dynamic|none]
      status                 [peer|commander|dynamic|none]
   end_routing_table
end_processor
```

```
routing_table
   # Routing values allowed are subordinate, peer, commander, dynamic, none. multiple entries can exist
   # for a given type of message.
   track_updates          none
   assign_track_updates   dynamic
   assignments            dynamic
   assignment_status      dynamic
   assignment_cancel      dynamic
   sensor_cue             dynamic
   status                 commander
end_routing_table
```

WSF_DUMP_MESSAGE_PROCESSOR

1、概述

此处理器从 1.7.4 版本开始已弃用。

WSF_DUMP_MESSAGE_PROCESSOR 是一个非常简单的处理器，非常适合用于调试。它接收消息，打印有关消息的信息，然后通过其内部和外部链接转发数据。

WSF_EXCHANGE_PROCESSOR

WSF_FUSION_CENTER

1、概述

WSF_FUSION_CENTER 相关联来自多个不同雷达站点的轨迹。必须定义它以处理来自 false-target jamming effect(虚假目标干扰效应)的 false_target（虚假目标）。

2、指令

plot_capacity

定义可在雷达示波器上绘制的目标数。

默认值：2000

frame_time

定义融合中心的更新频率，单位为秒。

默认值：5.0 sec

track_capacity

定义融合中心可以维护的轨迹数。

默认值：500

random_to_multiple_radars

设定任何雷达上出现的虚假目标中，将被识别并排除的比例。

默认值：0.0

consistent_to_multiple_radars

定义出现在多个雷达上的将被拒绝的 false target 的比率。

默认值：0.0

radar_site

指定要进行融合的雷达站点的名称。重复此命令以指定每个雷达站点。名称指的是雷达平台的玩家名称。

consistency_constrained

定义是否必须由所有雷达站点检测到轨迹，才能由融合中心进行融合。如果为假（False），则意味着轨迹只需要出现在一个雷达站点上即可。

默认值：true

WSF_GROUND_TARGET_FUSE

1、概述

WSF_GROUND_TARGET_FUSE 是一种特殊的 WSF_WEAPON_FUSE，具有适用于武器的默认值，即 与地面目标交战。以下其他默认值为：

air_to_ground_mode

use_current_target

detonate_below_height_agl 0.0 米

gross_proximity_range 500 米

各默认值详见**错误!未找到引用源。**。

WSF_GUIDANCE_COMPUTER

来源于：WSF_SCRIPT_PROCESSOR

脚本类：WsfBattleManager

1、概述

WSF_GUIDANCE_COMPUTER 是一个处理器，通常装配在武器上，为武器提供指导，其移动器通常为 WSF_GUIDED_MOVER 类型。它使用通过 CurrentTargetTrack 提供的轨迹来表示要追踪的目标。移动者调用此处理器请求指导更新。处理器计算所需的指导并将其反馈给 Mover。

2、全局命令

注意：不应为此处理器指定 update_interval 命令。它根据需要从 mover 中调用。

show_status

指定每当阶段或阶段转换发生时，应将相关信息输出到标准输出设备。

show_diagnostics

指定应将指导程序中的诊断信息写入标准输出。这通常用于当尝试调整制导计算机以生成具有 weapon_tools 的轨道发射计算机时。

注意：这会带来一些额外的开销，应仅在优化期间使用。

show_commands

指定从脚本命令发出的调用应写入标准输出。

show_evaluations

指定应将相变规则评估写入标准输出。

guide_to_truth A>

参数列表	参数类型	参数含义
A	布尔值<boolean-value>	

这个选项用于指定是否应该使用当前目标轨迹中指定的位置（如果设置为 false），或者使用当前目标轨迹中指定的平台的真实位置（如果设置为 true）。这个命令在两个地方存在：一次是在这里（可能是在某个全局设置或默认设置中），另一次是在“阶段”（phase）块中。如果在“阶段”块中没有明确指定这个命令，那么这里的形式就指定了该阶段的默认值。

默认值：true

phase <A> phase-commands end_phase

参数列表	参数类型	参数含义
A	phase-name <String>	阶段块的名称

它用于定义飞行各个阶段的制导规则以及阶段之间转换的规则。

阶段块的格式是：

```
phase <phase-name>
   ... phase commands ...
end_phase
```

“阶段”块应该为所需的每个独特阶段定义。第一个“阶段”块定义了武器发射时所使用的阶段。

edit phase <A> phase-commands end_phase

参数列表	参数类型	参数含义
A	phase-name <String>	阶段块的名称

这条命令通常用于通过派生自另一种制导计算机类型来创建新的制导计算机类型时。

示例：

```
processor DEMO_GUIDANCE WSF_GUIDANCE_COMPUTER
   guide_to_truth true
   phase TERMINAL
      ...
   end_phase
end_processor

processor DEMO_GUIDANCE_MOD DEMO_GUIDANCE
   edit phase TERMINAL
      guide_to_truth false      # Overrides the base class value
   end_phase
end_processor
```

program <name> <type> … end_program

定义一个具有指定 <name> 的制导程序，使用预定义的制导程序 <type>（参见 Guidance Program Types）。然后，可以通过指定 use_program <name> 在某个阶段选择使用该程序。

注意：程序必须出现在引用该程序的 use_program 之前。

例如：

```
program RESET_ATTITUDE ATTITUDE_PROGRAM
   reset
   yaw_rate 5 deg/sec
   pitch_rate 5 deg/sec
   roll_rate 5 deg/sec
end_program
...
phase BEGIN_DESCENT
   use_program RESET_ATTITUDE
   next_phase DESCENT when program RESET_ATTITUDE complete
end_phase
...
phase DESCENT
   ...
end_phase
```

3、阶段（phase）命令

phase 块中的子命令可以分为以下几大类：

- 常规子命令
- Aimpoint Selection 子命令 - 指定如何确定目标点
- 程序选择子命令 - 选择或定义要执行的指导程序。
- 导航航子命令 - 指定如何导航到目标点。
- Trajectory Shaping Subcommands 指定在导航时如何修改轨迹。
- 限制子命令 - 指定计算限制。
- Phase Changing 子命令根据许多条件过渡到其他阶段。

4、常规子命令

guidance_delay A>

参数列表	参数类型	参数含义
A	时间值 <time-value>	指定从阶段开始到应该开始计算制导命令的经过时间。

指定从阶段开始到应该开始计算制导命令的经过时间。这在起飞和其他不希望追求目标的阶段非常有用。

默认值：0 秒（进入阶段后开始计算引导命令）

on_entry ...end_on_entry

on_exit ...end_on_exit

这些命令定义在进入和退出阶段时要执行的脚本。

```
on_entry
   ... script commands ...
end_on_entry
```

```
on_exit
   ... script commands ...
end_on_exit
```

on_update ...

定义要在每个阶段更新时执行的脚本。

```
on_update
   ... script commands ...
end_on_update
```

此脚本通常仅在以下情况下使用：

- 在一个阶段中，指导命令的值必须发生变化
- 需要评估无法通过 next_phase 命令完成的相变条件。

注意：不要随意使用这个命令，因为它会在移动器的每个积分时间步骤中执行。尽量保持脚本的简洁性。

5、Aimpoint Selection 子命令

guidance_target [predicted_intercept | perception | truth]