

威胁源仿真库文档

欢迎使用威胁源仿真库文档。

简介

威胁源仿真库是一个用于模拟和仿真各种威胁源的.NET类库。它提供了以下主要功能：

- 导弹仿真（包括各种类型的导弹）
- 目标仿真
- 传感器仿真
- 制导系统仿真

支持的开发语言

- C# (.NET): 原生支持，提供完整的API和功能
- C++: 通过C++/CLI包装层支持，可以在C++项目中使用全部功能

快速开始

请参阅[入门指南](#)了解如何使用该库。该指南包含了C#和C++两种语言的使用示例。

API文档

完整的API文档请参阅[API参考](#)。

集成示例

第三方引擎集成

- [集成示例说明](#) - 第三方引擎集成的详细说明
- [虚幻引擎示例](#) - 虚幻引擎集成示例代码
- [Unity引擎示例](#) - Unity引擎集成示例代码

示例说明

这些示例展示了如何将不同的游戏引擎与仿真系统集成：

1. 虚幻引擎(UE)示例
 - 虚幻引擎与仿真系统的双向通信
 - 实体信息的同步和转换
 - 事件的发布和订阅
 - 数据的适配和转换
2. Unity引擎示例
 - Unity引擎与仿真系统的双向通信
 - GameObject与实体的映射和转换
 - MonoBehaviour生命周期管理
 - 事件系统的使用

所有示例代码都提供了详细的注释和说明，可以作为实际开发的参考。

入门指南

安装和使用

C#/.NET 版本

1. 下载 ThreatSourceLibrary 包并解压
2. 将以下文件复制到你的项目目录（建议放在 lib 文件夹下）：
 - ThreatSource.dll - 主要的库文件
 - ThreatSource.deps.json - 依赖配置文件
 - ThreatSource.xml - API 文档文件
3. 在 Visual Studio 中添加引用：
 - 右键项目 -> 添加 -> 引用
 - 浏览 -> 选择 ThreatSource.dll
4. 在代码中使用：

```

using ThreatSource.Simulation;

class Program
{
    static void Main()
    {
        try
        {
            // 创建仿真管理器
            var simulationManager = new SimulationManager();

            // 创建导弹配置
            var missileProperties = new MissileProperties
            {
                Id = "missile1",
                InitialPosition = new Vector3D(0, 0, 0),
                InitialVelocity = new Vector3D(0, 0, 0),
                MaxSpeed = 1000,
                MaxAcceleration = 10,
                MaxFlightTime = 30,
                MaxFlightDistance = 5000,
                Mass = 50
            };

            // 创建导弹实例
            var missile = new TerminalSensitiveMissile(
                "target1",
                missileProperties,
                simulationManager
            );

            // 注册实体
            simulationManager.RegisterEntity(missile.Id, missile);

            // 激活并发射导弹
            missile.Activate();
            missile.Fire();

            // 仿真主循环
            double deltaTime = 0.01;
            while (missile.IsActive)
            {
                missile.Update(deltaTime);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }
}

```

C++版本

C++项目有两种使用方式：

方式一：动态加载（推荐）

这种方式适合纯C++项目，不需要配置CLR支持。

1. 下载 ThreatSourceNative 包并解压
2. 将 bin 目录下的所有 DLL 文件复制到程序目录
3. 将 include/threat_source.h 添加到项目中
4. 使用 C 风格接口调用库功能

示例代码：

```

#include "threat_source.h"
#include <stdio.h>

int main() {
    // 初始化仿真
    if (TS_CreateSimulation() != THREATSOURCE_SUCCESS) {
        printf("Failed to create simulation\n");
        return 1;
    }

    // 创建导弹
    const char* missile_id = "missile1";
    int result = TS_CreateMissile(
        missile_id,
        0, 0, 0,      // 初始位置
        0, 0, 0,      // 初始速度
        1000,         // 最大速度
        10,           // 最大加速度
        30,           // 最大飞行时间
        5000,         // 最大飞行距离
        50            // 质量
    );

    if (result != THREATSOURCE_SUCCESS) {
        char error[256];
        TS_GetLastError(error, sizeof(error));
        printf("Failed to create missile: %s\n", error);
        return 1;
    }

    // 激活并发射导弹
    TS_ActivateMissile(missile_id);
    TS_FireMissile(missile_id);

    // 仿真主循环
    double deltaTime = 0.01;
    int is_active = 1;

    while (is_active) {
        TS_UpdateSimulation(deltaTime);
        TS_IsMissileActive(missile_id, &is_active);
    }

    // 清理
    TS_DestroySimulation();
    return 0;
}

```

方式二：CLR 集成

如果需要使用 .NET 的完整功能，可以选择 CLR 集成方式：

1. 配置项目属性：
 - C/C++ -> 常规 -> 公共语言运行时支持：/clr
 - 常规 -> 平台工具集：Visual Studio 2022 (v143)
 - 常规 -> .NET目标框架：net8.0
2. 引用 ThreatSource.dll

示例代码：

```

using namespace System;
using namespace ThreatSource::Simulation;

int main() {
    try {
        // 创建仿真管理器
        auto manager = gcnew SimulationManager();

        // 创建导弹配置
        auto properties = gcnew MissileProperties();
        properties->Id = "missile1";
        properties->InitialPosition = Vector3D(0, 0, 0);
        properties->InitialVelocity = Vector3D(0, 0, 0);
        properties->MaxSpeed = 1000;
        properties->MaxAcceleration = 10;
        properties->MaxFlightTime = 30;
        properties->MaxFlightDistance = 5000;
        properties->Mass = 50;

        // 创建导弹
        auto missile = gcnew TerminalSensitiveMissile("target1", properties, manager);
        manager->RegisterEntity(missile->Id, missile);

        // 激活并发射导弹
        missile->Activate();
        missile->Fire();

        // 仿真主循环
        double deltaTime = 0.01;
        while (missile->IsActive) {
            missile->Update(deltaTime);
        }

        return 0;
    }
    catch (Exception^ e) {
        Console::WriteLine("Error: {0}", e->Message);
        return 1;
    }
}

```

基本用法

初始化仿真管理器

```
var simulationManager = new SimulationManager();
```

创建导弹

```
// 创建导弹配置
var missileProperties = new MissileProperties
{
    Id = "missile1",
    InitialPosition = new Vector3D(0, 0, 0),
    InitialOrientation = new Orientation(0, 0, 0),
    InitialSpeed = 800,
    MaxSpeed = 1000,
    MaxFlightTime = 30,
    MaxFlightDistance = 5000,
    MaxAcceleration = 10,
    Mass = 50
};

// 创建导弹实例
var missile = new TerminalSensitiveMissile("target1", missileProperties, simulationManager);
```

创建目标

```
// 创建目标
var tank = new Tank(
    "target1",
    new Vector3D(1000, 0, 0),
    Math.PI/4,
    simulationManager
);
```

注册实体

```
simulationManager.RegisterEntity(missile.Id, missile);
simulationManager.RegisterEntity(tank.Id, tank);
```

运行仿真

```
// 激活实体
missile.Activate();
tank.Activate();

// 发射导弹
missile.Fire();

// 仿真主循环
double deltaTime = 0.01;
while (missile.IsActive)
{
    missile.Update(deltaTime);
    tank.Update(deltaTime);
}
```

C++使用指南

C++/CLI基本用法

1. 创建包装类

```

using namespace System;
using namespace ThreatSource::Simulation;

public ref class SimulationWrapper
{
private:
    SimulationManager^ manager;

public:
    SimulationWrapper()
    {
        manager = gcnew SimulationManager();
    }

    void CreateAndRunSimulation()
    {
        try
        {
            // 创建导弹配置
            auto properties = gcnew MissileProperties();
            properties->Id = "missile1";
            properties->InitialPosition = Vector3D(0, 0, 0);
            properties->InitialSpeed = 800;
            properties->MaxSpeed = 1000;
            properties->MaxFlightTime = 30;
            properties->MaxFlightDistance = 5000;
            properties->MaxAcceleration = 10;
            properties->Mass = 50;

            // 创建导弹
            auto missile = gcnew TerminalSensitiveMissile("target1", properties, manager);

            // 注册和激活
            manager->RegisterEntity(missile->Id, missile);
            missile->Activate();
            missile->Fire();

            // 仿真主循环
            double deltaTime = 0.01;
            while (missile->IsActive)
            {
                missile->Update(deltaTime);
            }
        }
        catch (Exception^ e)
        {
            Console::WriteLine("错误: {0}", e->Message);
        }
    }
};

```

2. 在原生C++代码中使用


```
int main()
{
    try
    {
        auto simulation = gcnew SimulationWrapper();
        simulation->CreateAndRunSimulation();
        return 0;
    }
    catch (Exception^ e)
    {
        Console::WriteLine("错误: {0}", e->Message);
        return 1;
    }
}
```

注意事项

1. 内存管理
 - C++/CLI使用垃圾回收
 - 使用gcnew创建托管对象
 - 注意托管和非托管资源的混合使用
2. 错误处理
 - 使用托管异常处理 (try/catch)
 - 异常信息更详细, 更容易调试
 - 可以直接使用.NET的日志机制
3. 类型系统
 - 使用托管类型 (^)
 - 注意值类型和引用类型的区别
 - 使用安全的类型转换

更多示例

更多示例请参考[示例代码](#)。

第三方引擎集成示例

本目录包含了将仿真系统与第三方引擎集成的示例代码。这些示例展示了如何使用适配器模式将不同的游戏引擎与仿真系统进行集成。

示例文件

UEExample.cs

虚幻引擎(Unreal Engine)集成示例，展示了：

- 虚幻引擎与仿真系统的双向通信
- 实体信息的同步和转换
- 事件的发布和订阅
- 数据的适配和转换

源代码

```

#if NEVER // 使用编译指令确保此文件永远不会被编译

// 第三方系统实现适配器示例代码
// 以虚幻引擎为例
// 需要实现的方法: GetEntity, PublishToExternalSimulation, ReceiveFromExternalSimulation

using ThreatSource.Simulation;

/// <summary>
/// 虚幻引擎适配器类, 演示如何将第三方系统集成到仿真系统中
/// </summary>
public class UnrealEngineAdapter : ISimulationAdapter
{
    /// <summary>
    /// 虚幻引擎API接口, 定义了与虚幻引擎交互的基本方法
    /// </summary>
    public interface IUnrealEngine
    {
        /// <summary>
        /// 根据ID获取虚幻引擎中的Actor对象
        /// </summary>
        object? GetActor(string id);

        /// <summary>
        /// 在虚幻引擎中生成导弹实体
        /// </summary>
        void SpawnMissile(string senderId, string targetId);
    }

    private readonly IUnrealEngine _unrealEngine;
    private readonly ISimulationManager _simulationManager;

    public UnrealEngineAdapter(IUnrealEngine unrealEngine, ISimulationManager simulationManager)
    {
        _unrealEngine = unrealEngine ?? throw new ArgumentNullException(nameof(unrealEngine));
        _simulationManager = simulationManager ?? throw new ArgumentNullException(nameof(simulationManager));
    }

    public object? GetEntity(string id)
    {
        return _unrealEngine.GetActor(id);
    }

    public void PublishToExternalSimulation<T>(T evt)
    {
        if (evt is MissileFireEvent missileEvt)
        {
            _unrealEngine.SpawnMissile(missileEvt.SenderId, missileEvt.TargetId);
        }
    }

    public void ReceiveFromExternalSimulation<T>(T evt)
    {
        // 处理来自虚幻引擎的事件
    }
}

#endif

```

UnityExample.cs

Unity引擎集成示例, 展示了:

- Unity引擎与仿真系统的双向通信
- GameObject与实体的映射和转换
- MonoBehaviour生命周期管理
- 事件系统的使用

源代码

```
#if NEVER // 使用编译指令确保此文件永远不会被编译

// 第三方系统实现适配器示例代码
// 以Unity引擎为例
// 需要实现的方法: GetEntity, PublishToExternalSimulation, ReceiveFromExternalSimulation

using ThreatSource.Simulation;
using UnityEngine; // 仅用于示例, 实际项目中需要引用真实的Unity命名空间

/// <summary>
/// Unity引擎适配器类, 演示如何将Unity引擎集成到仿真系统中
/// </summary>
public class UnityEngineAdapter : MonoBehaviour, ISimulationAdapter
{
    /// <summary>
    /// Unity引擎API接口, 定义了与Unity引擎交互的基本方法
    /// </summary>
    public interface IUnityEngine
    {
        /// <summary>
        /// 根据ID获取Unity场景中的GameObject对象
        /// </summary>
        GameObject GetGameObject(string id);

        /// <summary>
        /// 在Unity场景中实例化导弹预制体
        /// </summary>
        GameObject InstantiateMissile(string prefabPath, Vector3 position, Quaternion rotation);
    }

    private readonly IUnityEngine _unityEngine;
    private readonly ISimulationManager _simulationManager;
    private const string MissilePrefabPath = "Prefabs/Missile";

    public UnityEngineAdapter(IUnityEngine unityEngine, ISimulationManager simulationManager)
    {
        _unityEngine = unityEngine ?? throw new ArgumentNullException(nameof(unityEngine));
        _simulationManager = simulationManager ?? throw new ArgumentNullException(nameof(simulationManager));
    }

    public object? GetEntity(string id)
    {
        return _unityEngine.GetGameObject(id);
    }

    public void PublishToExternalSimulation<T>(T evt)
    {
        if (evt is MissileFireEvent missileEvt)
        {
            var sender = _unityEngine.GetGameObject(missileEvt.SenderId);
            if (sender != null)
            {
                UnityMainThreadDispatcher.Instance.Enqueue(() =>
                {
                    var missile = _unityEngine.InstantiateMissile(
                        MissilePrefabPath,
                        sender.transform.position,
```

```

        sender.transform.rotation
    );

    var missileComponent = missile.GetComponent<MissileController>();
    if (missileComponent != null)
    {
        missileComponent.SetTarget(missileEvt.TargetId);
    }
    });
}

}

}

public void ReceiveFromExternalSimulation<T>(T evt)
{
    if (evt is CollisionEvent collisionEvt)
    {
        _simulationManager.HandleCollision(collisionEvt);
    }
}

private void Update()
{
    SyncSimulationState();
}

private void SyncSimulationState()
{
    // 实现仿真状态同步逻辑
}
}

#endif

```

使用说明

1. 这些文件仅作为参考示例，不参与实际编译（使用 `#if NEVER` 编译指令）
2. 实际项目中需要根据具体需求修改和扩展
3. 示例中的接口和类名仅供参考，应根据实际项目规范调整

关键概念

适配器模式

- 实现 `ISimulationAdapter` 接口
- 转换不同引擎的数据格式
- 处理事件的发布和订阅

实体映射

- 在仿真系统和游戏引擎之间建立实体对应关系
- 同步实体状态和属性
- 处理实体的创建和销毁

事件系统

- 定义统一的事件数据结构
- 处理事件的双向转换
- 确保事件的正确分发

注意事项

1. 需要处理线程安全问题
2. 注意性能优化，特别是在状态同步时
3. 合理处理资源的加载和释放
4. 确保异常处理的完整性

导弹仿真示例

本目录包含了使用仿真系统进行导弹仿真的示例代码。这些示例展示了如何配置和运行不同类型的导弹仿真。

关于本库

ThreatSource 是一个基于 .NET 8.0 的类库，提供了完整的导弹仿真功能。

系统要求

C#/.NET 用户

- .NET 8.0 或更高版本
- 通过 NuGet 包管理器安装或直接引用 ThreatSource.dll

C++用户

本库是一个 .NET 类库，C++用户需要通过 C++/CLI 包装层来使用：

- Windows 操作系统
- Visual Studio 2019 或更高版本
- 创建 C++/CLI 项目并引用 ThreatSource.dll

示例文件

C#示例 ([IRMissileSimulation.cs](#))

红外成像制导导弹仿真示例，展示了：

- 如何创建和配置导弹实体
- 如何设置红外成像制导系统
- 如何配置仿真环境和参数
- 如何运行仿真并获取结果

源代码

```
using ThreatSource.Simulation;
using ThreatSource.Missile;
using ThreatSource.Sensor;
using ThreatSource.Target;

/// <summary>
/// 红外成像制导导弹仿真示例
/// </summary>
public class IRMissileSimulation
{
    private readonly ISimulationManager _simulationManager;
    private readonly IMissileFactory _missileFactory;
    private readonly ITargetFactory _targetFactory;
    private readonly ISensorFactory _sensorFactory;
```

```

public IRMissileSimulation()
{
    _simulationManager = new SimulationManager();
    _missileFactory = new MissileFactory();
    _targetFactory = new TargetFactory();
    _sensorFactory = new SensorFactory();
}

/// <summary>
/// 运行仿真
/// </summary>
public async Task RunSimulation()
{
    // 创建目标
    var target = _targetFactory.CreateTarget(new TargetConfig
    {
        Id = "target_001",
        Position = new Vector3(1000, 0, 100),
        Velocity = new Vector3(-100, 0, 0),
        Signature = new IRSignature
        {
            Temperature = 350, // 开尔文
            EmissivityFactor = 0.8f
        }
    });

    // 创建导弹
    var missile = _missileFactory.CreateMissile(new MissileConfig
    {
        Id = "missile_001",
        Position = new Vector3(0, 0, 0),
        MaxSpeed = 800, // 米/秒
        MaxAcceleration = 30, // G
        MaxTurnRate = 20 // 度/秒
    });

    // 创建红外成像传感器
    var sensor = _sensorFactory.CreateSensor(new IRSensorConfig
    {
        Id = "sensor_001",
        Resolution = new Vector2(640, 480),
        FieldOfView = 60, // 度
        MinTemperature = 270, // 开尔文
        MaxTemperature = 400 // 开尔文
    });

    // 配置仿真参数
    var config = new SimulationConfig
    {
        TimeStep = 0.02f, // 仿真步长 (秒)
        MaxSimulationTime = 60.0f, // 最大仿真时间 (秒)
        EnvironmentConditions = new EnvironmentConfig
        {
            Temperature = 288, // 开尔文
            Pressure = 101325, // 帕斯卡
            Humidity = 0.5f // 相对湿度
        }
    };

    // 初始化仿真
    await _simulationManager.Initialize(config);

    // 添加实体
    _simulationManager.AddEntity(target);
    _simulationManager.AddEntity(missile);
    _simulationManager.AddEntity(sensor);
}

```



```

        // 启动仿真
        await _simulationManager.StartSimulation();

        // 等待仿真完成
        while (_simulationManager.IsRunning)
        {
            await Task.Delay(100);
        }

        // 获取仿真结果
        var results = _simulationManager.GetSimulationResults();
        ProcessResults(results);
    }

    private void ProcessResults(SimulationResults results)
    {
        // 处理仿真结果
        Console.WriteLine($"仿真完成时间: {results.CompletionTime}秒");
        Console.WriteLine($"命中精度: {results.HitAccuracy}米");
        Console.WriteLine($"导引头跟踪时间: {results.TrackingTime}秒");
    }
}

```

C++示例 (IRMissileSimulation.cpp)

这是一个使用C++/CLI的示例代码，展示了如何在C++项目中使用本库：

- 如何创建C++/CLI包装层
- 如何配置导弹实体
- 如何设置仿真参数
- 如何运行仿真并获取结果

源代码

```

#include "ThreatSource.h"
using namespace System;
using namespace ThreatSource::Simulation;
using namespace ThreatSource::Missile;
using namespace ThreatSource::Sensor;
using namespace ThreatSource::Target;

/// <summary>
/// C++/CLI包装类，用于在C++项目中使用仿真系统
/// </summary>
public ref class SimulationWrapper
{
private:
    ISimulationManager^ _simulationManager;
    IMissileFactory^ _missileFactory;
    ITargetFactory^ _targetFactory;
    ISensorFactory^ _sensorFactory;

public:
    SimulationWrapper()
    {
        _simulationManager = gcnew SimulationManager();
        _missileFactory = gcnew MissileFactory();
        _targetFactory = gcnew TargetFactory();
        _sensorFactory = gcnew SensorFactory();
    }

    void RunSimulation()
    {
        // 创建目标
        auto target = _targetFactory->CreateTarget(gcnew TargetConfig {
            Id = "target_001",

```

```

        Position = Vector3(1000, 0, 100),
        Velocity = Vector3(-100, 0, 0),
        Signature = gcnew IRSignature {
            Temperature = 350,
            EmissivityFactor = 0.8f
        }
    });

    // 创建导弹
    auto missile = _missileFactory->CreateMissile(gcnew MissileConfig {
        Id = "missile_001",
        Position = Vector3(0, 0, 0),
        MaxSpeed = 800,
        MaxAcceleration = 30,
        MaxTurnRate = 20
    });

    // 创建传感器
    auto sensor = _sensorFactory->CreateSensor(gcnew IRSensorConfig {
        Id = "sensor_001",
        Resolution = Vector2(640, 480),
        FieldOfView = 60,
        MinTemperature = 270,
        MaxTemperature = 400
    });

    // 配置仿真参数
    auto config = gcnew SimulationConfig {
        TimeStep = 0.02f,
        MaxSimulationTime = 60.0f,
        EnvironmentConditions = gcnew EnvironmentConfig {
            Temperature = 288,
            Pressure = 101325,
            Humidity = 0.5f
        }
    };

    // 初始化仿真
    _simulationManager->Initialize(config)->Wait();

    // 添加实体
    _simulationManager->AddEntity(target);
    _simulationManager->AddEntity(missile);
    _simulationManager->AddEntity(sensor);

    // 启动仿真
    _simulationManager->StartSimulation()->Wait();

    // 等待仿真完成
    while (_simulationManager->IsRunning)
    {
        System::Threading::Thread::Sleep(100);
    }

    // 获取仿真结果
    auto results = _simulationManager->GetSimulationResults();
    ProcessResults(results);
}

private:
    void ProcessResults(SimulationResults^ results)
    {
        Console::WriteLine("仿真完成时间: {0}秒", results->CompletionTime);
        Console::WriteLine("命中精度: {0}米", results->HitAccuracy);
        Console::WriteLine("导引头跟踪时间: {0}秒", results->TrackingTime);
    }
};

```

```
// 在C++代码中使用
int main()
{
    auto simulation = gcnew SimulationWrapper();
    simulation->RunSimulation();
    return 0;
}
```

使用说明

C#/.NET使用方式

1. 创建仿真适配器

```
var adapter = new TestSimulationAdapter();
```

2. 配置导弹和目标

```
var missile = new SimulationEntity
{
    Id = "missile_001",
    Position = new Vector3(0, 0, 0),
    // ... 其他配置
};
```

3. 设置仿真参数

```
var simConfig = new SimulationConfig
{
    TimeStep = 0.02f, // 仿真步长 (秒)
    MaxSimulationTime = 60.0f, // 最大仿真时间 (秒)
    // ... 其他配置
};
```

4. 运行仿真

```
await adapter.Initialize(simConfig);
await adapter.StartSimulation();
```

C++使用方式

1. 创建C++/CLI项目

- 在Visual Studio中创建新的C++/CLI项目
- 添加对 ThreatSource.dll 的引用

2. 创建包装类

```
public ref class SimulationWrapper
{
private:
    TestSimulationAdapter^ adapter;

public:
    SimulationWrapper()
    {
        adapter = gcnew TestSimulationAdapter();
    }

    // ... 其他包装方法
};
```

3. 在C++代码中使用

```
auto simulation = gcnew SimulationWrapper();
simulation->Initialize();
simulation->StartSimulation();
```

注意事项

1. 确保所有参数单位正确（米、秒、开尔文等）
2. 合理设置仿真时间步长，平衡精度和性能
3. 注意处理异步操作和事件回调
4. C++/CLI注意事项：
 - 仅支持Windows平台
 - 需要正确配置项目的目标框架
 - 注意托管和非托管资源的正确释放

Class MissileProperties

导弹配置类，定义了导弹的所有基本属性和性能参数

Inheritance

↳ [object](#)
↳ MissileProperties

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: ThreatSource.dll

Syntax

```
public class MissileProperties
```

Remarks

该类用于：

- 初始化导弹的基本参数
- 配置导弹的性能限制
- 设置导弹的物理特性
- 定义导弹的作战能力 所有导弹实例都基于此配置进行初始化

Constructors

MissileProperties()

初始化导弹配置类的新实例

Declaration

```
public MissileProperties()
```

Remarks

构造过程：

- 设置所有数值参数的默认值
- 初始化基本属性（ID、位置、朝向）
- 设置默认导弹类型

Properties

ExplosionRadius

获取或设置导弹的爆炸半径

Declaration

```
public double ExplosionRadius { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米 爆炸效果的有效作用范围 用于计算对目标的伤害

HitProbability

获取或设置导弹的命中概率

Declaration

```
public double HitProbability { get; set; }
```

Property Value

Type	Description
double	

Remarks

范围：0-1 表示在理想条件下的命中可能性 用于评估导弹的作战效能

Id

获取或设置导弹的唯一标识符

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和追踪特定的导弹实例

InitialOrientation

获取或设置导弹的初始朝向

Declaration

```
public Orientation InitialOrientation { get; set; }
```

Property Value

Type	Description
Orientation	

Remarks

使用欧拉角表示初始姿态 包含偏航角、俯仰角和滚转角

InitialPosition

获取或设置导弹的初始位置

Declaration

```
public Vector3D InitialPosition { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

使用三维向量表示发射位置 坐标系：右手坐标系，X向右，Y向上，Z向前

InitialSpeed

获取或设置导弹的初始速度

Declaration

```
public double InitialSpeed { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米/秒 发射时的初始运动速度

LaunchAcceleration

获取或设置导弹的发射推力加速度

Declaration

```
public double LaunchAcceleration { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米/秒² 发射阶段的推进加速度 影响导弹的起飞性能

Mass

获取或设置导弹的质量

Declaration

```
public double Mass { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：千克 导弹的总质量，包括弹体和燃料 影响导弹的运动性能和惯性特性

MaxAcceleration

获取或设置导弹的最大加速度

Declaration

```
public double MaxAcceleration { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米/秒² 导弹机动时的最大加速度限制 基于导弹的结构强度和推进系统能力

MaxEngineBurnTime

获取或设置发动机的最大燃烧时间

Declaration

```
public double MaxEngineBurnTime { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 发动机能够持续工作的最长时间 基于燃料容量和燃烧速率

MaxFlightDistance

获取或设置导弹的最大飞行距离

Declaration

```
public double MaxFlightDistance { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米 超过此距离导弹将自毁 基于导弹的燃料容量和设计射程

MaxFlightTime

获取或设置导弹的最大飞行时间

Declaration

```
public double MaxFlightTime { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 超过此时间导弹将自毁 用于防止导弹无限飞行

MaxSpeed

获取或设置导弹的最大速度限制

Declaration

```
public double MaxSpeed { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米/秒 导弹在飞行过程中不能超过此速度

ProportionalNavigationCoefficient

获取或设置导弹的比例导引系数

Declaration

```
public double ProportionalNavigationCoefficient { get; set; }
```

Property Value

Type	Description
double	

Remarks

无量纲参数 用于计算制导指令的增益 影响导弹的制导精度和稳定性

Type

获取或设置导弹的类型

Declaration

```
public MissileType Type { get; set; }
```

Property Value

Type	Description
MissileType	

Remarks

决定导弹的制导方式和行为特征 影响导弹的性能参数和作战能力

Methods

SetDefaultValues()

将所有数值参数设置为默认值

Declaration

```
public void SetDefaultValues()
```

Remarks

重置以下参数：

- 速度相关参数
- 时间和距离限制
- 加速度参数
- 性能参数 用于初始化或重置导弹配置

Class LaserBeamUpdateEvent

激光波束更新事件，表示激光波束位置更新

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserBeamUpdateEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class LaserBeamUpdateEvent : SimulationEvent
```

Remarks

用于实时更新激光波束的位置和方向 在制导过程中周期性触发

Properties

LaserBeamRiderId

获取或设置激光波束制导器的ID

Declaration

```
public string? LaserBeamRiderId { get; set; }
```

Property Value

Type	Description
string	

Class TerminalSensitiveMissile

末敏导弹类，实现了末端敏感引信的导弹功能

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [TerminalSensitiveMissile](#)

Implements

[IMissile](#)

Inherited Members

[BaseMissile.FlightTime](#)
[BaseMissile.FlightDistance](#)
[BaseMissile.EngineBurnTime](#)
[BaseMissile.IsGuidance](#)
[BaseMissile.LostGuidanceTime](#)
[BaseMissile.LastKnownVelocity](#)
[BaseMissile.GuidanceAcceleration](#)
[BaseMissile.ThrustAcceleration](#)
[BaseMissile.MissileProperties](#)
[BaseMissile.UpdateMotionState\(double\)](#)
[BaseMissile.Fire\(\)](#)
[BaseMissile.ShouldSelfDestruct\(\)](#)
[BaseMissile.Explode\(\)](#)
[BaseMissile.Activate\(\)](#)
[BaseMissile.Deactivate\(\)](#)
[BaseMissile.SelfDestruct\(\)](#)
[BaseMissile.UpdateGuidanceStatus\(\)](#)
[BaseMissile.GetRunningState\(\)](#)
[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class TerminalSensitiveMissile : BaseMissile, IMissile
```

Remarks

该类提供了末敏导弹的完整实现：

- 继承自BaseMissile，具备基本的导弹功能
- 实现了末端敏感引信的特殊功能
- 支持子弹药分离和释放
- 具备高度和距离控制能力
- 提供精确的目标打击能力

工作流程：

1. 导弹发射并飞向预定分离点
2. 到达分离高度时释放子弹药
3. 子弹药独立制导飞向目标
4. 母弹完成任务后自动销毁

Constructors

TerminalSensitiveMissile(string, MissileProperties, ISimulationManager)

初始化末敏导弹的新实例

Declaration

```
public TerminalSensitiveMissile(string targetId, MissileProperties properties, ISimulationManager manager)
```

Parameters

Type	Name	Description
string	targetId	目标的唯一标识符
MissileProperties	properties	导弹的配置参数
ISimulationManager	manager	仿真管理器实例

Remarks

构造过程：

1. 初始化基本属性
2. 创建子弹药数组
3. 计算分离点位置
4. 计算最佳发射角度
5. 设置初始速度和方向

Exceptions

Type	Condition
Exception	当目标不存在时抛出
InvalidOperationException	当无法计算发射方向时抛出

Methods

GetStatus()

获取导弹的状态信息字符串

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含导弹状态的详细描述

Overrides

[BaseMissile.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基类状态信息
- 分离点位置
- 母弹特有的状态信息

Update(double)

更新导弹的状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BaseMissile.Update\(double\)](#)

Remarks

更新过程：

1. 检查导弹是否处于活动状态
2. 验证自毁条件（时间、距离、高度）
3. 更新基本运动状态
4. 计算与分离点的距离
5. 记录状态信息
6. 必要时触发分离动作

自毁条件：

- 超出最大飞行时间
- 超出最大飞行距离
- 高度低于地面

Implements

[IMissile](#)

Class InfraredGuidanceCommandEvent

红外指令制导事件，表示发送制导指令

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [InfraredGuidanceCommandEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredGuidanceCommandEvent : SimulationEvent
```

Remarks

用于红外指令制导系统 包含目标和导弹的视线向量信息

Properties

TargetMissileId

获取或设置目标导弹的ID

Declaration

```
public string? TargetMissileId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识接收制导指令的导弹

TrackerToMissileVector

获取或设置跟踪器到导弹的视线向量

Declaration

```
public Vector3D TrackerToMissileVector { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

表示从跟踪器到导弹的方向 单位：米

TrackerToTargetVector

获取或设置跟踪器到目标的视线向量

Declaration

```
public Vector3D TrackerToTargetVector { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

表示从跟踪器到目标的方向 单位：米

Class InfraredDetector

红外探测器类，实现了红外辐射的探测和目标识别功能

Inheritance

↳ [object](#)
↳ [Sensor](#)
↳ InfraredDetector

Implements

[ISensor](#)

Inherited Members

[Sensor.IsActive](#)
[Sensor.Position](#)
[Sensor.Orientation](#)
[Sensor.Activate\(\)](#)
[Sensor.Deactivate\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredDetector : Sensor, ISensor
```

Remarks

该类提供了红外探测器的核心功能：

- 红外辐射强度测量
- 目标存在性判断
- 视场角范围探测
- 实时状态更新 用于末敏子弹的目标探测和跟踪

Constructors

InfraredDetector(TerminalSensitiveSubmunition, double, double)

初始化红外探测器的新实例

Declaration

```
public InfraredDetector(TerminalSensitiveSubmunition submunition, double detectionRange, double fieldOfView)
```

Parameters

Type	Name	Description
TerminalSensitiveSubmunition	<i>submunition</i>	未敏子弹实例
double	<i>detectionRange</i>	探测范围，单位：米
double	<i>fieldOfView</i>	视场角，单位：度

Remarks

- 构造过程：
- 设置探测参数
 - 关联未敏子弹
 - 初始化传感器数据
 - 继承基类位置和姿态

Properties

DetectionRange

获取或设置探测范围，单位：米

Declaration

```
public double DetectionRange { get; set; }
```

Property Value

Type	Description
double	

Remarks

定义了探测器的最大作用距离 影响目标探测的有效范围

FieldOfView

获取或设置视场角，单位：度

Declaration

```
public double FieldOfView { get; set; }
```

Property Value

Type	Description
double	

Remarks

定义了探测器的视场范围 影响目标探测的空间覆盖

Methods

GetSensorData()

获取红外探测器的最新数据

Declaration

```
public override SensorData GetSensorData()
```

Returns

Type	Description
SensorData	包含探测结果的数据对象

Overrides

[Sensor.GetSensorData\(\)](#)

Remarks

返回数据：

- 目标探测状态
- 温度测量结果
- 时间戳信息

Update(double)

更新红外探测器的工作状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒

Overrides

[Sensor.Update\(double\)](#)

Remarks

更新过程：

- 检查激活状态
- 测量辐射强度
- 判断目标存在
- 更新传感器数据

Implements

[ISensor](#)

Namespace ThreatSource.Guidance

Classes

BasicGuidanceSystem

基础制导系统类，实现了制导系统的基本功能框架

InfraredCommandGuidanceSystem

红外指令导引系统类，实现了基于红外跟踪器的指令制导功能

InfraredImagingGuidanceSystem

红外成像引导系统类，实现了基于红外图像的目标探测和跟踪功能

LaserBeamRiderGuidanceSystem

激光驾束制导系统类，实现了基于激光束跟踪的制导功能

LaserSemiActiveGuidanceSystem

激光半主动制导系统类，实现了基于激光照射的目标跟踪和制导功能

MillimeterWaveGuidanceSystem

毫米波导引头系统类，实现了基于毫米波雷达的目标探测和跟踪功能

Interfaces

IGuidanceSystem

制导系统接口，定义了所有制导系统的通用功能

[Show / Hide Table of Contents](#)

Interface ISimulationManager

仿真管理器接口，提供仿真系统的核心功能

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public interface ISimulationManager
```

Remarks

该接口定义了仿真系统的主要功能，包括：

- 事件系统：用于实体间的通信和状态同步
- 实体管理：负责实体的注册、注销和查询
- 第三方集成：支持与其他仿真环境的对接

Methods

GetAllEntities()

获取仿真系统中的所有实体

Declaration

```
IReadOnlyList<object> GetAllEntities()
```

Returns

Type	Description
IReadOnlyList<object>	所有实体的列表

GetEntitiesByType<T>()

获取特定类型的所有实体

Declaration

```
IReadOnlyList<T> GetEntitiesByType<T>() where T : class
```

Returns

Type	Description
IReadOnlyList<T>	指定类型的实体列表

Type Parameters

Name	Description
<i>T</i>	实体类型

GetEntityById(string)

根据ID获取实体

Declaration

```
object? GetEntityById(string id)
```

Parameters

Type	Name	Description
string	<i>id</i>	实体ID

Returns

Type	Description
object	实体对象，如果不存在则返回null

GetSimulationAdapter()

获取当前的仿真环境适配器

Declaration

```
ISimulationAdapter? GetSimulationAdapter()
```

Returns

Type	Description
ISimulationAdapter	当前配置的适配器实例，如果未配置则返回null

PublishEvent<T>(T)

发布事件到仿真系统

Declaration

```
void PublishEvent<T>(T evt)
```

Parameters

Type	Name	Description
<i>T</i>	<i>evt</i>	要发布的事件

Type Parameters

Name	Description
<i>T</i>	事件类型

RegisterEntity(string, object)

注册实体到仿真系统

Declaration

```
bool RegisterEntity(string id, object entity)
```

Parameters

Type	Name	Description
string	<i>id</i>	实体ID
object	<i>entity</i>	实体对象

Returns

Type	Description
bool	注册是否成功

SetSimulationAdapter(ISimulationAdapter)

设置第三方仿真环境适配器

Declaration

```
void SetSimulationAdapter(ISimulationAdapter adapter)
```

Parameters

Type	Name	Description
ISimulationAdapter	<i>adapter</i>	要设置的适配器实例

Remarks

用于配置与外部仿真环境的集成

SubscribeToEvent<T>(Action<T>)

订阅特定类型的事件

Declaration

```
void SubscribeToEvent<T>(Action<T> handler)
```

Parameters

Type	Name	Description
Action<T>	<i>handler</i>	事件处理函数

Type Parameters

Name	Description
<i>T</i>	事件类型

UnregisterEntity(string)

从仿真系统注销实体

Declaration

```
bool UnregisterEntity(string id)
```

Parameters

Type	Name	Description
string	<i>id</i>	要注销的实体ID

Returns

Type	Description
bool	注销是否成功

UnsubscribeFromEvent<T>(Action<T>)

取消订阅特定类型的事件

Declaration

```
void UnsubscribeFromEvent<T>(Action<T> handler)
```

Parameters

Type	Name	Description
Action<T>	handler	要取消的事件处理函数

Type Parameters

Name	Description
T	事件类型

[Show / Hide Table of Contents](#)

Class InfraredWarnerAlarmStopEvent

红外告警器警报停止事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [InfraredWarnerAlarmStopEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class InfraredWarnerAlarmStopEvent : SimulationEvent
```

Properties

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Class SensorData

传感器数据的抽象基类，定义了所有传感器数据的通用属性

Inheritance

↳ [object](#)

↳ [SensorData](#)

↳ [AltimeterSensorData](#)

↳ [InfraredSensorData](#)

↳ [RadiometerSensorData](#)

↳ [RangefinderSensorData](#)

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public abstract class SensorData
```

Remarks

该类作为所有传感器数据类型的基础：

- 提供基本的时间戳信息
- 统一数据接口规范
- 支持数据类型扩展 用于传感器数据的采集和处理

Properties

Timestamp

获取或设置数据采集的时间戳

Declaration

```
public DateTime Timestamp { get; set; }
```

Property Value

Type	Description
DateTime	

Remarks

记录传感器数据的采集时间 用于数据时序分析和同步处理

[Show / Hide Table of Contents](#)

Class InfraredTrackerConfig

红外测角仪配置类

Inheritance

↳ [object](#)
↳ InfraredTrackerConfig

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredTrackerConfig
```

Constructors

InfraredTrackerConfig()

构造函数,设置默认值

Declaration

```
public InfraredTrackerConfig()
```

Properties

AngleMeasurementAccuracy

角度测量精度(弧度)

Declaration

```
public double AngleMeasurementAccuracy { get; set; }
```

Property Value

Type	Description
double	

FieldOfView

视场角(弧度)

Declaration

```
public double FieldOfView { get; set; }
```

Property Value

Type	Description
double	

Id

红外测角仪ID

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

InitialOrientation

初始朝向

Declaration

```
public Orientation InitialOrientation { get; set; }
```

Property Value

Type	Description
Orientation	

InitialPosition

初始位置

Declaration

```
public Vector3D InitialPosition { get; set; }
```

Property Value

Type	Description
Vector3D	

MaxTrackingRange

最大跟踪距离(米)

Declaration

```
public double MaxTrackingRange { get; set; }
```

Property Value

Type	Description
double	

UpdateFrequency

更新频率(赫兹)

Declaration

```
public double UpdateFrequency { get; set; }
```

Property Value

Type	Description
double	

[Show / Hide Table of Contents](#)

Class LaserJammerConfig

激光干扰器配置类

Inheritance

↳ [object](#)
↳ LaserJammerConfig

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserJammerConfig
```

Constructors

LaserJammerConfig()

构造函数,设置默认值

Declaration

```
public LaserJammerConfig()
```

Properties

Id

激光干扰器ID

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

InitialJammingPower

初始干扰功率(瓦特)

Declaration

```
public double InitialJammingPower { get; set; }
```

Property Value

Type	Description
double	

MaxJammingCooldown

最大干扰冷却时间(秒)

Declaration

```
public double MaxJammingCooldown { get; set; }
```

Property Value

Type	Description
double	

MaxJammingPower

最大干扰功率(瓦特)

Declaration

```
public double MaxJammingPower { get; set; }
```

Property Value

Type	Description
double	

PowerIncreaseRate

功率增加速率(瓦特/秒)

Declaration

```
public double PowerIncreaseRate { get; set; }
```

Property Value

Type	Description
double	

Class MillimeterWaveTerminalGuidedMissile

毫米波末制导导弹类，继承自基础导弹类

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [MillimeterWaveTerminalGuidedMissile](#)

Implements

[IMissile](#)

Inherited Members

[BaseMissile.FlightTime](#)
[BaseMissile.FlightDistance](#)
[BaseMissile.EngineBurnTime](#)
[BaseMissile.IsGuidance](#)
[BaseMissile.LostGuidanceTime](#)
[BaseMissile.LastKnownVelocity](#)
[BaseMissile.GuidanceAcceleration](#)
[BaseMissile.ThrustAcceleration](#)
[BaseMissile.MissileProperties](#)
[BaseMissile.UpdateMotionState\(double\)](#)
[BaseMissile.Fire\(\)](#)
[BaseMissile.ShouldSelfDestruct\(\)](#)
[BaseMissile.Activate\(\)](#)
[BaseMissile.Deactivate\(\)](#)
[BaseMissile.SelfDestruct\(\)](#)
[BaseMissile.UpdateGuidanceStatus\(\)](#)
[BaseMissile.GetRunningState\(\)](#)
[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class MillimeterWaveTerminalGuidedMissile : BaseMissile, IMissile
```

Remarks

- 该类提供了毫米波末制导导弹的完整实现：
- 继承自BaseMissile，具备基本的导弹功能
 - 实现了毫米波制导系统
 - 支持多阶段飞行控制
 - 具备全天候目标探测能力
 - 提供精确的末制导打击能力

- 工作流程：
1. 导弹发射并进入发射阶段
 2. 切换到巡航阶段，进行中程飞行
 3. 进入末制导阶段，启动毫米波制导
 4. 接近目标后引爆或达到自毁条件后自毁

Constructors

MillimeterWaveTerminalGuidedMissile(MissileProperties, ISimulationManager)

初始化毫米波末制导导弹的新实例

Declaration

```
public MillimeterWaveTerminalGuidedMissile(MissileProperties properties, ISimulationManager manager)
```

Parameters

Type	Name	Description
MissileProperties	properties	导弹的配置参数
ISimulationManager	manager	仿真管理器实例

Remarks

- 构造过程：
1. 调用基类构造函数
 2. 创建制导系统实例
 3. 设置初始飞行阶段

Methods

Explode()

执行导弹爆炸

Declaration

```
public override void Explode()
```

Overrides

BaseMissile.Explode()

Remarks

- 爆炸过程：
1. 调用基类的爆炸方法
 2. 切换到爆炸阶段

GetStatus()

获取导弹的状态信息字符串

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含导弹状态的详细描述

Overrides

[BaseMissile.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基类状态信息
- 当前飞行阶段
- 制导系统状态
- 目标跟踪信息

Update(double)

更新导弹的状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BaseMissile.Update\(double\)](#)

Remarks

更新过程：

1. 调用基类的更新方法
2. 根据当前阶段选择更新方法
3. 执行相应阶段的更新逻辑
4. 检查是否需要自毁

Implements

[IMissile](#)

Class TerminalSensitiveSubmunition

末敏子弹类，实现了多传感器融合的末端制导功能

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [TerminalSensitiveSubmunition](#)

Implements

[IMissile](#)

Inherited Members

[BaseMissile.FlightTime](#)
[BaseMissile.FlightDistance](#)
[BaseMissile.EngineBurnTime](#)
[BaseMissile.IsGuidance](#)
[BaseMissile.LostGuidanceTime](#)
[BaseMissile.LastKnownVelocity](#)
[BaseMissile.GuidanceAcceleration](#)
[BaseMissile.ThrustAcceleration](#)
[BaseMissile.MissileProperties](#)
[BaseMissile.UpdateMotionState\(double\)](#)
[BaseMissile.Fire\(\)](#)
[BaseMissile.ShouldSelfDestruct\(\)](#)
[BaseMissile.Explode\(\)](#)
[BaseMissile.SelfDestruct\(\)](#)
[BaseMissile.UpdateGuidanceStatus\(\)](#)
[BaseMissile.GetRunningState\(\)](#)
[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class TerminalSensitiveSubmunition : BaseMissile, IMissile
```

Remarks

该类提供了末敏子弹的核心功能：

- 多传感器融合（红外、毫米波、激光）
- 螺旋扫描搜索
- 自主目标识别
- 末端精确制导 通过多种传感器的协同工作实现对目标的精确打击

Constructors

TerminalSensitiveSubmunition(string, MissileProperties, ISimulationManager)

初始化末敏子弹的新实例

Declaration

```
public TerminalSensitiveSubmunition(string targetId, MissileProperties properties, ISimulationManager manager)
```

Parameters

Type	Name	Description
string	targetId	目标ID
MissileProperties	properties	子弹配置参数
ISimulationManager	manager	仿真管理器实例

Remarks

构造过程：

- 初始化基本属性
- 创建传感器系统
- 配置传感器参数
- 设置初始状态

Methods

Activate()

激活子弹

Declaration

```
public override void Activate()
```

Overrides

BaseMissile.Activate()

Remarks

激活过程：

- 调用基类激活方法
- 订阅目标辐射事件
- 准备传感器系统

Deactivate()

停用子弹

Declaration

```
public override void Deactivate()
```

Overrides

[BaseMissile.Deactivate\(\)](#)

Remarks

停用过程：

- 调用基类停用方法
- 停用所有传感器
- 清理制导状态

DetectTarget(double)

检测目标是否在视场内

Declaration

```
public bool DetectTarget(double fieldOfView)
```

Parameters

Type	Name	Description
double	fieldOfView	视场角，单位：度

Returns

Type	Description
bool	如果目标在视场内返回true，否则返回false

Remarks

检测过程：

- 获取目标位置
- 计算目标方向
- 判断是否在视场内

GetStatus()

获取子弹状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含子弹状态的详细描述

Overrides

[BaseMissile.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基本状态信息
- 当前飞行阶段
- 扫描和检测状态
- 传感器工作状态

Update(double)

更新子弹状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	deltaTime	时间步长，单位：秒

Overrides

[BaseMissile.Update\(double\)](#)

Remarks

更新过程：

- 更新所有传感器
- 根据当前阶段更新状态
- 处理阶段转换
- 调用基类更新

Implements

[IMissile](#)

Class LaserBeamRider

激光波束制导器类，用于生成激光波束制导场

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [LaserBeamRider](#)

Implements

[IIndicator](#)

Inherited Members

[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Indicator](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserBeamRider : SimulationElement, IIndicator
```

Remarks

该类提供了激光波束制导系统的核心功能：

- 生成和维护激光波束制导场
- 控制激光波束的开启和关闭
- 实时更新波束位置和方向
- 发布波束状态事件

Constructors

LaserBeamRider(string, string, string, double, LaserBeamRiderConfig, ISimulationManager)

初始化激光波束制导器的新实例

Declaration

```
public LaserBeamRider(string id, string missileId, string targetId, double speed, LaserBeamRiderConfig config, ISimulationManager simulationManager)
```

Parameters

Type	Name	Description
string	id	制导器的唯一标识符
string	missileId	导弹ID
string	targetId	目标ID
double	speed	初始速度
LaserBeamRiderConfig	config	配置参数
ISimulationManager	simulationManager	仿真管理器实例

Remarks

- 构造过程：
- 初始化基本属性
 - 设置激光参数
 - 配置制导范围
 - 建立目标关联

Properties

BeamDivergence

获取或设置激光发散角

Declaration

```
public double BeamDivergence { get; }
```

Property Value

Type	Description
double	

Remarks

单位：毫弧度 影响激光波束的扩散特性和制导精度

ControlFieldDiameter

获取或设置控制场直径

Declaration

```
public double ControlFieldDiameter { get; }
```

Property Value

Type	Description
double	

Remarks

单位：米 定义了激光波束制导场的有效范围

IsBeamOn

获取激光束是否开启

Declaration

```
public bool IsBeamOn { get; }
```

Property Value

Type	Description
bool	

Remarks

true表示激光波束正在发射 false表示激光波束已关闭

LaserDirection

获取或设置激光方向

Declaration

```
public Vector3D LaserDirection { get; }
```

Property Value

Type	Description
Vector3D	

Remarks

使用三维向量表示激光波束的指向 向量的模为1（归一化）

LaserPower

获取或设置激光功率

Declaration

```
public double LaserPower { get; }
```

Property Value

Type	Description
double	

Remarks

单位： 瓦特 影响激光波束的有效作用距离和制导精度

MaxGuidanceDistance

获取或设置最大导引距离

Declaration

```
public double MaxGuidanceDistance { get; }
```

Property Value

Type	Description
double	

Remarks

单位：米 超出此距离的导弹将无法接收到有效的制导信号

MissileId

获取当前制导的导弹ID

Declaration

```
public string? MissileId { get; }
```

Property Value

Type	Description
string	

Remarks

记录正在接受制导的导弹标识符 如果为null表示当前没有制导目标

TargetId

获取目标ID

Declaration

```
public string? TargetId { get; }
```

Property Value

Type	Description
string	

Remarks

记录当前跟踪的目标标识符 如果为null表示当前没有跟踪目标

Methods

Activate()

激活激光驾束仪

Declaration

```
public override void Activate()
```

Overrides

[SimulationElement.Activate\(\)](#)

Remarks

激活过程：

- 设置激活状态
- 开始激光照射
- 发布激活事件

Deactivate()

停用激光驾束仪

Declaration

```
public override void Deactivate()
```

Overrides

[SimulationElement.Deactivate\(\)](#)

Remarks

停用过程：

- 停止激光照射
- 清理状态
- 发布停用事件

GetRunningState()

获取激光驾束仪运行状态

Declaration

```
public IndicatorRunningState GetRunningState()
```

Returns

Type	Description
IndicatorRunningState	包含完整状态信息的结构体

Remarks

返回信息包括：

- 目标和导弹ID
- 设备类型和工作状态
- 位置和朝向信息
- 激活状态

GetStatus()

获取激光驾束仪状态字符串

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含设备状态的详细描述

Overrides

[SimulationElement.GetStatus\(\)](#)

Remarks

返回信息包括：

- 设备ID和位置
- 激光参数（功率、发散角等）
- 工作状态（激活、照射等）
- 制导范围参数

StartBeamIllumination()

开启激光束照射

Declaration

```
public void StartBeamIllumination()
```

Remarks

开启过程：

- 计算初始指向
- 设置照射状态
- 发布开始事件

StopBeamIllumination()

停止激光束照射

Declaration

```
public void StopBeamIllumination()
```

Remarks

停止过程：

- 关闭激光输出
- 清除指向信息
- 发布停止事件

Update(double)

更新激光驾束仪状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[SimulationElement.Update\(double\)](#)

Remarks

更新过程：

- 检查激活状态
- 更新激光指向
- 发布状态更新事件

Implements

[IIndicator](#)

Class InfraredDetectionEvent

红外探测事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [InfraredDetectionEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class InfraredDetectionEvent : SimulationEvent
```

Properties

Intensity

红外辐射强度

Declaration

```
public double Intensity { get; set; }
```

Property Value

Type	Description
double	

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Interface ITarget

定义目标对象的基本接口，提供目标的基本特征和属性

Namespace: [ThreatSource.Target](#)

Assembly: ThreatSource.dll

Syntax

```
public interface ITarget
```

Remarks

该接口定义了目标的关键特征：

- 目标类型标识
- 雷达散射截面积
- 红外辐射特征 用于在仿真系统中表示和识别不同类型的目标

Properties

InfraredRadiationIntensity

获取目标的红外辐射强度

Declaration

```
double InfraredRadiationIntensity { get; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特/平方米 表示目标的热辐射特征 影响红外制导系统的探测和跟踪能力

RadarCrossSection

获取目标的雷达散射截面积

Declaration

```
double RadarCrossSection { get; }
```

Property Value

Type	Description
double	

Remarks

单位：平方米 表示目标对雷达波的反射能力 影响目标在雷达系统中的探测距离和识别概率

Type

获取目标的类型标识

Declaration

```
string Type { get; }
```

Property Value

Type	Description
string	

Remarks

用于区分不同种类的目标，如坦克、装甲车等 在仿真系统中用于目标识别和分类

Class InfraredCommandGuidedMissile

红外指令制导导弹类，实现了红外指令制导的导弹功能

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [InfraredCommandGuidedMissile](#)

Implements

[IMissile](#)

Inherited Members

[BaseMissile.FlightTime](#)
[BaseMissile.FlightDistance](#)
[BaseMissile.EngineBurnTime](#)
[BaseMissile.IsGuidance](#)
[BaseMissile.LostGuidanceTime](#)
[BaseMissile.LastKnownVelocity](#)
[BaseMissile.GuidanceAcceleration](#)
[BaseMissile.ThrustAcceleration](#)
[BaseMissile.MissileProperties](#)
[BaseMissile.Update\(double\)](#)
[BaseMissile.Fire\(\)](#)
[BaseMissile.ShouldSelfDestruct\(\)](#)
[BaseMissile.Explode\(\)](#)
[BaseMissile.SelfDestruct\(\)](#)
[BaseMissile.UpdateGuidanceStatus\(\)](#)
[BaseMissile.GetRunningState\(\)](#)
[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class InfraredCommandGuidedMissile : BaseMissile, IMissile
```

Remarks

该类提供了红外指令制导导弹的核心功能：

- 红外热源管理（点亮和熄灭）
- 指令制导控制
- 飞行阶段管理
- 状态监控和事件处理 通过红外测角仪和指令制导系统实现对导弹的制导控制

Constructors

InfraredCommandGuidedMissile(MissileProperties, ISimulationManager)

初始化红外指令制导导弹的新实例

Declaration

```
public InfraredCommandGuidedMissile(MissileProperties config, ISimulationManager manager)
```

Parameters

Type	Name	Description
MissileProperties	<i>config</i>	导弹配置参数
ISimulationManager	<i>manager</i>	仿真管理器实例

Remarks

构造过程：

- 初始化基本属性
- 设置红外辐射功率
- 创建指令导引系统
- 配置PID参数

Properties

RadiationPower

获取或设置红外热源辐射功率

Declaration

```
public double RadiationPower { get; set; }
```

Property Value

Type	Description
double	辐射功率，单位：瓦特

Remarks

影响导弹的红外信号强度 用于红外测角仪的跟踪和制导

Methods

Activate()

激活导弹

Declaration

```
public override void Activate()
```

Overrides

[BaseMissile.Activate\(\)](#)

Remarks

激活过程：

- 调用基类激活方法
- 订阅制导指令事件
- 准备接收制导指令

Deactivate()

停用导弹

Declaration

```
public override void Deactivate()
```

Overrides

[BaseMissile.Deactivate\(\)](#)

Remarks

停用过程：

- 调用基类停用方法
- 熄灭红外热源
- 取消订阅制导事件

GetStatus()

获取导弹状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含导弹状态的详细描述

Overrides

[BaseMissile.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基本状态信息
- 当前飞行阶段
- 制导系统状态

LightInfraredSource()

点亮红外热源

Declaration

```
public void LightInfraredSource()
```

Remarks

发布红外热源点亮事件 包含当前辐射功率信息

LightOffInfraredSource()

熄灭红外热源

Declaration

```
public void LightOffInfraredSource()
```

Remarks

发布红外热源熄灭事件 停止红外信号发射

UpdateMotionState(double)

更新导弹运动状态

Declaration

```
protected override void UpdateMotionState(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BaseMissile.UpdateMotionState\(double\)](#)

Remarks

更新过程：

- 点亮红外热源
- 根据当前阶段更新状态
- 调用基类的运动更新

Implements

[IMissile](#)

- [ThreatSource.Guidance](#)
 - [BasicGuidanceSystem](#)
 - [IGuidanceSystem](#)
 - [InfraredCommandGuidanceSystem](#)
 - [InfraredImagingGuidanceSystem](#)
 - [LaserBeamRiderGuidanceSystem](#)
 - [LaserSemiActiveGuidanceSystem](#)
 - [MillimeterWaveGuidanceSystem](#)
- [ThreatSource.Indicator](#)
 - [Indicator](#)
 - [IndicatorRunningState](#)
 - [IndicatorState](#)
 - [IndicatorType](#)
 - [InfraredTracker](#)
 - [LaserBeamRider](#)
 - [LaserDesignator](#)
- [ThreatSource.Missile](#)
 - [BaseMissile](#)
 - [IMissile](#)
 - [InfraredCommandGuidedMissile](#)
 - [InfraredImagingTerminalGuidedMissile](#)
 - [LaserBeamRiderMissile](#)
 - [LaserSemiActiveGuidedMissile](#)
 - [MillimeterWaveTerminalGuidedMissile](#)
 - [MissileProperties](#)
 - [MissileRunningState](#)
 - [MissileType](#)
 - [TerminalSensitiveMissile](#)
 - [TerminalSensitiveSubmunition](#)
- [ThreatSource.Sensor](#)
 - [AltimeterSensorData](#)
 - [ISensor](#)
 - [InfraredDetector](#)
 - [InfraredSensorData](#)
 - [LaserRangefinder](#)
 - [MillimeterWaveAltimeter](#)
 - [MillimeterWaveRadiometer](#)
 - [RadiometerSensorData](#)
 - [RangefinderSensorData](#)
 - [Sensor](#)
 - [SensorData](#)
- [ThreatSource.Simulation](#)
 - [EntityActivatedEvent](#)
 - [EntityDeactivatedEvent](#)
 - [EntityDestroyedEvent](#)
 - [ISimulationAdapter](#)
 - [ISimulationManager](#)
 - [InfraredDetectionEvent](#)
 - [InfraredGuidanceCommandEvent](#)
 - [InfraredGuidanceMissileLightEvent](#)
 - [InfraredGuidanceMissileLightOffEvent](#)
 - [InfraredJammingEvent](#)
 - [InfraredTrackerConfig](#)
 - [InfraredWarnerAlarmEvent](#)
 - [InfraredWarnerAlarmStopEvent](#)
 - [InfraredWarnerConfig](#)
 - [LaserBeamRiderConfig](#)
 - [LaserBeamStartEvent](#)
 - [LaserBeamStopEvent](#)
 - [LaserBeamUpdateEvent](#)
 - [LaserDesignatorConfig](#)
 - [LaserIlluminationStartEvent](#)
 - [LaserIlluminationStopEvent](#)
 - [LaserIlluminationUpdateEvent](#)
 - [LaserJammerConfig](#)
 - [LaserJammingEvent](#)

- [LaserWarnerAlarmEvent](#)
- [LaserWarnerAlarmStopEvent](#)
- [LaserWarnerConfig](#)
- [MillimeterWaveDetectionEvent](#)
- [MillimeterWaveJammerConfig](#)
- [MillimeterWaveJammingEvent](#)
- [MillimeterWaveWarnerAlarmEvent](#)
- [MillimeterWaveWarnerAlarmStopEvent](#)
- [MillimeterWaveWarnerConfig](#)
- [MissileFireEvent](#)
- [SimulationElement](#)
- [SimulationEvent](#)
- [SimulationManager](#)
- [TankRadiationEvent](#)
- [TargetDestroyedEvent](#)
- [TargetHitEvent](#)
- [UltravioletDetectionEvent](#)
- [UltravioletWarnerAlarmEvent](#)
- [UltravioletWarnerAlarmStopEvent](#)
- [UltravioletWarnerConfig](#)
- [ThreatSource.Simulation.Testing](#)
 - [TestSimulationAdapter](#)
- [ThreatSource.Target](#)
 - [ITarget](#)
 - [Tank](#)
- [ThreatSource.Utils](#)
 - [MotionAlgorithm](#)
 - [Orientation](#)
 - [Vector2D](#)
 - [Vector3D](#)

Class LaserDesignator

激光指示器类，实现了对目标的激光照射和制导功能

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [LaserDesignator](#)

Inherited Members

[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Indicator](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserDesignator : SimulationElement
```

Remarks

该类提供了激光指示器的核心功能：

- 激光目标照射
- 抗干扰处理
- 事件管理
- 状态监控 通过激光照射目标实现半主动激光制导

Constructors

LaserDesignator(string, string, string, double, LaserDesignatorConfig, ISimulationManager)

初始化激光指示器的新实例

Declaration


```
public LaserDesignator(string id, string targetId, string missileId, double speed, LaserDesignatorConfig config, ISimulationManager simulationManager)
```

Parameters

Type	Name	Description
string	<i>id</i>	激光指示器ID
string	<i>targetId</i>	目标ID
string	<i>missileId</i>	导弹ID
double	<i>speed</i>	移动速度
LaserDesignatorConfig	<i>config</i>	配置参数
ISimulationManager	<i>simulationManager</i>	仿真管理器实例

Remarks

构造过程：

- 初始化基本属性
- 设置目标和导弹关联
- 配置激光参数
- 设置初始状态

Properties

IsIlluminationOn

获取或设置是否正在照射状态

Declaration

```
public bool IsIlluminationOn { get; }
```

Property Value

Type	Description
bool	

Remarks

true表示正在执行激光照射 false表示未在照射状态 控制激光发射状态

IsJammed

获取或设置是否被干扰状态

Declaration

```
public bool IsJammed { get; }
```

Property Value

Type	Description
bool	

Remarks

true表示当前受到有效干扰 false表示工作正常 影响激光照射的有效性

JammingThreshold

获取或设置干扰阈值，单位：分贝

Declaration

```
public double JammingThreshold { get; }
```

Property Value

Type	Description
double	

Remarks

定义了激光指示器的抗干扰能力 当干扰信号强度超过此阈值时触发干扰效果

LaserDivergenceAngle

获取或设置激光发散角，单位：弧度

Declaration

```
public double LaserDivergenceAngle { get; }
```

Property Value

Type	Description
double	

Remarks

定义了激光束的扩展角度 影响照射面积和能量密度

LaserPower

获取或设置激光功率，单位：瓦特

Declaration

```
public double LaserPower { get; }
```

Property Value

Type	Description
double	

Remarks

定义了激光照射的能量强度 影响照射距离和制导效果

MissileId

获取或设置导弹ID

Declaration

```
public string? MissileId { get; }
```

Property Value

Type	Description
string	

Remarks

记录当前关联的导弹标识符 用于导弹制导控制

TargetId

获取或设置目标ID

Declaration

```
public string? TargetId { get; }
```

Property Value

Type	Description
string	

Remarks

记录当前照射的目标标识符 用于目标识别和状态更新

Methods

Activate()

激活激光指示器

Declaration

```
public override void Activate()
```

Overrides

[SimulationElement.Activate\(\)](#)

Remarks

激活过程：

- 设置激活状态
- 清除干扰状态
- 开始激光照射
- 订阅相关事件
- 调用基类激活

Deactivate()

停用激光指示器

Declaration

```
public override void Deactivate()
```

Overrides

[SimulationElement.Deactivate\(\)](#)

Remarks

停用过程：

- 清除激活状态
- 停止激光照射
- 取消事件订阅
- 清理工作状态
- 调用基类停用

GetRunningState()

获取激光指示器运行状态

Declaration

```
public IndicatorRunningState GetRunningState()
```

Returns

Type	Description
IndicatorRunningState	包含指示器完整状态信息的结构体

Remarks

返回信息包括：

- 目标和导弹关联状态
- 工作类型和状态
- 位置和姿态信息
- 激活状态

GetStatus()

获取激光指示器状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含指示器状态的详细描述

Overrides

[SimulationElement.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基本标识信息
- 位置信息
- 目标关联状态
- 工作状态
- 干扰状态
- 激光参数

Update(double)

更新激光指示器状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[SimulationElement.Update\(double\)](#)

Remarks

更新过程：

- 检查激活状态
- 检查干扰状态

- 更新照射状态
- 发布状态事件

[Show / Hide Table of Contents](#)

Class MillimeterWaveJammerConfig

毫米波干扰器配置类

Inheritance

↳ [object](#)

↳ MillimeterWaveJammerConfig

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveJammerConfig
```

Constructors

MillimeterWaveJammerConfig()

构造函数

Declaration

```
public MillimeterWaveJammerConfig()
```

Properties

Id

干扰器ID

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

InitialJammingPower

初始干扰功率(瓦特)

Declaration

```
public double InitialJammingPower { get; set; }
```

Property Value

Type	Description
double	

MaxJammingCooldown

最大冷却时间(秒)

Declaration

```
public double MaxJammingCooldown { get; set; }
```

Property Value

Type	Description
double	

MaxJammingPower

最大干扰功率(瓦特)

Declaration

```
public double MaxJammingPower { get; set; }
```

Property Value

Type	Description
double	

PowerIncreaseRate

功率增长率(瓦特/秒)

Declaration

```
public double PowerIncreaseRate { get; set; }
```

Property Value

Type	Description
double	

[Show / Hide Table of Contents](#)

Class InfraredGuidanceMissileLightOffEvent

红外指令制导导弹熄灭红外热源事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ InfraredGuidanceMissileLightOffEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredGuidanceMissileLightOffEvent : SimulationEvent
```

Remarks

用于模拟导弹尾部红外热源的关闭 触发时机：导弹发动机关闭时

Class BaseMissile

导弹基类，实现了导弹的基本功能和状态管理

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [InfraredCommandGuidedMissile](#)
↳ [InfraredImagingTerminalGuidedMissile](#)
↳ [LaserBeamRiderMissile](#)
↳ [LaserSemiActiveGuidedMissile](#)
↳ [MillimeterWaveTerminalGuidedMissile](#)
↳ [TerminalSensitiveMissile](#)
↳ [TerminalSensitiveSubmunition](#)

Implements

[IMissile](#)

Inherited Members

[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: ThreatSource.dll

Syntax

```
public class BaseMissile : SimulationElement, IMissile
```

Remarks

该类提供了导弹的核心功能：

- 运动状态计算和更新
- 制导系统管理
- 发动机控制
- 自毁和爆炸处理
- 状态监控和事件处理 所有具体的导弹类型都继承自此基类

Constructors

BaseMissile(MissileProperties, ISimulationManager)

初始化导弹基类的新实例

Declaration

```
public BaseMissile(MissileProperties properties, ISimulationManager manager)
```

Parameters

Type	Name	Description
MissileProperties	<i>properties</i>	导弹的配置参数
ISimulationManager	<i>manager</i>	仿真管理器实例

Remarks

构造过程：

- 初始化基本属性（位置、朝向、速度）
- 设置初始状态（未激活、未制导）
- 清零计时器和计数器
- 初始化加速度向量

Fields

LastKnownVelocity

获取或设置导弹的最后已知速度向量

Declaration

```
protected Vector3D LastKnownVelocity
```

Field Value

Type	Description
Vector3D	三维速度向量

Remarks

用于在失去制导时保持导弹的运动状态 作为弹道计算的参考数据

MissileProperties

获取导弹的固定配置参数

Declaration

```
public readonly MissileProperties MissileProperties
```

Field Value

Type	Description
MissileProperties	导弹属性配置对象

Remarks

包含导弹的所有基本属性和性能限制 在导弹创建时设置，运行期间保持不变

Properties

EngineBurnTime

获取发动机的当前燃烧时间

Declaration

```
public double EngineBurnTime { get; protected set; }
```

Property Value

Type	Description
double	发动机燃烧时间，单位：秒

Remarks

发动机工作的累计时间 用于控制推力变化和燃料消耗

FlightDistance

获取导弹的当前飞行距离

Declaration

```
public double FlightDistance { get; protected set; }
```

Property Value

Type	Description
double	飞行距离，单位：米

Remarks

从发射点到当前位置的累计飞行距离 用于判断是否超出最大射程

FlightTime

获取导弹的当前飞行时间

Declaration

```
public double FlightTime { get; protected set; }
```

Property Value

Type	Description
double	飞行时间，单位：秒

Remarks

从发射时刻开始计时 用于控制导弹的生命周期

GuidanceAcceleration

获取或设置导弹的制导加速度

Declaration

```
protected Vector3D GuidanceAcceleration { get; set; }
```

Property Value

Type	Description
Vector3D	三维加速度向量，单位：米/秒 ²

Remarks

由制导系统计算得出的期望加速度 用于修正导弹的飞行路径

IsGuidance

获取导弹是否处于制导状态

Declaration

```
public bool IsGuidance { get; protected set; }
```

Property Value

Type	Description
bool	true表示导弹当前在制导，false表示导弹处于非制导状态

Remarks

影响导弹的运动学计算方法 制导状态下使用制导律计算加速度 非制导状态下使用弹道方程计算运动

LostGuidanceTime

获取或设置导弹失去制导的持续时间

Declaration

```
protected double LostGuidanceTime { get; set; }
```

Property Value

Type	Description
double	失去制导的时间，单位：秒

Remarks

用于判断是否需要触发自毁 超过阈值时可能导致导弹自毁

ThrustAcceleration

获取或设置导弹的推力加速度

Declaration

```
protected Vector3D ThrustAcceleration { get; set; }
```

Property Value

Type	Description
Vector3D	三维加速度向量，单位：米/秒 ²

Remarks

由发动机产生的推进加速度 影响导弹的速度变化

Methods

Activate()

激活仿真元素，使其参与仿真

Declaration

```
public override void Activate()
```

Overrides

[SimulationElement.Activate\(\)](#)

Remarks

激活操作会：

- 将IsActive设置为true
- 发布实体激活事件
- 允许实体参与仿真计算

Deactivate()

停用仿真元素，将其从仿真中移除

Declaration

```
public override void Deactivate()
```

Overrides

[SimulationElement.Deactivate\(\)](#)

Remarks

停用操作会：

- 将IsActive设置为false
- 发布实体停用事件
- 停止实体参与仿真计算

Explode()

导弹爆炸

Declaration

```
public virtual void Explode()
```

Remarks

爆炸过程：

- 停止导弹运动
- 触发爆炸效果
- 发布爆炸事件
- 结束导弹任务

Fire()

发射导弹

Declaration

```
public virtual void Fire()
```

Remarks

发射过程：

- 激活导弹
- 开始计时和计数

- 启动发动机
- 初始化运动状态

GetRunningState()

获取导弹的完整运行状态

Declaration

```
public MissileRunningState GetRunningState()
```

Returns

Type	Description
MissileRunningState	包含导弹所有状态信息的结构体

Remarks

返回的状态包括：

- 基本信息（ID、类型）
- 运动状态（位置、速度、朝向）
- 飞行数据（时间、距离）
- 工作状态（制导、发动机）

GetStatus()

获取导弹的状态信息字符串

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含导弹状态的详细描述

Overrides

[SimulationElement.GetStatus\(\)](#)

Remarks

返回信息包括：

- 导弹ID和类型
- 位置和速度信息
- 飞行时间和距离
- 制导和发动机状态

SelfDestruct()

导弹自毁

Declaration

```
public void SelfDestruct()
```

Remarks

自毁过程：

- 记录自毁原因
- 停止导弹运动
- 触发自毁效果
- 结束导弹任务

ShouldSelfDestruct()

检查是否应该自毁

Declaration

```
protected bool ShouldSelfDestruct()
```

Returns

Type	Description
bool	true表示需要自毁，false表示可以继续飞行

Remarks

自毁条件：

- 超出最大飞行时间
- 超出最大飞行距离
- 高度低于安全阈值
- 失去制导时间过长

Update(double)

更新导弹的状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[SimulationElement.Update\(double\)](#)

Remarks

更新过程：

- 检查导弹是否处于活动状态
- 更新导弹的运动状态
- 更新计时器和计数器

UpdateGuidanceStatus()

更新导弹的制导状态

Declaration

```
protected virtual void UpdateGuidanceStatus()
```

Remarks

基类中的默认实现为空 具体的导弹类型需要重写此方法 实现各自的制导逻辑

UpdateMotionState(double)

Declaration

```
protected virtual void UpdateMotionState(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	

Implements

[IMissile](#)

[Show / Hide Table of Contents](#)

Class MillimeterWaveDetectionEvent

毫米波探测事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [MillimeterWaveDetectionEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveDetectionEvent : SimulationEvent
```

Properties

Frequency

毫米波频率(GHz)

Declaration

```
public double Frequency { get; set; }
```

Property Value

Type	Description
double	

Intensity

毫米波辐射强度

Declaration

```
public double Intensity { get; set; }
```

Property Value

Type	Description
double	

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Class MillimeterWaveWarnerAlarmEvent

毫米波告警器警报事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [MillimeterWaveWarnerAlarmEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class MillimeterWaveWarnerAlarmEvent : SimulationEvent
```

Properties

DetectedFrequency

探测到的频率(GHz)

Declaration

```
public double DetectedFrequency { get; set; }
```

Property Value

Type	Description
double	

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

威胁源仿真库文档

[Back to top](#)

Namespace ThreatSource.Simulation

Classes

EntityActivatedEvent

实体激活事件，表示仿真实体被激活

EntityDeactivatedEvent

实体停用事件，表示仿真实体被停用

EntityDestroyedEvent

实体销毁事件，表示仿真实体被销毁

InfraredDetectionEvent

红外探测事件

InfraredGuidanceCommandEvent

红外指令制导事件，表示发送制导指令

InfraredGuidanceMissileLightEvent

红外指令制导导弹点亮红外热源事件

InfraredGuidanceMissileLightOffEvent

红外指令制导导弹熄灭红外热源事件

InfraredJammingEvent

红外干扰事件

InfraredTrackerConfig

红外测角仪配置类

InfraredWarnerAlarmEvent

红外告警器警报事件

InfraredWarnerAlarmStopEvent

红外告警器警报停止事件

InfraredWarnerConfig

红外告警器配置类，用于设置红外探测和告警系统的参数

LaserBeamRiderConfig

激光波束制导仪配置类，用于设置激光波束制导系统的参数

LaserBeamStartEvent

激光波束开始事件，表示激光波束制导开始

LaserBeamStopEvent

激光波束停止事件，表示激光波束制导结束

LaserBeamUpdateEvent

激光波束更新事件，表示激光波束位置更新

LaserDesignatorConfig

激光指示器配置类，用于设置激光半主动导引系统中的激光指示器参数

LaserIlluminationStartEvent

激光照射开始事件，表示激光定位器开始照射目标

LaserIlluminationStopEvent

激光照射停止事件，表示激光定位器停止照射目标

LaserIlluminationUpdateEvent

激光照射更新事件，表示激光照射状态的更新

LaserJammerConfig

激光干扰器配置类

LaserJammingEvent

激光干扰事件，表示对激光制导系统的干扰

LaserWarnerAlarmEvent

激光告警器警报事件，表示检测到激光照射

LaserWarnerAlarmStopEvent

激光告警器警报停止事件，表示激光照射结束

LaserWarnerConfig

激光告警器配置类，用于设置激光探测和告警系统的参数

MillimeterWaveDetectionEvent

毫米波探测事件

MillimeterWaveJammerConfig

毫米波干扰器配置类

MillimeterWaveJammingEvent

毫米波干扰事件，表示对毫米波雷达的干扰

MillimeterWaveWarnerAlarmEvent

毫米波告警器警报事件

MillimeterWaveWarnerAlarmStopEvent

毫米波告警器警报停止事件

MillimeterWaveWarnerConfig

毫米波告警器配置类，用于设置毫米波探测和告警系统的参数

MissileFireEvent

导弹发射事件，表示导弹被发射的状态变化

SimulationElement

仿真元素的抽象基类，所有仿真中的实体都继承自此类

SimulationEvent

仿真事件的基类，定义所有仿真事件的共同属性

SimulationManager

仿真管理器的默认实现，提供仿真系统的核心功能

TankRadiationEvent

坦克辐射事件，表示坦克的多波段辐射特性

TargetDestroyedEvent

目标被摧毁事件

TargetHitEvent

目标被击中事件

UltravioletDetectionEvent

紫外探测事件

UltravioletWarnerAlarmEvent

紫外告警器警报事件

UltravioletWarnerAlarmStopEvent

紫外告警器警报停止事件

UltravioletWarnerConfig

紫外告警器配置类，用于设置紫外探测和告警系统的参数

Interfaces

ISimulationAdapter

第三方仿真环境适配器接口

ISimulationManager

仿真管理器接口，提供仿真系统的核心功能

[Show / Hide Table of Contents](#)

Namespace ThreatSource.Target

Classes

Tank

坦克类，实现了目标接口的具体坦克目标

Interfaces

ITarget

定义目标对象的基本接口，提供目标的基本特征和属性

Class InfraredTracker

红外测角仪类，实现了红外制导导弹的跟踪和制导功能

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [InfraredTracker](#)

Implements

[IIndicator](#)

Inherited Members

[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Indicator](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredTracker : SimulationElement, IIndicator
```

Remarks

该类提供了红外测角仪的核心功能：

- 红外目标跟踪
- 角度测量和计算
- 制导指令生成
- 事件处理和状态管理 通过测量目标的红外辐射实现对导弹的制导控制

Constructors

InfraredTracker(string, string, InfraredTrackerConfig, ISimulationManager)

初始化红外测角仪的新实例

Declaration

```
public InfraredTracker(string id, string targetId, InfraredTrackerConfig config, ISimulationManager manager)
```

Parameters

Type	Name	Description
string	id	测角仪ID
string	targetId	目标ID
InfraredTrackerConfig	config	配置参数
ISimulationManager	manager	仿真管理器实例

Remarks

构造过程：

- 初始化基本属性
- 设置目标关联
- 配置工作参数
- 设置初始状态

Properties

IsTracking

获取或设置红外测角仪是否正在跟踪目标

Declaration

```
public bool IsTracking { get; }
```

Property Value

Type	Description
bool	

Remarks

true表示正在跟踪目标 false表示未在跟踪状态 用于控制测角仪的工作状态

MissileId

获取或设置当前跟踪的导弹ID

Declaration

```
public string? MissileId { get; }
```

Property Value

Type	Description
string	

Remarks

记录当前正在跟踪的导弹标识符 用于导弹识别和制导控制

TargetId

获取或设置目标ID

Declaration

```
public string? TargetId { get; }
```

Property Value

Type	Description
string	

Remarks

记录当前跟踪的目标标识符 用于目标识别和状态更新

Methods

Activate()

激活红外测角仪

Declaration

```
public override void Activate()
```

Overrides

[SimulationElement.Activate\(\)](#)

Remarks

激活过程：

- 调用基类激活方法
- 订阅红外事件
- 准备开始工作

Deactivate()

停用红外测角仪

Declaration

```
public override void Deactivate()
```

Overrides

[SimulationElement.Deactivate\(\)](#)

Remarks

停用过程：

- 调用基类停用方法
- 取消事件订阅
- 清理工作状态

GetRunningState()

获取红外测角仪运行状态

Declaration

```
public IndicatorRunningState GetRunningState()
```

Returns

Type	Description
------	-------------

Type	Description
IndicatorRunningState	包含测角仪完整状态信息的结构体

Remarks

返回信息包括：

- 目标和导弹关联状态
- 工作类型和状态
- 位置和姿态信息
- 激活状态

GetStatus()

获取红外测角仪状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含测角仪状态的详细描述

Overrides

[SimulationElement.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基本标识信息
- 位置信息
- 跟踪状态
- 目标关联状态
- 工作状态

StopTracking()

停止跟踪目标

Declaration

```
public void StopTracking()
```

Remarks

停止过程：

- 清除导弹关联
- 清除目标关联
- 重置跟踪状态

Update(double)

更新红外测角仪状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
------	------	-------------

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[SimulationElement.Update\(double\)](#)

Remarks

更新过程：

- 检查激活状态
- 更新跟踪状态
- 处理目标跟踪

Implements

[IIndicator](#)

[Show / Hide Table of Contents](#)

Class AltimeterSensorData

测高仪传感器数据类，包含高度测量的具体数据

Inheritance

↳ [object](#)
↳ [SensorData](#)
↳ [AltimeterSensorData](#)

Inherited Members

[SensorData.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class AltimeterSensorData : SensorData
```

Remarks

该类封装了毫米波测高仪的测量结果：

- 对地高度信息 用于飞行器高度控制和地形跟随

Properties

Altitude

获取或设置测量的高度，单位：米

Declaration

```
public double Altitude { get; set; }
```

Property Value

Type	Description
double	

Remarks

Class MillimeterWaveAltimeter

毫米波测高雷达类，实现了高度测量和数据采集功能

Inheritance

↳ [object](#)
↳ [Sensor](#)
↳ [MillimeterWaveAltimeter](#)

Implements

[ISensor](#)

Inherited Members

[Sensor.IsActive](#)
[Sensor.Position](#)
[Sensor.Orientation](#)
[Sensor.Activate\(\)](#)
[Sensor.Deactivate\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveAltimeter : Sensor, ISensor
```

Remarks

该类提供了毫米波测高雷达的核心功能：

- 实时高度测量
- 精度控制
- 干扰监测
- 数据采集 用于末敏子弹的高度测量和地形跟踪

Constructors

MillimeterWaveAltimeter(TerminalSensitiveSubmunition, double, double)

初始化毫米波测高雷达的新实例

Declaration


```
public MillimeterWaveAltimeter(TerminalSensitiveSubmunition submunition, double maxAltitude, double accuracy)
```

Parameters

Type	Name	Description
TerminalSensitiveSubmunition	<i>submunition</i>	末敏子弹实例
double	<i>maxAltitude</i>	最大测量高度，单位：米
double	<i>accuracy</i>	测量精度，单位：米

Remarks

构造过程：

- 设置量程参数
- 初始化高度记录
- 继承基类位置和姿态

Properties

Accuracy

获取或设置测量精度，单位：米

Declaration

```
public double Accuracy { get; set; }
```

Property Value

Type	Description
double	

Remarks

定义了高度测量的误差范围 实际测量值在真实高度±精度范围内

MaxAltitude

获取或设置最大测量高度，单位：米

Declaration

```
public double MaxAltitude { get; set; }
```

Property Value

Type	Description
double	

Remarks

定义了测高雷达的量程上限 超出此高度的测量可能不准确

Methods

GetSensorData()

获取毫米波测高雷达的最新数据

Declaration

```
public override SensorData GetSensorData()
```

Returns

Type	Description
SensorData	包含高度测量结果的数据对象

Overrides

[Sensor.GetSensorData\(\)](#)

Remarks

返回数据：

- 当前高度值
- 测量时间戳
- 数据可靠性

OnMillimeterWaveJamming(object, EventArgs)

处理毫米波干扰事件

Declaration

```
public void OnMillimeterWaveJamming(object sender, EventArgs e)
```

Parameters

Type	Name	Description
object	<i>sender</i>	事件源对象
EventArgs	<i>e</i>	事件参数

Remarks

当检测到毫米波干扰时：

- 输出警告信息
- 可能影响测量精度
- 需要采取抗干扰措施

Update(double)

更新毫米波测高雷达的工作状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒

Overrides

[Sensor.Update\(double\)](#)

Remarks

更新过程：

- 获取当前位置高度
- 添加测量精度误差
- 更新高度记录

Implements

ISensor

威胁源仿真库文档

[Back to top](#)

Class TargetHitEvent

目标被击中事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [TargetHitEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class TargetHitEvent : SimulationEvent
```

Properties

MissileId

导弹ID

Declaration

```
public string? MissileId { get; set; }
```

Property Value

Type	Description
string	

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

威胁源仿真库文档

[Back to top](#)

Class LaserSemiActiveGuidedMissile

激光半主动制导导弹类，实现了激光半主动寻的制导功能

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [LaserSemiActiveGuidedMissile](#)

Implements

[IMissile](#)

Inherited Members

[BaseMissile.FlightTime](#)
[BaseMissile.FlightDistance](#)
[BaseMissile.EngineBurnTime](#)
[BaseMissile.IsGuidance](#)
[BaseMissile.LostGuidanceTime](#)
[BaseMissile.LastKnownVelocity](#)
[BaseMissile.GuidanceAcceleration](#)
[BaseMissile.ThrustAcceleration](#)
[BaseMissile.MissileProperties](#)
[BaseMissile.UpdateMotionState\(double\)](#)
[BaseMissile.Fire\(\)](#)
[BaseMissile.ShouldSelfDestruct\(\)](#)
[BaseMissile.Explode\(\)](#)
[BaseMissile.SelfDestruct\(\)](#)
[BaseMissile.UpdateGuidanceStatus\(\)](#)
[BaseMissile.GetRunningState\(\)](#)
[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class LaserSemiActiveGuidedMissile : BaseMissile, IMissile
```

Remarks

- 该类提供了激光半主动制导导弹的核心功能：
- 激光照射目标跟踪
 - 半主动制导控制
 - 飞行阶段管理
 - 状态监控和事件处理 通过跟踪目标反射的激光能量实现对导弹的制导控制

Constructors

LaserSemiActiveGuidedMissile(MissileProperties, ISimulationManager)

初始化激光半主动制导导弹的新实例

Declaration

```
public LaserSemiActiveGuidedMissile(MissileProperties config, ISimulationManager manager)
```

Parameters

Type	Name	Description
MissileProperties	config	导弹配置参数
ISimulationManager	manager	仿真管理器实例

Remarks

- 构造过程：
- 初始化基本属性
 - 创建制导系统
 - 配置制导参数
 - 设置初始飞行阶段

Properties

LaserDesignatorId

获取或设置激光指示器ID

Declaration

```
public string LaserDesignatorId { get; set; }
```

Property Value

Type	Description
string	激光指示器的唯一标识符

Remarks

用于识别和关联激光照射源 影响导弹的制导精度

Methods

Activate()

激活导弹

Declaration

```
public override void Activate()
```

Overrides

[BaseMissile.Activate\(\)](#)

Remarks

激活过程：

- 调用基类激活方法
- 订阅激光照射事件
- 准备目标跟踪

Deactivate()

停用导弹

Declaration

```
public override void Deactivate()
```

Overrides

[BaseMissile.Deactivate\(\)](#)

Remarks

停用过程：

- 调用基类停用方法
- 取消订阅激光照射事件
- 清理制导状态

GetStatus()

获取导弹状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含导弹状态的详细描述

Overrides

[BaseMissile.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基本状态信息
- 制导系统状态
- 目标跟踪状态

Update(double)

更新导弹状态

Declaration


```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BaseMissile.Update\(double\)](#)

Remarks

更新过程：

- 根据当前阶段更新状态
- 处理阶段转换
- 调用基类更新

Implements

[IMissile](#)

[Show / Hide Table of Contents](#)

Class UltravioletWarnerAlarmStopEvent

紫外告警器警报停止事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ UltravioletWarnerAlarmStopEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class UltravioletWarnerAlarmStopEvent : SimulationEvent
```

Properties

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Class SimulationEvent

仿真事件的基类，定义所有仿真事件的共同属性

Inheritance

↳ [object](#)

↳ [SimulationEvent](#)

- ↳ [EntityActivatedEvent](#)
- ↳ [EntityDeactivatedEvent](#)
- ↳ [EntityDestroyedEvent](#)
- ↳ [InfraredDetectionEvent](#)
- ↳ [InfraredGuidanceCommandEvent](#)
- ↳ [InfraredGuidanceMissileLightEvent](#)
- ↳ [InfraredGuidanceMissileLightOffEvent](#)
- ↳ [InfraredJammingEvent](#)
- ↳ [InfraredWarnerAlarmEvent](#)
- ↳ [InfraredWarnerAlarmStopEvent](#)
- ↳ [LaserBeamStartEvent](#)
- ↳ [LaserBeamStopEvent](#)
- ↳ [LaserBeamUpdateEvent](#)
- ↳ [LaserIlluminationStartEvent](#)
- ↳ [LaserIlluminationStopEvent](#)
- ↳ [LaserIlluminationUpdateEvent](#)
- ↳ [LaserJammingEvent](#)
- ↳ [LaserWarnerAlarmEvent](#)
- ↳ [LaserWarnerAlarmStopEvent](#)
- ↳ [MillimeterWaveDetectionEvent](#)
- ↳ [MillimeterWaveJammingEvent](#)
- ↳ [MillimeterWaveWarnerAlarmEvent](#)
- ↳ [MillimeterWaveWarnerAlarmStopEvent](#)
- ↳ [MissileFireEvent](#)
- ↳ [TankRadiationEvent](#)
- ↳ [TargetDestroyedEvent](#)
- ↳ [TargetHitEvent](#)
- ↳ [UltravioletDetectionEvent](#)
- ↳ [UltravioletWarnerAlarmEvent](#)
- ↳ [UltravioletWarnerAlarmStopEvent](#)

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class SimulationEvent
```

Remarks

所有仿真事件都继承自此类，提供：

- 事件发送者标识
- 事件发生时间戳 用于在仿真系统中追踪和处理各类事件

Properties

SenderId

获取或设置事件发送者的ID

Declaration

```
public string? SenderId { get; set; }
```

Property Value

Type	Description
string	

Remarks

用于标识事件的来源实体 可以为null，表示系统事件

Timestamp

获取或设置事件发生的时间戳

Declaration

```
public double Timestamp { get; set; }
```

Property Value

Type	Description
double	

Remarks

使用UTC时间的Ticks值 用于事件的时序管理和同步

Interface IMissile

导弹接口，定义了导弹的基本行为和状态

Namespace: [ThreatSource.Missile](#)

Assembly: ThreatSource.dll

Syntax

```
public interface IMissile
```

Remarks

该接口定义了导弹的核心功能：

- 制导状态管理
- 发射控制
- 爆炸和自毁功能
- 运行状态查询 所有具体的导弹类型都应实现此接口

Properties

IsGuidance

获取导弹是否处于制导状态

Declaration

```
bool IsGuidance { get; }
```

Property Value

Type	Description
bool	

Remarks

true表示导弹当前处于制导阶段 false表示导弹处于非制导飞行状态 用于控制导弹的制导系统工作状态

Methods

Explode()

引爆导弹，触发爆炸效果

Declaration

```
void Explode()
```

Remarks

爆炸过程包括：

- 计算爆炸范围和伤害
- 对周围目标造成伤害
- 触发爆炸事件
- 导弹自身销毁

Fire()

发射导弹，使导弹进入工作状态

Declaration

```
void Fire()
```

Remarks

发射过程包括：

- 点火启动发动机
- 初始化制导系统
- 开始飞行状态计算
- 触发导弹发射事件

GetRunningState()

获取导弹的当前运行状态信息

Declaration

```
MissileRunningState GetRunningState()
```

Returns

Type	Description
MissileRunningState	包含导弹完整状态信息的结构体

Remarks

返回的状态信息包括：

- 基本信息（ID、类型）
- 运动状态（位置、速度、朝向）
- 飞行数据（时间、距离）
- 工作状态（制导、发动机）

SelfDestruct()

执行导弹自毁程序

Declaration

```
void SelfDestruct()
```

Remarks

自毁触发条件：

- 超出最大飞行时间
- 超出最大飞行距离
- 失去制导时间过长
- 接收到自毁指令 自毁会立即停止导弹的工作并销毁

[Show / Hide Table of Contents](#)

Class InfraredWarnerAlarmEvent

红外告警器警报事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ InfraredWarnerAlarmEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredWarnerAlarmEvent : SimulationEvent
```

Properties

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Class Tank

坦克类，实现了目标接口的具体坦克目标

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [Tank](#)

Implements

[ITarget](#)

Inherited Members

[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.GetStatus\(\)](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[SimulationElement.Activate\(\)](#)
[SimulationElement.Deactivate\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Target](#)

Assembly: ThreatSource.dll

Syntax

```
public class Tank : SimulationElement, ITarget
```

Remarks

该类提供了坦克目标的完整实现：

- 继承自SimulationElement，具备基本的仿真功能
- 实现ITarget接口，提供目标特征
- 包含生命值系统，可以响应伤害
- 支持位置更新和状态同步

Constructors

Tank(string, Vector3D, double, ISimulationManager)

初始化坦克类的新实例

Declaration

```
public Tank(string id, Vector3D position, double speed, ISimulationManager simulationManager)
```

Parameters

Type	Name	Description
string	id	坦克的唯一标识符
Vector3D	position	初始位置坐标
double	speed	初始速度，单位：米/秒
ISimulationManager	simulationManager	仿真管理器实例

Remarks

构造函数设置坦克的初始状态：

- 使用默认朝向
- 设置初始位置和速度
- 生命值初始化为100

Properties

Health

获取或设置坦克的当前生命值

Declaration

```
public double Health { get; set; }
```

Property Value

Type	Description
double	

Remarks

范围：0-100 初始值为100 当生命值降至0或以下时，坦克被销毁

InfraredRadiationIntensity

获取坦克的红外辐射强度

Declaration

```
public double InfraredRadiationIntensity { get; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特/平方米 固定值100.0，表示坦克发动机和装甲的热辐射特性

RadarCrossSection

获取坦克的雷达散射截面积

Declaration

```
public double RadarCrossSection { get; }
```

Property Value

Type	Description
double	

Remarks

单位：平方米 固定值10.0，表示标准坦克的雷达反射特性

Type

获取坦克的类型标识

Declaration

```
public string Type { get; }
```

Property Value

Type	Description
string	

Remarks

固定返回"Tank"，用于标识这是一个坦克目标

Methods

TakeDamage(double)

对坦克造成伤害

Declaration

```
public void TakeDamage(double damage)
```

Parameters

Type	Name	Description
double	damage	伤害值

Remarks

伤害处理过程：

- 从当前生命值中扣除伤害值
- 如果生命值降至0或以下，触发目标销毁事件
- 销毁事件会通知仿真系统移除该目标

Update(double)

更新坦克的状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[SimulationElement.Update\(double\)](#)

Remarks

更新过程：

- 根据当前速度更新位置
- 位置更新使用简单的线性运动模型

Implements

[ITarget](#)

Class LaserIlluminationStopEvent

激光照射停止事件，表示激光定位器停止照射目标

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserIlluminationStopEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class LaserIlluminationStopEvent : SimulationEvent
```

Remarks

用于通知系统激光照射已结束 触发时机：激光定位器停止照射或照射中断时

Properties

LaserDesignatorId

获取或设置激光定位器的ID

Declaration

```
public string? LaserDesignatorId { get; set; }
```

Property Value

Type	Description
string	

TargetId

获取或设置被照射目标的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Namespace ThreatSource.Sensor

Classes

AltimeterSensorData

测高仪传感器数据类，包含高度测量的具体数据

InfraredDetector

红外探测器类，实现了红外辐射的探测和目标识别功能

InfraredSensorData

红外传感器数据类，包含红外探测的具体数据

LaserRangefinder

激光测距仪类，实现了目标距离测量和数据采集功能

MillimeterWaveAltimeter

毫米波测高雷达类，实现了高度测量和数据采集功能

MillimeterWaveRadiometer

毫米波辐射计类，实现了目标辐射温度测量和目标识别功能

RadiometerSensorData

辐射计传感器数据类，包含毫米波辐射探测的具体数据

RangefinderSensorData

测距仪传感器数据类，包含距离测量的具体数据

Sensor

传感器的抽象基类，实现了ISensor接口的基本功能

SensorData

传感器数据的抽象基类，定义了所有传感器数据的通用属性

Interfaces

ISensor

Interface IIndicator

指示器接口，定义了所有指示器的通用功能

Namespace: [ThreatSource.Indicator](#)

Assembly: ThreatSource.dll

Syntax

```
public interface IIndicator
```

Remarks

该接口提供了指示器的基本功能：

- 目标和导弹的关联
- 运行状态的获取
- 位置和姿态的管理 是激光指示器、红外跟踪器等具体指示器的抽象基础

Properties

MissileId

获取或设置导弹ID

Declaration

```
string? MissileId { get; }
```

Property Value

Type	Description
string	

Remarks

用于标识和控制关联的导弹 可以为null表示当前没有关联导弹

TargetId

获取或设置目标ID

Declaration

```
string? TargetId { get; }
```

Property Value

Type	Description
string	

Remarks

用于标识和跟踪目标 可以为null表示当前没有关联目标

Methods

GetRunningState()

获取指示器的当前运行状态

Declaration

```
IndicatorRunningState GetRunningState()
```

Returns

Type	Description
IndicatorRunningState	包含指示器完整状态信息的结构体

Remarks

返回的状态信息包括：

- 目标和导弹的关联状态
- 指示器的工作状态
- 位置、速度和姿态信息
- 激活状态

Namespace ThreatSource.Indicator

Classes

InfraredTracker

红外测角仪类，实现了红外制导导弹的跟踪和制导功能

LaserBeamRider

激光波束制导器类，用于生成激光波束制导场

LaserDesignator

激光指示器类，实现了对目标的激光照射和制导功能

Structs

IndicatorRunningState

指示器运行状态结构体，包含了指示器的完整状态信息

Interfaces

IIndicator

指示器接口，定义了所有指示器的通用功能

Enums

IndicatorState

指示器状态枚举，定义了指示器的工作状态

IndicatorType

指示器类型枚举，定义了支持的指示器类型

Class InfraredImagingTerminalGuidedMissile

红外成像末制导导弹类，继承自基础导弹类

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [InfraredImagingTerminalGuidedMissile](#)

Implements

[IMissile](#)

Inherited Members

[BaseMissile.FlightTime](#)
[BaseMissile.FlightDistance](#)
[BaseMissile.EngineBurnTime](#)
[BaseMissile.IsGuidance](#)
[BaseMissile.LostGuidanceTime](#)
[BaseMissile.LastKnownVelocity](#)
[BaseMissile.GuidanceAcceleration](#)
[BaseMissile.ThrustAcceleration](#)
[BaseMissile.MissileProperties](#)
[BaseMissile.UpdateMotionState\(double\)](#)
[BaseMissile.Fire\(\)](#)
[BaseMissile.ShouldSelfDestruct\(\)](#)
[BaseMissile.Activate\(\)](#)
[BaseMissile.Deactivate\(\)](#)
[BaseMissile.SelfDestruct\(\)](#)
[BaseMissile.UpdateGuidanceStatus\(\)](#)
[BaseMissile.GetRunningState\(\)](#)
[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class InfraredImagingTerminalGuidedMissile : BaseMissile, IMissile
```

Remarks

该类提供了红外成像末制导导弹的完整实现：

- 继承自BaseMissile，具备基本的导弹功能
- 实现了红外成像制导系统
- 支持多阶段飞行控制
- 具备红外目标识别能力
- 提供精确的末制导打击能力

工作流程：

1. 导弹发射并进入发射阶段
2. 切换到巡航阶段，进行中程飞行
3. 进入末制导阶段，启动红外成像制导
4. 接近目标后引爆或达到自毁条件后自毁

Constructors

InfraredImagingTerminalGuidedMissile(MissileProperties, ISimulationManager)

初始化红外成像末制导导弹的新实例

Declaration

```
public InfraredImagingTerminalGuidedMissile(MissileProperties config, ISimulationManager manager)
```

Parameters

Type	Name	Description
MissileProperties	<i>config</i>	导弹的配置参数
ISimulationManager	<i>manager</i>	仿真管理器实例

Remarks

构造过程：

1. 调用基类构造函数
2. 创建制导系统实例
3. 设置初始飞行阶段

Methods

Explode()

执行导弹爆炸

Declaration

```
public override void Explode()
```

Overrides

[BaseMissile.Explode\(\)](#)

Remarks

爆炸过程：

1. 调用基类的爆炸方法
2. 切换到爆炸阶段

GetStatus()

获取导弹的状态信息字符串

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含导弹状态的详细描述

Overrides

[BaseMissile.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基类状态信息
- 当前飞行阶段
- 制导系统状态
- 目标跟踪信息

Update(double)

更新导弹的状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BaseMissile.Update\(double\)](#)

Remarks

更新过程：

1. 调用基类的更新方法
2. 根据当前阶段选择更新方法
3. 执行相应阶段的更新逻辑
4. 检查是否需要自毁

Implements

[IMissile](#)

Class LaserBeamRiderGuidanceSystem

激光驾束制导系统类，实现了基于激光束跟踪的制导功能

Inheritance

↳ [object](#)
↳ [BasicGuidanceSystem](#)
↳ [LaserBeamRiderGuidanceSystem](#)

Implements

[IGuidanceSystem](#)

Inherited Members

[BasicGuidanceSystem.HasGuidance](#)
[BasicGuidanceSystem.MaxAcceleration](#)
[BasicGuidanceSystem.ProportionalNavigationCoefficient](#)
[BasicGuidanceSystem.Position](#)
[BasicGuidanceSystem.Velocity](#)
[BasicGuidanceSystem.GuidanceAcceleration](#)
[BasicGuidanceSystem.GetGuidanceAcceleration\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Guidance](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserBeamRiderGuidanceSystem : BasicGuidanceSystem, IGuidanceSystem
```

Remarks

该类提供了激光驾束制导系统的核心功能：

- 激光束生成和控制
- 偏差检测和计算
- PID控制器
- 非线性增益控制 用于实现高精度的制导控制

Constructors

LaserBeamRiderGuidanceSystem(double, double)

初始化激光驾束制导系统的新实例

Declaration

```
public LaserBeamRiderGuidanceSystem(double maxAcceleration, double guidanceCoefficient)
```

Parameters

Type	Name	Description
double	<i>maxAcceleration</i>	最大加速度，单位：米/平方秒
double	<i>guidanceCoefficient</i>	制导系数

Remarks

构造过程：

- 初始化基类参数
- 初始化激光参数
- 初始化控制参数

Properties

LaserPower

获取或设置激光功率，单位：瓦特

Declaration

```
public double LaserPower { get; set; }
```

Property Value

Type	Description
double	

Remarks

记录激光源的发射功率 影响系统的探测距离和可靠性

LastGuidanceAcceleration

获取上一次的制导加速度

Declaration

```
public Vector3D LastGuidanceAcceleration { get; }
```

Property Value

Type	Description
Vector3D	

Remarks

记录历史制导指令 用于实现低通滤波

Methods

CalculateGuidanceAcceleration(double)

计算制导加速度

Declaration

```
protected override void CalculateGuidanceAcceleration(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BasicGuidanceSystem.CalculateGuidanceAcceleration\(double\)](#)

Remarks

计算过程：

- 计算偏差向量
- 执行PID控制
- 应用非线性增益
- 计算横向加速度
- 计算前向加速度
- 合成最终制导指令
- 应用低通滤波

DeactivateLaserBeam()

关闭激光照射系统

Declaration

```
public void DeactivateLaserBeam()
```

Remarks

关闭过程：

- 清除位置信息
- 清除方向信息
- 清除功率参数
- 停止制导

GetStatus()

获取制导系统的详细状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含完整状态参数的字符串

Overrides

[BasicGuidanceSystem.GetStatus\(\)](#)

Remarks

返回信息：

- 基本状态信息
- 制导参数
- 控制状态 用于系统监控和调试

Update(double, Vector3D, Vector3D)

更新制导系统的状态和计算结果

Declaration

```
public override void Update(double deltaTime, Vector3D missilePosition, Vector3D missileVelocity)
```

Parameters

Type	Name	Description
double	deltaTime	时间步长，单位：秒
Vector3D	missilePosition	导弹位置，单位：米
Vector3D	missileVelocity	导弹速度，单位：米/秒

Overrides

[BasicGuidanceSystem.Update\(double, Vector3D, Vector3D\)](#)

Remarks

更新过程：

- 检查激光照射状态
- 更新制导状态
- 计算制导指令
- 更新输出参数

UpdateLaserBeamRider(Vector3D, Vector3D, double)

更新激光驾束仪参数

Declaration

```
public void UpdateLaserBeamRider(Vector3D sourcePosition, Vector3D direction, double laserPower)
```

Parameters

Type	Name	Description
Vector3D	sourcePosition	激光源位置，单位：米
Vector3D	direction	激光方向向量
double	laserPower	激光功率，单位：瓦特

Remarks

更新过程：

- 激活激光照射
- 更新位置信息
- 更新方向信息
- 更新功率参数

Implements

[IGuidanceSystem](#)

Class InfraredImagingGuidanceSystem

红外成像引导系统类，实现了基于红外图像的目标探测和跟踪功能

Inheritance

↳ [object](#)
↳ [BasicGuidanceSystem](#)
↳ [InfraredImagingGuidanceSystem](#)

Implements

[IGuidanceSystem](#)

Inherited Members

[BasicGuidanceSystem.HasGuidance](#)
[BasicGuidanceSystem.MaxAcceleration](#)
[BasicGuidanceSystem.ProportionalNavigationCoefficient](#)
[BasicGuidanceSystem.Position](#)
[BasicGuidanceSystem.Velocity](#)
[BasicGuidanceSystem.GuidanceAcceleration](#)
[BasicGuidanceSystem.GetGuidanceAcceleration\(\)](#)
[BasicGuidanceSystem.CalculateGuidanceAcceleration\(double\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Guidance](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredImagingGuidanceSystem : BasicGuidanceSystem, IGuidanceSystem
```

Remarks

该类提供了红外成像制导系统的核心功能：

- 目标自动探测
- 图像识别处理
- 信噪比计算
- 比例导引控制 用于实现自主寻的制导

Constructors

InfraredImagingGuidanceSystem(double, double, ISimulationManager)

初始化红外成像引导系统的新实例

Declaration

```
public InfraredImagingGuidanceSystem(double maxAcceleration, double proportionalNavigationCoefficient, ISimulationManager simulationManager)
```

Parameters

Type	Name	Description
double	<i>maxAcceleration</i>	最大加速度，单位：米/平方秒
double	<i>proportionalNavigationCoefficient</i>	比例导引系数
ISimulationManager	<i>simulationManager</i>	仿真管理器实例

Remarks

构造过程：

- 初始化基类参数
- 设置仿真管理器
- 初始化目标位置

Properties

SimulationManager

获取或设置仿真管理器实例

Declaration

```
public ISimulationManager SimulationManager { get; set; }
```

Property Value

Type	Description
ISimulationManager	

Remarks

用于获取场景中的目标信息 实现目标探测功能

Methods

GetStatus()

获取制导系统的详细状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含完整状态参数的字符串

Overrides

[BasicGuidanceSystem.GetStatus\(\)](#)

Remarks

返回信息：

- 基本状态信息
- 制导加速度

- 目标位置 用于系统监控和调试

Update(double, Vector3D, Vector3D)

更新引导系统的状态和计算结果

Declaration

```
public override void Update(double deltaTime, Vector3D missilePosition, Vector3D missileVelocity)
```

Parameters

Type	Name	Description
double	deltaTime	自上次更新以来的时间间隔，单位：秒
Vector3D	missilePosition	导弹当前位置，单位：米
Vector3D	missileVelocity	导弹当前速度，单位：米/秒

Overrides

[BasicGuidanceSystem.Update\(double, Vector3D, Vector3D\)](#)

Remarks

更新过程：

- 探测目标位置
- 计算目标速度
- 生成制导指令
- 限制最大加速度

Implements

[IGuidanceSystem](#)

[Show / Hide Table of Contents](#)

Class UltravioletWarnerConfig

紫外告警器配置类，用于设置紫外探测和告警系统的参数

Inheritance

↳ [object](#)

↳ UltravioletWarnerConfig

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class UltravioletWarnerConfig
```

Remarks

该类定义了紫外告警器的关键参数：

- 设备标识
- 灵敏度阈值
- 警报持续时间
- 探测波长范围 这些参数决定了告警器对紫外辐射的探测能力

Constructors

UltravioletWarnerConfig()

初始化紫外告警器配置的新实例

Declaration

```
public UltravioletWarnerConfig()
```

Remarks

设置默认值：

- ID为空字符串
- 灵敏度阈值为0
- 警报持续时间为5秒
- 波长范围为0-0纳米

Properties

AlarmDuration

获取或设置警报持续时间

Declaration

```
public double AlarmDuration { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 定义了每次告警的持续时间

Id

获取或设置紫外告警器的唯一标识符

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和查找特定的告警器

SensitivityThreshold

获取或设置告警灵敏度阈值

Declaration

```
public double SensitivityThreshold { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特/平方米 当接收到的紫外辐射强度超过此阈值时触发告警

WavelengthMax

获取或设置最大探测波长

Declaration

```
public double WavelengthMax { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最长紫外波长

WavelengthMin

获取或设置最小探测波长

Declaration

```
public double WavelengthMin { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最短紫外波长

[Show / Hide Table of Contents](#)

Class BasicGuidanceSystem

基础制导系统类，实现了制导系统的基本功能框架

Inheritance

↳ [object](#)

↳ [BasicGuidanceSystem](#)

↳ [InfraredCommandGuidanceSystem](#)

↳ [InfraredImagingGuidanceSystem](#)

↳ [LaserBeamRiderGuidanceSystem](#)

↳ [LaserSemiActiveGuidanceSystem](#)

↳ [MillimeterWaveGuidanceSystem](#)

Implements

[IGuidanceSystem](#)

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Guidance](#)

Assembly: ThreatSource.dll

Syntax

```
public class BasicGuidanceSystem : IGuidanceSystem
```

Remarks

该类提供了制导系统的通用实现：

- 制导状态管理
- 运动参数记录
- 加速度限制
- 比例导引系数 是其他具体制导系统的基类

Constructors

BasicGuidanceSystem(double, double)

初始化基础制导系统的新实例

Declaration

```
public BasicGuidanceSystem(double maxAcceleration, double proportionalNavigationCoefficient)
```

Parameters

Type	Name	Description
double	<i>maxAcceleration</i>	最大加速度，单位：米/平方秒
double	<i>proportionalNavigationCoefficient</i>	比例导引系数

Remarks

构造过程：

- 初始化制导状态
- 设置运动参数
- 配置制导参数

Properties

GuidanceAcceleration

获取或设置制导加速度，单位：米/平方秒

Declaration

```
protected Vector3D GuidanceAcceleration { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

存储计算得到的制导指令 用于控制导弹的运动轨迹

HasGuidance

获取或设置是否有有效的制导信息

Declaration

```
public bool HasGuidance { get; protected set; }
```

Property Value

Type	Description
bool	

Remarks

true表示当前有可用的制导信息 false表示无法获得有效制导 用于判断制导系统的工作状态

MaxAcceleration

获取或设置最大加速度，单位：米/平方秒

Declaration

```
public double MaxAcceleration { get; set; }
```

Property Value

Type	Description
double	

Remarks
定义了制导加速度的上限 用于限制导弹的机动能力 考虑结构和动力限制

Position

获取或设置导弹位置，单位：米

Declaration

```
protected Vector3D Position { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks
记录导弹的当前位置 用于制导计算和状态更新

ProportionalNavigationCoefficient

获取或设置比例导引系数

Declaration

```
public double ProportionalNavigationCoefficient { get; set; }
```

Property Value

Type	Description
double	

Remarks
定义了制导指令的增益 影响制导系统的响应特性 通常取值3~5

Velocity

获取或设置导弹速度，单位：米/秒

Declaration

```
protected Vector3D Velocity { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks
记录导弹的当前速度 用于制导计算和状态更新

Methods

CalculateGuidanceAcceleration(double)

计算制导加速度（需要在子类中实现）

Declaration

```
protected virtual void CalculateGuidanceAcceleration(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间间隔，单位：秒

Remarks

计算要求：

- 实现特定的制导律
- 考虑最大加速度限制
- 更新制导加速度 基类中不实现具体计算

GetGuidanceAcceleration()

获取制导加速度指令

Declaration

```
public Vector3D GetGuidanceAcceleration()
```

Returns

Type	Description
Vector3D	三维制导加速度向量，单位：米/平方秒

Remarks

返回数据：

- X分量：横向制导加速度
- Y分量：垂直制导加速度
- Z分量：纵向制导加速度 用于导弹的轨迹控制

GetStatus()

获取制导系统的状态信息

Declaration

```
public virtual string GetStatus()
```

Returns

Type	Description
string	包含关键状态参数的字符串

Remarks

返回信息：

- 制导状态
- 位置信息
- 速度信息
- 加速度信息 用于状态监控和调试

Update(double, Vector3D, Vector3D)

更新制导系统的状态和计算结果

Declaration

```
public virtual void Update(double deltaTime, Vector3D missilePosition, Vector3D missileVelocity)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒
Vector3D	<i>missilePosition</i>	导弹当前位置，单位：米
Vector3D	<i>missileVelocity</i>	导弹当前速度，单位：米/秒

Remarks

更新过程：

- 更新位置信息
- 更新速度信息
- 准备制导计算

Implements

[IGuidanceSystem](#)

[Show / Hide Table of Contents](#)

Class MissileFireEvent

导弹发射事件，表示导弹被发射的状态变化

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [MissileFireEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class MissileFireEvent : SimulationEvent
```

Remarks

在导弹发射时触发，包含目标信息 用于通知其他实体和系统导弹已发射

Properties

TargetId

获取或设置目标实体的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识导弹的预定目标 可以为null 表示无预定目标

[Show / Hide Table of Contents](#)

Interface ISensor

定义传感器的基本接口，规范了所有传感器的通用功能

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public interface ISensor
```

Remarks

该接口提供了传感器的基本功能规范：

- 激活和停用控制
- 状态更新机制
- 数据采集接口 是所有具体传感器实现的基础

Properties

IsActive

获取或设置传感器是否处于激活状态

Declaration

```
bool IsActive { get; set; }
```

Property Value

Type	Description
bool	

Remarks

true表示传感器正在工作 false表示传感器处于待机状态 用于控制传感器的工作状态

Methods

Activate()

激活传感器，使其开始工作

Declaration

```
void Activate()
```

Remarks

激活过程：

- 初始化工作参数
- 启动数据采集
- 开始状态监控

Deactivate()

停用传感器，使其停止工作

Declaration

```
void Deactivate()
```

Remarks

停用过程：

- 停止数据采集
- 清理工作状态
- 释放相关资源

GetSensorData()

获取传感器采集的最新数据

Declaration

```
SensorData GetSensorData()
```

Returns

Type	Description
SensorData	包含传感器测量结果的数据对象

Remarks

返回数据：

- 测量的物理量
- 时间戳信息
- 状态标志

Update(double)

更新传感器的工作状态

Declaration

```
void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒

Remarks

更新过程：

- 采集最新数据
- 更新内部状态
- 处理异常情况

Class LaserWarnerAlarmStopEvent

激光告警器警报停止事件，表示激光照射结束

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserWarnerAlarmStopEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserWarnerAlarmStopEvent : SimulationEvent
```

Remarks

用于模拟激光告警器停止报警 触发时机：激光照射结束或低于警戒阈值时

Properties

TargetId

获取或设置目标的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识激光照射已结束的实体

Class LaserBeamRiderMissile

激光波束制导导弹类，实现了激光波束跟踪制导的导弹功能

Inheritance

↳ [object](#)
↳ [SimulationElement](#)
↳ [BaseMissile](#)
↳ [LaserBeamRiderMissile](#)

Implements

[IMissile](#)

Inherited Members

[BaseMissile.FlightTime](#)
[BaseMissile.FlightDistance](#)
[BaseMissile.EngineBurnTime](#)
[BaseMissile.IsGuidance](#)
[BaseMissile.LostGuidanceTime](#)
[BaseMissile.LastKnownVelocity](#)
[BaseMissile.GuidanceAcceleration](#)
[BaseMissile.ThrustAcceleration](#)
[BaseMissile.MissileProperties](#)
[BaseMissile.UpdateMotionState\(double\)](#)
[BaseMissile.Fire\(\)](#)
[BaseMissile.ShouldSelfDestruct\(\)](#)
[BaseMissile.Explode\(\)](#)
[BaseMissile.SelfDestruct\(\)](#)
[BaseMissile.UpdateGuidanceStatus\(\)](#)
[BaseMissile.GetRunningState\(\)](#)
[SimulationElement.Id](#)
[SimulationElement.Position](#)
[SimulationElement.Speed](#)
[SimulationElement.Velocity](#)
[SimulationElement.Orientation](#)
[SimulationElement.IsActive](#)
[SimulationElement.SimulationManager](#)
[SimulationElement.PublishEvent\(SimulationEvent\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: [ThreatSource.dll](#)

Syntax


```
public class LaserBeamRiderMissile : BaseMissile, IMissile
```

Remarks

该类提供了激光波束制导导弹的核心功能：

- 激光波束跟踪
- 波束制导控制
- 飞行阶段管理
- 状态监控和事件处理 通过跟踪激光波束的中心线实现对导弹的制导控制

Constructors

LaserBeamRiderMissile(MissileProperties, ISimulationManager)

初始化激光波束制导导弹的新实例

Declaration

```
public LaserBeamRiderMissile(MissileProperties config, ISimulationManager manager)
```

Parameters

Type	Name	Description
MissileProperties	<i>config</i>	导弹配置参数
ISimulationManager	<i>manager</i>	仿真管理器实例

Remarks

构造过程：

- 初始化基本属性
- 创建波束制导系统
- 配置制导参数
- 设置初始飞行阶段

Methods

Activate()

激活导弹

Declaration

```
public override void Activate()
```

Overrides

[BaseMissile.Activate\(\)](#)

Remarks

激活过程：

- 调用基类激活方法
- 订阅激光波束事件
- 准备波束跟踪

Deactivate()

停用导弹

Declaration

```
public override void Deactivate()
```

Overrides

[BaseMissile.Deactivate\(\)](#)

Remarks

停用过程：

- 调用基类停用方法
- 取消订阅波束事件
- 清理制导状态

GetStatus()

获取导弹状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含导弹状态的详细描述

Overrides

[BaseMissile.GetStatus\(\)](#)

Remarks

返回信息包括：

- 基本状态信息
- 波束跟踪状态
- 制导系统状态

Update(double)

更新导弹状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BaseMissile.Update\(double\)](#)

Remarks

更新过程：

- 根据当前阶段更新状态
- 处理阶段转换
- 调用基类更新

Implements

[IMissile](#)

[Show / Hide Table of Contents](#)

Struct IndicatorRunningState

指示器运行状态结构体，包含了指示器的完整状态信息

Inherited Members

[ValueType.Equals\(object\)](#)
[ValueType.GetHashCode\(\)](#)
[ValueType.ToString\(\)](#)
[object.Equals\(object, object\)](#)
[object.GetType\(\)](#)
[object.ReferenceEquals\(object, object\)](#)

Namespace: [ThreatSource.Indicator](#)

Assembly: ThreatSource.dll

Syntax

```
public struct IndicatorRunningState
```

Remarks

该结构体封装了指示器的所有关键状态：

- 关联信息（目标和导弹）
- 工作状态（类型和状态）
- 运动状态（位置、速度、姿态）
- 功能状态（是否激活） 用于状态监控和控制决策

Properties

IsActive

获取或设置指示器是否激活

Declaration

```
public bool IsActive { readonly get; set; }
```

Property Value

Type	Description
bool	

Remarks

表示指示器是否处于工作状态 true表示正在工作，false表示待机

MissileId

获取或设置导弹ID

Declaration

```
public string MissileId { readonly get; set; }
```

Property Value

Type	Description
string	

Remarks

标识当前控制的导弹 用于导弹制导和状态同步

Orientation

获取或设置指示器朝向

Declaration

```
public Orientation Orientation { readonly get; set; }
```

Property Value

Type	Description
Orientation	

Remarks

指示器在三维空间中的姿态 影响指示器的工作效果和精度

Position

获取或设置指示器位置

Declaration

```
public Vector3D Position { readonly get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

指示器在三维空间中的位置向量 用于空间定位和运动控制

State

获取或设置指示器状态

Declaration

```
public IndicatorState State { readonly get; set; }
```

Property Value

Type	Description
IndicatorState	

Remarks

表示指示器的当前工作状态 用于控制指示器的工作模式

TargetId

获取或设置目标ID

Declaration

```
public string TargetId { readonly get; set; }
```

Property Value

Type	Description
string	

Remarks

标识当前跟踪的目标 用于目标关联和状态更新

Type

获取或设置指示器类型

Declaration

```
public IndicatorType Type { readonly get; set; }
```

Property Value

Type	Description
IndicatorType	

Remarks

表示指示器的具体类型 影响指示器的工作模式和性能参数

Velocity

获取或设置指示器速度

Declaration

```
public Vector3D Velocity { readonly get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

指示器在三维空间中的速度向量 用于运动状态更新和预测

Class EntityDeactivatedEvent

实体停用事件，表示仿真实体被停用

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ EntityDeactivatedEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class EntityDeactivatedEvent : SimulationEvent
```

Remarks

用于通知系统某个实体已停止活动 触发时机：实体被停用或暂时移除时

Properties

DeactivatedEntityId

获取或设置被停用实体的ID

Declaration

```
public string? DeactivatedEntityId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识已被停用的实体

Class LaserBeamRiderConfig

激光波束制导仪配置类，用于设置激光波束制导系统的参数

Inheritance

↳ [object](#)

↳ LaserBeamRiderConfig

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserBeamRiderConfig
```

Remarks

该类定义了波束制导系统的关键参数：

- 设备标识
- 发射位置
- 激光功率
- 控制场尺寸
- 最大制导距离 这些参数决定了波束制导系统的性能和工作范围

Constructors

LaserBeamRiderConfig()

初始化激光波束制导仪配置的新实例

Declaration

```
public LaserBeamRiderConfig()
```

Remarks

设置默认值：

- ID为空字符串
- 初始位置为原点(0,0,0)
- 激光功率为0瓦特
- 控制场直径为0米

Properties

ControlFieldDiameter

获取或设置控制场直径

Declaration

```
public double ControlFieldDiameter { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米 定义了导弹可以接收制导信号的横向范围

Id

获取或设置激光波束制导仪的唯一标识符

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和查找特定的波束制导仪

InitialPosition

获取或设置波束制导仪的初始位置

Declaration

```
public Vector3D InitialPosition { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

使用三维向量表示空间位置 坐标系：右手坐标系，X向右，Y向上，Z向前

LaserPower

获取或设置激光功率

Declaration

```
public double LaserPower { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特 影响制导波束的有效距离和导引精度

MaxGuidanceDistance

获取或设置最大导引距离

Declaration

```
public double MaxGuidanceDistance { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米 定义了波束制导系统的最大有效工作距离

Class LaserWarnerAlarmEvent

激光告警器警报事件，表示检测到激光照射

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserWarnerAlarmEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserWarnerAlarmEvent : SimulationEvent
```

Remarks

用于模拟激光告警器的响应 触发时机：检测到激光照射时

Properties

TargetId

获取或设置被照射目标的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识检测到激光照射的实体

[Show / Hide Table of Contents](#)

Class LaserJammingEvent

激光干扰事件，表示对激光制导系统的干扰

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserJammingEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class LaserJammingEvent : SimulationEvent
```

Remarks

用于模拟激光干扰效果 包含干扰功率信息

Properties

JammingPower

获取或设置干扰功率值

Declaration

```
public double JammingPower { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特 表示干扰源的输出功率

TargetId

获取或设置被干扰目标的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识受到激光干扰的目标实体

Struct Vector2D

表示二维平面上的向量

Inherited Members

[ValueType.Equals\(object\)](#)
[ValueType.GetHashCode\(\)](#)
[ValueType.ToString\(\)](#)
[object.Equals\(object, object\)](#)
[object.GetType\(\)](#)
[object.ReferenceEquals\(object, object\)](#)

Namespace: [ThreatSource.Utils](#)

Assembly: ThreatSource.dll

Syntax

```
public struct Vector2D
```

Remarks

用于处理二维平面上的计算，如目标投影等

Constructors

Vector2D(double, double)

初始化二维向量的新实例

Declaration

```
public Vector2D(double x, double y)
```

Parameters

Type	Name	Description
double	<i>x</i>	X分量的值
double	<i>y</i>	Y分量的值

Properties

X

获取或设置向量的X分量

Declaration

```
public double X { readonly get; set; }
```

Property Value

Type	Description
double	

Y

获取或设置向量的Y分量

Declaration

```
public double Y { readonly get; set; }
```

Property Value

Type	Description
double	

Zero

零向量 (0, 0)

Declaration

```
public static Vector2D Zero { get; }
```

Property Value

Type	Description
Vector2D	

Interface ISimulationAdapter

第三方仿真环境适配器接口

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public interface ISimulationAdapter
```

Remarks

该接口用于与外部仿真环境进行集成，提供实体查询和事件交互功能。实现此接口可以将威胁源仿真库与其他仿真系统对接。

Methods

GetEntity(string)

从外部仿真环境获取实体

Declaration

```
object? GetEntity(string id)
```

Parameters

Type	Name	Description
string	<i>id</i>	实体ID

Returns

Type	Description
object	实体对象，如果不存在则返回null

PublishToExternalSimulation<T>(T)

发布事件到第三方仿真环境

Declaration

```
void PublishToExternalSimulation<T>(T evt)
```

Parameters

Type	Name	Description
T	<i>evt</i>	要发布的事件

Type Parameters

Name	Description
<i>T</i>	事件类型

Remarks

用于将威胁源仿真库中的事件同步到外部仿真环境

ReceiveFromExternalSimulation<T>(T)

从第三方仿真环境接收事件

Declaration

```
void ReceiveFromExternalSimulation<T>(T evt)
```

Parameters

Type	Name	Description
T	<i>evt</i>	接收到的事件

Type Parameters

Name	Description
<i>T</i>	事件类型

Remarks

用于接收外部仿真环境的事件并在威胁源仿真库中处理

[Show / Hide Table of Contents](#)

Class LaserWarnerConfig

激光告警器配置类，用于设置激光探测和告警系统的参数

Inheritance

↳ [object](#)
↳ LaserWarnerConfig

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserWarnerConfig
```

Remarks

该类定义了激光告警器的关键参数：

- 设备标识
- 灵敏度阈值
- 警报持续时间
- 探测波长范围 这些参数决定了告警器的探测能力和响应特性

Constructors

LaserWarnerConfig()

初始化激光告警器配置的新实例

Declaration

```
public LaserWarnerConfig()
```

Remarks

设置默认值：

- ID为空字符串
- 灵敏度阈值为0
- 警报持续时间为5秒
- 波长范围为0-0纳米

Properties

AlarmDuration

获取或设置警报持续时间

Declaration

```
public double AlarmDuration { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 定义了每次告警的持续时间

Id

获取或设置激光告警器的唯一标识符

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和查找特定的告警器

SensitivityThreshold

获取或设置告警灵敏度阈值

Declaration

```
public double SensitivityThreshold { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特/平方米 当接收到的激光功率密度超过此阈值时触发告警

WavelengthMax

获取或设置最大探测波长

Declaration

```
public double WavelengthMax { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最长波长

WavelengthMin

获取或设置最小探测波长

Declaration

```
public double WavelengthMin { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最短波长

Class InfraredCommandGuidanceSystem

红外指令导引系统类，实现了基于红外跟踪器的指令制导功能

Inheritance

↳ [object](#)
↳ [BasicGuidanceSystem](#)
↳ [InfraredCommandGuidanceSystem](#)

Implements

[IGuidanceSystem](#)

Inherited Members

[BasicGuidanceSystem.HasGuidance](#)
[BasicGuidanceSystem.MaxAcceleration](#)
[BasicGuidanceSystem.ProportionalNavigationCoefficient](#)
[BasicGuidanceSystem.Position](#)
[BasicGuidanceSystem.Velocity](#)
[BasicGuidanceSystem.GuidanceAcceleration](#)
[BasicGuidanceSystem.GetGuidanceAcceleration\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Guidance](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredCommandGuidanceSystem : BasicGuidanceSystem, IGuidanceSystem
```

Remarks

该类提供了红外指令制导系统的核心功能：

- 指令接收和处理
- 转向速率控制
- 提前量计算
- 制导加速度生成 用于实现地面指令制导的导弹控制

Constructors

InfraredCommandGuidanceSystem(double, double)

初始化红外指令导引系统的新实例

Declaration

```
public InfraredCommandGuidanceSystem(double maxAcceleration, double guidanceCoefficient)
```

Parameters

Type	Name	Description
double	<i>maxAcceleration</i>	最大加速度，单位：米/平方秒
double	<i>guidanceCoefficient</i>	制导系数

Remarks

构造过程：

- 初始化基类参数
- 初始化向量记录
- 清零转向速率

Methods

CalculateGuidanceAcceleration(double)

计算制导加速度

Declaration

```
protected override void CalculateGuidanceAcceleration(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间间隔，单位：秒

Overrides

[BasicGuidanceSystem.CalculateGuidanceAcceleration\(double\)](#)

Remarks

计算过程：

- 计算期望方向
- 更新转向速率
- 计算提前量
- 生成制导指令
- 限制最大加速度

GetStatus()

获取制导系统的详细状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含完整状态参数的字符串

Overrides

[BasicGuidanceSystem.GetStatus\(\)](#)

Remarks

返回信息：

- 基本状态信息
- 制导加速度
- 方向向量
- 转向速率 用于系统监控和调试

ReceiveGuidanceCommand(Vector3D, Vector3D)

接收并处理制导指令

Declaration

```
public void ReceiveGuidanceCommand(Vector3D trackerToMissileVector, Vector3D trackerToTargetVector)
```

Parameters

Type	Name	Description
Vector3D	trackerToMissileVector	跟踪器到导弹的方向向量
Vector3D	trackerToTargetVector	跟踪器到目标的方向向量

Remarks

处理过程：

- 更新制导状态
- 记录方向向量
- 准备制导计算

Update(double, Vector3D, Vector3D)

更新制导系统的状态和计算结果

Declaration

```
public override void Update(double deltaTime, Vector3D missilePosition, Vector3D missileVelocity)
```

Parameters

Type	Name	Description
double	deltaTime	自上次更新以来的时间间隔，单位：秒
Vector3D	missilePosition	导弹当前位置，单位：米
Vector3D	missileVelocity	导弹当前速度，单位：米/秒

Overrides

[BasicGuidanceSystem.Update\(double, Vector3D, Vector3D\)](#)

Remarks

更新过程：

- 更新基类状态
- 计算制导加速度

Implements

[IGuidanceSystem](#)

Class Sensor

传感器的抽象基类，实现了ISensor接口的基本功能

Inheritance

↳ [object](#)

- ↳ [Sensor](#)
 - ↳ [InfraredDetector](#)
 - ↳ [LaserRangefinder](#)
 - ↳ [MillimeterWaveAltimeter](#)
 - ↳ [MillimeterWaveRadiometer](#)

Implements

[ISensor](#)

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public abstract class Sensor : ISensor
```

Remarks

该类提供了传感器的通用实现：

- 位置和姿态管理
- 激活状态控制
- 数据采集框架 是具体传感器类的实现基础

Constructors

Sensor(Vector3D, Orientation)

初始化传感器的新实例

Declaration

```
protected Sensor(Vector3D position, Orientation orientation)
```

Parameters

Type	Name	Description
------	------	-------------

Type	Name	Description
Vector3D	<i>position</i>	传感器的初始位置
Orientation	<i>orientation</i>	传感器的初始朝向

Remarks

构造过程：

- 设置初始位置
- 设置初始朝向
- 初始化工作状态

Properties

IsActive

获取或设置传感器是否处于激活状态

Declaration

```
public bool IsActive { get; set; }
```

Property Value

Type	Description
bool	

Remarks

true表示传感器正在工作 false表示传感器处于待机状态 控制传感器的工作模式

Orientation

获取或设置传感器的朝向

Declaration

```
protected Orientation Orientation { get; set; }
```

Property Value

Type	Description
Orientation	

Remarks

传感器在三维空间中的姿态 影响传感器的探测方向和精度

Position

获取或设置传感器的位置

Declaration

```
protected Vector3D Position { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

传感器在三维空间中的位置向量 用于空间定位和测量计算

Methods

Activate()

激活传感器，使其开始工作

Declaration

```
public virtual void Activate()
```

Remarks

激活过程：

- 设置激活标志
- 准备数据采集
- 开始工作循环

Deactivate()

停用传感器，使其停止工作

Declaration

```
public virtual void Deactivate()
```

Remarks

停用过程：

- 清除激活标志
- 停止数据采集
- 清理工作状态

GetSensorData()

获取传感器数据（需要在子类中实现）

Declaration

```
public abstract SensorData GetSensorData()
```

Returns

Type	Description
SensorData	包含传感器测量结果的数据对象

Remarks

数据要求：

- 包含最新测量结果
- 附加时间戳信息
- 提供状态标志
- 确保数据有效性

Update(double)

更新传感器状态（需要在子类中实现）

Declaration

```
public abstract void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒

Remarks

更新要求：

- 检查工作状态
- 采集最新数据
- 更新内部状态
- 处理异常情况

Implements

[ISensor](#)

Class MotionAlgorithm

运动算法静态类，提供各种运动计算方法

Inheritance

↳ [object](#)
↳ MotionAlgorithm

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Utils](#)

Assembly: ThreatSource.dll

Syntax

```
public static class MotionAlgorithm
```

Remarks

该类包含了导弹运动相关的各种算法，包括：

- 弹道计算
- 运动学计算
- 制导算法
- 扰动计算 所有方法都是静态的，可以直接调用。

Methods

AddRandomPerturbation(Vector3D)

为向量添加高斯噪声

Declaration

```
public static Vector3D AddRandomPerturbation(Vector3D vector)
```

Parameters

Type	Name	Description
Vector3D	<i>vector</i>	原始向量

Returns

Type	Description
------	-------------

Type	Description
Vector3D	添加高斯噪声后的向量

Remarks

使用Box-Muller变换生成高斯随机数：

- 为向量的每个分量添加独立的高斯噪声
- 噪声强度由标准差控制（默认0.1）
- 用于模拟传感器误差和环境扰动

CalculateBallisticMotion(Vector3D, Vector3D, Vector3D, double)

使用运动学定律计算导弹运动状态

Declaration

```
public static (Vector3D newPosition, Vector3D newVelocity) CalculateBallisticMotion(Vector3D currentPosition, Vector3D currentVelocity, Vector3D acceleration, double deltaTime)
```

Parameters

Type	Name	Description
Vector3D	currentPosition	当前位置坐标
Vector3D	currentVelocity	当前速度向量
Vector3D	acceleration	加速度向量（包含重力加速度）
double	deltaTime	时间步长，单位：秒

Returns

Type	Description
(Vector3D newPosition, Vector3D newVelocity)	包含新位置和新速度的元组

Remarks

该方法适用于无制导状态下的导弹运动计算，使用标准运动学方程：

- 位置更新： $p = p_0 + v_0t + 0.5a*t^2$
- 速度更新： $v = v_0 + a*t$

CalculateBestLaunchOrientation(Vector3D, Vector3D, double)

计算抛物线弹道最佳发射方向（选择较小的仰角）

Declaration

```
public static (Orientation? orientation, Vector3D? velocity) CalculateBestLaunchOrientation(Vector3D startPos, Vector3D targetPos, double initialSpeed)
```

Parameters

Type	Name	Description
Vector3D	startPos	发射位置坐标
Vector3D	targetPos	目标位置坐标
double	initialSpeed	发射初速度，单位：米/秒

Returns

Type	Description
(Orientation? orientation, Vector3D velocity)	包含最佳发射方向和初始速度向量的元组，若无解则返回null

Remarks

该方法使用弹道方程计算两个可能的发射角度，并选择较小的仰角作为最佳发射方向。 计算考虑了重力影响，但未考虑空气阻力。

CalculateLaunchAngles(double, double, double, double)

计算抛物线弹道发射角度

Declaration

```
public static double[]? CalculateLaunchAngles(double v0, double x, double y, double g = 9.81)
```

Parameters

Type	Name	Description
double	<i>v0</i>	初始速度，单位：米/秒
double	<i>x</i>	目标水平距离，单位：米
double	<i>y</i>	目标高度差，单位：米
double	<i>g</i>	重力加速度，默认9.81米/秒²

Returns

Type	Description
double[]	两个可能的发射角度（弧度），如果无解则返回null

Remarks

- 使用标准弹道方程计算发射角度，会返回两个解：
- 一个是低角度解（较小仰角）
 - 一个是高角度解（较大仰角） 如果目标距离超出武器射程，则返回null。

CalculateProportionalNavigation(double, Vector3D, Vector3D, Vector3D, Vector3D)

计算比例导引加速度

Declaration

```
public static Vector3D CalculateProportionalNavigation(double proportionalNavigationCoefficient, Vector3D missilePosition, Vector3D missileVelocity, Vector3D targetPosition, Vector3D targetVelocity)
```

Parameters

Type	Name	Description
double	<i>proportionalNavigationCoefficient</i>	比例导引系数
Vector3D	<i>missilePosition</i>	导弹当前位置
Vector3D	<i>missileVelocity</i>	导弹当前速度
Vector3D	<i>targetPosition</i>	目标当前位置
Vector3D	<i>targetVelocity</i>	目标当前速度

Returns

Type	Description
Vector3D	比例导引产生的加速度向量

Remarks

- 使用比例导引法计算制导加速度：
- 预测目标未来位置
 - 计算视线角速率

- 根据比例导引公式计算所需加速度 加速度方向垂直于导弹速度方向。

RungeKutta4(double, Vector3D, Vector3D, Vector3D)

使用四阶龙格库塔法计算导弹运动状态

Declaration

```
public static (Vector3D newPosition, Vector3D newVelocity) RungeKutta4(double deltaTime,
    Vector3D position, Vector3D velocity, Vector3D acceleration)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒
Vector3D	<i>position</i>	当前位置坐标
Vector3D	<i>velocity</i>	当前速度向量
Vector3D	<i>acceleration</i>	当前加速度向量

Returns

Type	Description
(Vector3D newPosition, Vector3D newVelocity)	包含新位置和新速度的元组

Remarks

四阶龙格库塔法提供了更高精度的数值解：

- 适用于有制导状态下的导弹运动计算
- 考虑了加速度随时间的变化
- 比简单欧拉法具有更高的精度

Class EntityDestroyedEvent

实体销毁事件，表示仿真实体被销毁

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ EntityDestroyedEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class EntityDestroyedEvent : SimulationEvent
```

Remarks

用于通知系统某个实体已被销毁 触发时机：实体被摧毁或删除时

Properties

DestroyedEntityId

获取或设置被销毁实体的ID

Declaration

```
public string? DestroyedEntityId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识已被销毁的实体

[Show / Hide Table of Contents](#)

Class MillimeterWaveJammingEvent

毫米波干扰事件，表示对毫米波雷达的干扰

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [MillimeterWaveJammingEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveJammingEvent : SimulationEvent
```

Remarks

用于模拟对毫米波制导系统的干扰 包含干扰功率信息

Properties

JammingPower

获取或设置干扰功率值

Declaration

```
public double JammingPower { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特 表示毫米波干扰源的输出功率

TargetId

获取或设置被干扰目标的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识受到毫米波干扰的目标实体

Namespace ThreatSource.Missile

Classes

BaseMissile

导弹基类，实现了导弹的基本功能和状态管理

InfraredCommandGuidedMissile

红外指令制导导弹类，实现了红外指令制导的导弹功能

InfraredImagingTerminalGuidedMissile

红外成像末制导导弹类，继承自基础导弹类

LaserBeamRiderMissile

激光波束制导导弹类，实现了激光波束跟踪制导的导弹功能

LaserSemiActiveGuidedMissile

激光半主动制导导弹类，实现了激光半主动寻的制导功能

MillimeterWaveTerminalGuidedMissile

毫米波末制导导弹类，继承自基础导弹类

MissileProperties

导弹配置类，定义了导弹的所有基本属性和性能参数

TerminalSensitiveMissile

末敏导弹类，实现了末端敏感引信的导弹功能

TerminalSensitiveSubmunition

末敏子弹类，实现了多传感器融合的末端制导功能

Structs

MissileRunningState

导弹运行状态信息结构体，包含导弹的完整状态数据

Interfaces

IMissile

导弹接口，定义了导弹的基本行为和状态

Enums

MissileType

导弹类型枚举，定义了系统支持的所有导弹类型

[Show / Hide Table of Contents](#)

Class LaserDesignatorConfig

激光指示器配置类，用于设置激光半主动导引系统中的激光指示器参数

Inheritance

↳ [object](#)

↳ LaserDesignatorConfig

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserDesignatorConfig
```

Remarks

该类定义了激光指示器的关键参数：

- 设备标识
- 空间位置
- 激光功率
- 光束发散角 这些参数影响激光指示器的性能和工作特性

Constructors

LaserDesignatorConfig()

初始化激光指示器配置的新实例

Declaration

```
public LaserDesignatorConfig()
```

Remarks

设置默认值：

- ID为空字符串
- 初始位置为原点(0,0,0)
- 激光功率为0瓦特
- 发散角为0弧度

Properties

Id

获取或设置激光指示器的唯一标识符

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和查找特定的激光指示器

InitialPosition

获取或设置激光指示器的初始位置

Declaration

```
public Vector3D InitialPosition { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

使用三维向量表示空间位置 坐标系：右手坐标系，X向右，Y向上，Z向前

LaserDivergenceAngle

获取或设置激光发散角

Declaration

```
public double LaserDivergenceAngle { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：弧度 影响激光束的扩散特性和照射面积

LaserPower

获取或设置激光功率

Declaration

```
public double LaserPower { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特 影响激光照射的有效距离和目标反射强度

Class RangefinderSensorData

测距仪传感器数据类，包含距离测量的具体数据

Inheritance

↳ [object](#)
↳ [SensorData](#)
↳ RangefinderSensorData

Inherited Members

[SensorData.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class RangefinderSensorData : SensorData
```

Remarks

该类封装了激光测距仪的测量结果：

- 目标距离信息 用于目标定位和距离测量

Properties

Distance

获取或设置测量的距离，单位：米

Declaration

```
public double Distance { get; set; }
```

Property Value

Type	Description
double	

Remarks

Struct Orientation

表示三维空间中的方向，使用欧拉角表示

Inherited Members

[ValueType.Equals\(object\)](#)
[ValueType.GetHashCode\(\)](#)
[object.Equals\(object, object\)](#)
[object.GetType\(\)](#)
[object.ReferenceEquals\(object, object\)](#)

Namespace: [ThreatSource.Utils](#)

Assembly: ThreatSource.dll

Syntax

```
public struct Orientation
```

Remarks

使用欧拉角（偏航角、俯仰角、滚转角）来表示三维空间中的方向：

- 偏航角（Yaw）：绕Y轴旋转的角度
- 俯仰角（Pitch）：绕X轴旋转的角度
- 滚转角（Roll）：绕Z轴旋转的角度 所有角度均使用弧度制

Constructors

Orientation(double, double, double)

初始化方向的新实例

Declaration

```
public Orientation(double yaw, double pitch, double roll)
```

Parameters

Type	Name	Description
double	<i>yaw</i>	偏航角，单位：弧度
double	<i>pitch</i>	俯仰角，单位：弧度
double	<i>roll</i>	滚转角，单位：弧度

Properties

Pitch

获取或设置俯仰角（绕X轴旋转的角度），单位：弧度

Declaration

```
public double Pitch { readonly get; set; }
```

Property Value

Type	Description
double	

Roll

获取或设置滚转角（绕Z轴旋转的角度），单位：弧度

Declaration

```
public double Roll { readonly get; set; }
```

Property Value

Type	Description
double	

Yaw

获取或设置偏航角（绕Y轴旋转的角度），单位：弧度

Declaration

```
public double Yaw { readonly get; set; }
```

Property Value

Type	Description
double	

Methods

FromVector(Vector3D)

从向量创建方向

Declaration

```
public static Orientation FromVector(Vector3D vector)
```

Parameters

Type	Name	Description
Vector3D	<i>vector</i>	输入向量

Returns

Type	Description
Orientation	对应的方向

LookAt(Vector3D)

根据给定的方向向量创建方向

Declaration

```
public static Orientation LookAt(Vector3D direction)
```

Parameters

Type	Name	Description
Vector3D	<i>direction</i>	方向向量

Returns

Type	Description
Orientation	对应的方向

Normalize()

将角度归一化到 $[-\pi, \pi]$ 范围内

Declaration

```
public void Normalize()
```

ToString()

将方向转换为字符串表示

Declaration

```
public override readonly string ToString()
```

Returns

Type	Description
string	方向的字符串表示

Overrides

[ValueType.ToString\(\)](#)

ToVector()

将方向转换为单位向量

Declaration

```
public readonly Vector3D ToVector()
```

Returns

Type	Description
Vector3D	对应的单位向量

[Show / Hide Table of Contents](#)

Class LaserBeamStartEvent

激光波束开始事件，表示激光波束制导开始

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserBeamStartEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class LaserBeamStartEvent : SimulationEvent
```

Remarks

用于激光波束制导系统 触发时机：开始发射制导激光波束时

Properties

LaserBeamRiderId

获取或设置激光波束制导器的ID

Declaration

```
public string? LaserBeamRiderId { get; set; }
```

Property Value

Type	Description
string	

[Show / Hide Table of Contents](#)

Class UltravioletDetectionEvent

紫外探测事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ UltravioletDetectionEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class UltravioletDetectionEvent : SimulationEvent
```

Properties

Intensity

紫外辐射强度

Declaration

```
public double Intensity { get; set; }
```

Property Value

Type	Description
double	

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

威胁源仿真库文档

[Back to top](#)

Class LaserRangefinder

激光测距仪类，实现了目标距离测量和数据采集功能

Inheritance

↳ [object](#)
↳ [Sensor](#)
↳ [LaserRangefinder](#)

Implements

[ISensor](#)

Inherited Members

[Sensor.IsActive](#)
[Sensor.Position](#)
[Sensor.Orientation](#)
[Sensor.Activate\(\)](#)
[Sensor.Deactivate\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserRangefinder : Sensor, ISensor
```

Remarks

该类提供了激光测距仪的核心功能：

- 实时距离测量
- 脉冲频率控制
- 量程范围控制
- 数据采集 用于目标距离的精确测量和跟踪

Constructors

LaserRangefinder(Vector3D, Orientation, double, double)

初始化激光测距仪的新实例

Declaration

```
public LaserRangefinder(Vector3D position, Orientation orientation, double maxRange, double pulseRate)
```

Parameters

Type	Name	Description
Vector3D	<i>position</i>	测距仪的位置坐标
Orientation	<i>orientation</i>	测距仪的朝向
double	<i>maxRange</i>	最大测量距离，单位：米
double	<i>pulseRate</i>	脉冲频率，单位：赫兹

Remarks

构造过程：

- 设置位置和姿态
- 配置测量参数
- 初始化工作状态

Properties

MaxRange

获取或设置最大测量距离，单位：米

Declaration

```
public double MaxRange { get; set; }
```

Property Value

Type	Description
double	

Remarks

定义了测距仪的量程上限 超出此距离的测量可能不准确 受大气条件和目标反射率影响

PulseRate

获取或设置脉冲频率，单位：赫兹

Declaration

```
public double PulseRate { get; set; }
```

Property Value

Type	Description
double	

Remarks

定义了激光发射的重复频率 影响测量的更新速率和精度 需要根据目标运动特性选择

Methods

GetSensorData()

获取激光测距仪的最新数据

Declaration

```
public override SensorData GetSensorData()
```

Returns

Type	Description
SensorData	包含距离测量结果的数据对象

Overrides

[Sensor.GetSensorData\(\)](#)

Remarks

返回数据：

- 目标距离值
- 信号强度
- 测量时间戳
- 数据可靠性

Update(double)

更新激光测距仪的工作状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒

Overrides

[Sensor.Update\(double\)](#)

Remarks

更新过程：

- 发射激光脉冲
- 接收回波信号
- 计算目标距离
- 更新测量数据

Implements

[ISensor](#)

[Show / Hide Table of Contents](#)

Enum IndicatorType

指示器类型枚举，定义了支持的指示器类型

Namespace: [ThreatSource.Indicator](#)

Assembly: ThreatSource.dll

Syntax

```
public enum IndicatorType
```

Remarks

包含三种主要的指示器类型：

- LaserDisintegrator：激光指示器，用于半主动激光制导
- LaserBeamRider：激光波束制导器，用于波束乘波制导
- InfraredTracker：红外跟踪器，用于红外制导 每种类型具有不同的工作原理和性能特点

Fields

Name	Description
InfraredTracker	
LaserBeamRider	
LaserDisintegrator	

Class MillimeterWaveWarnerConfig

毫米波告警器配置类，用于设置毫米波探测和告警系统的参数

Inheritance

↳ [object](#)

↳ MillimeterWaveWarnerConfig

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveWarnerConfig
```

Remarks

该类定义了毫米波告警器的关键参数：

- 设备标识
- 灵敏度阈值
- 警报持续时间
- 探测波长范围 这些参数决定了告警器对毫米波辐射的探测能力

Constructors

MillimeterWaveWarnerConfig()

初始化毫米波告警器配置的新实例

Declaration

```
public MillimeterWaveWarnerConfig()
```

Remarks

设置默认值：

- ID为空字符串
- 灵敏度阈值为0
- 警报持续时间为5秒
- 波长范围为0-0纳米

Properties

AlarmDuration

获取或设置警报持续时间

Declaration

```
public double AlarmDuration { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 定义了每次告警的持续时间

Id

获取或设置毫米波告警器的唯一标识符

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和查找特定的告警器

SensitivityThreshold

获取或设置告警灵敏度阈值

Declaration

```
public double SensitivityThreshold { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特/平方米 当接收到的毫米波辐射强度超过此阈值时触发告警

WavelengthMax

获取或设置最大探测波长

Declaration

```
public double WavelengthMax { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最长毫米波波长

WavelengthMin

获取或设置最小探测波长

Declaration

```
public double WavelengthMin { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最短毫米波波长

[Show / Hide Table of Contents](#)

Class LaserSemiActiveGuidanceSystem

激光半主动制导系统类，实现了基于激光照射的目标跟踪和制导功能

Inheritance

↳ [object](#)
↳ [BasicGuidanceSystem](#)
↳ [LaserSemiActiveGuidanceSystem](#)

Implements

[IGuidanceSystem](#)

Inherited Members

[BasicGuidanceSystem.HasGuidance](#)
[BasicGuidanceSystem.MaxAcceleration](#)
[BasicGuidanceSystem.ProportionalNavigationCoefficient](#)
[BasicGuidanceSystem.Position](#)
[BasicGuidanceSystem.Velocity](#)
[BasicGuidanceSystem.GuidanceAcceleration](#)
[BasicGuidanceSystem.GetGuidanceAcceleration\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Guidance](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserSemiActiveGuidanceSystem : BasicGuidanceSystem, IGuidanceSystem
```

Remarks

该类提供了激光半主动制导系统的核心功能：

- 激光目标照射
- 反射光探测
- 信号处理
- 比例导引控制 用于实现精确制导打击

Constructors

LaserSemiActiveGuidanceSystem(double, double)

初始化激光半主动制导系统的新实例

Declaration

```
public LaserSemiActiveGuidanceSystem(double maxAcceleration, double guidanceCoefficient)
```

Parameters

Type	Name	Description
double	<i>maxAcceleration</i>	最大加速度，单位：米/平方秒
double	<i>guidanceCoefficient</i>	制导系数

Remarks

构造过程：

- 初始化基类参数
- 初始化目标信息
- 初始化激光参数

Methods

CalculateGuidanceAcceleration(double)

计算制导加速度

Declaration

```
protected override void CalculateGuidanceAcceleration(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Overrides

[BasicGuidanceSystem.CalculateGuidanceAcceleration\(double\)](#)

Remarks

计算过程：

- 使用比例导引算法
- 考虑目标运动
- 限制最大加速度

DeactivateLaserDesignator()

关闭激光照射系统

Declaration

```
public void DeactivateLaserDesignator()
```

Remarks

关闭过程：

- 停止激光照射
- 清除位置信息
- 清除目标信息
- 清除激光参数

GetStatus()

获取制导系统的详细状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含完整状态参数的字符串

Overrides

BasicGuidanceSystem.GetStatus()

Remarks

返回信息：

- 基本状态信息
- 接收功率数据
- 锁定阈值 用于系统监控和调试

Update(double, Vector3D, Vector3D)

更新制导系统的状态和计算结果

Declaration

```
public override void Update(double deltaTime, Vector3D missilePosition, Vector3D missileVelocity)
```

Parameters

Type	Name	Description
double	deltaTime	时间步长，单位：秒
Vector3D	missilePosition	导弹位置，单位：米
Vector3D	missileVelocity	导弹速度，单位：米/秒

Overrides

BasicGuidanceSystem.Update(double, Vector3D, Vector3D)

Remarks

更新过程：

- 检查激光照射状态
- 计算接收功率
- 判断是否锁定目标
- 计算制导指令

UpdateLaserDesignator(Vector3D, Vector3D, Vector3D, double, double)

更新激光指示器参数

Declaration

```
public void UpdateLaserDesignator(Vector3D sourcePosition, Vector3D targetPosition, Vector3D targetVelocity, double laserPower, double laserDivergenceAngle)
```

Parameters

Type	Name	Description
Vector3D	sourcePosition	激光源位置，单位：米
Vector3D	targetPosition	目标位置，单位：米
Vector3D	targetVelocity	目标速度，单位：米/秒

Type	Name	Description
double	<i>laserPower</i>	激光功率，单位：瓦特
double	<i>laserDivergenceAngle</i>	激光发散角，单位：弧度

Remarks

更新过程：

- 激活激光照射
- 更新位置信息
- 更新目标信息
- 更新激光参数

Implements

[IGuidanceSystem](#)

[Show / Hide Table of Contents](#)

Enum IndicatorState

指示器状态枚举，定义了指示器的工作状态

Namespace: [ThreatSource.Indicator](#)

Assembly: ThreatSource.dll

Syntax

```
public enum IndicatorState
```

Remarks

包含两种基本状态：

- Indicating：指示状态，正在执行指示或跟踪任务
- Idle：空闲状态，等待新的指示任务 用于控制指示器的工作流程

Fields

Name	Description
Idle	
Indicating	

Class Vector3D

表示三维空间中的向量，提供基本的向量运算功能

Inheritance

↳ [object](#)
↳ Vector3D

Inherited Members

[object.Equals\(object, object\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)

Namespace: [ThreatSource.Utils](#)

Assembly: ThreatSource.dll

Syntax

```
public class Vector3D
```

Remarks

该类实现了三维向量的基本运算，包括：

- 向量的加减乘除运算
- 向量的点积和叉积
- 向量的归一化
- 向量的旋转变换 所有计算都使用双精度浮点数（double）以保证精度

Constructors

Vector3D(double, double, double)

初始化三维向量的新实例

Declaration

```
public Vector3D(double x, double y, double z)
```

Parameters

Type	Name	Description
double	<i>x</i>	X分量的值
double	<i>y</i>	Y分量的值
double	<i>z</i>	Z分量的值

Properties

UnitX

X轴单位向量 (1, 0, 0)

Declaration

```
public static Vector3D UnitX { get; }
```

Property Value

Type	Description
Vector3D	

UnitY

Y轴单位向量 (0, 1, 0)

Declaration

```
public static Vector3D UnitY { get; }
```

Property Value

Type	Description
Vector3D	

UnitZ

Z轴单位向量 (0, 0, 1)

Declaration

```
public static Vector3D UnitZ { get; }
```

Property Value

Type	Description
Vector3D	

X

获取或设置向量的X分量

Declaration

```
public double X { get; set; }
```

Property Value

Type	Description
double	

Y

获取或设置向量的Y分量

Declaration

```
public double Y { get; set; }
```

Property Value

Type	Description
double	

Z

获取或设置向量的Z分量

Declaration

```
public double Z { get; set; }
```

Property Value

Type	Description
double	

Zero

零向量 (0, 0, 0)

Declaration

```
public static Vector3D Zero { get; }
```

Property Value

Type	Description
Vector3D	

Methods

CrossProduct(Vector3D, Vector3D)

计算两个向量的叉积

Declaration

```
public static Vector3D CrossProduct(Vector3D a, Vector3D b)
```

Parameters

Type	Name	Description
Vector3D	<i>a</i>	第一个向量
Vector3D	<i>b</i>	第二个向量

Returns

Type	Description
Vector3D	叉积结果

Distance(Vector3D, Vector3D)

计算两个向量之间的距离

Declaration

```
public static double Distance(Vector3D v1, Vector3D v2)
```

Parameters

Type	Name	Description
Vector3D	v1	第一个向量
Vector3D	v2	第二个向量

Returns

Type	Description
double	两个向量之间的距离

DotProduct(Vector3D, Vector3D)

计算两个向量的点积

Declaration

```
public static double DotProduct(Vector3D a, Vector3D b)
```

Parameters

Type	Name	Description
Vector3D	a	第一个向量
Vector3D	b	第二个向量

Returns

Type	Description
double	点积结果

Equals(object?)

判断两个向量是否相等

Declaration

```
public override bool Equals(object? obj)
```

Parameters

Type	Name	Description
object	obj	

Returns

Type	Description
bool	

Overrides

object.Equals(object)

GetHashCode()

获取向量的哈希码

Declaration

```
public override int GetHashCode()
```

Returns

Type	Description
int	向量的哈希码

Overrides

[object.GetHashCode\(\)](#)

Magnitude()

计算向量的模长

Declaration

```
public double Magnitude()
```

Returns

Type	Description
double	向量的模长

MagnitudeSquared()

计算向量模长的平方

Declaration

```
public double MagnitudeSquared()
```

Returns

Type	Description
double	向量模长的平方

Negate(Vector3D)

向量取反

Declaration

```
public static Vector3D Negate(Vector3D a)
```

Parameters

Type	Name	Description
Vector3D	<i>a</i>	输入向量

Returns

Type	Description
Vector3D	取反后的向量

Normalize()

向量归一化

Declaration

```
public Vector3D Normalize()
```

Returns

Type	Description
Vector3D	归一化后的向量

PointOnLine(Vector3D, Vector3D, double)

计算AB两点之间距离 A 点一定距离的点的坐标

Declaration

```
public static Vector3D PointOnLine(Vector3D a, Vector3D b, double distance)
```

Parameters

Type	Name	Description
Vector3D	<i>a</i>	直线起点
Vector3D	<i>b</i>	直线终点
double	<i>distance</i>	距离起点距离

Returns

Type	Description
Vector3D	直线上的点

ToString()

将向量转换为字符串表示

Declaration

```
public override string ToString()
```

Returns

Type	Description
string	向量的字符串表示

Overrides

[object.ToString\(\)](#)

Operators

operator +(Vector3D, Vector3D)

向量加法运算符重载

Declaration

```
public static Vector3D operator +(Vector3D a, Vector3D b)
```

Parameters

Type	Name	Description
------	------	-------------

Type	Name	Description
Vector3D	<i>a</i>	
Vector3D	<i>b</i>	

Returns

Type	Description
Vector3D	

operator /(Vector3D, double)

向量与标量除法运算符重载

Declaration

```
public static Vector3D operator /(Vector3D a, double scalar)
```

Parameters

Type	Name	Description
Vector3D	<i>a</i>	
double	<i>scalar</i>	

Returns

Type	Description
Vector3D	

operator ==(Vector3D, Vector3D)

向量相等运算符重载

Declaration

```
public static bool operator ==(Vector3D left, Vector3D right)
```

Parameters

Type	Name	Description
Vector3D	<i>left</i>	
Vector3D	<i>right</i>	

Returns

Type	Description
bool	

operator !=(Vector3D, Vector3D)

向量不相等运算符重载

Declaration

```
public static bool operator !=(Vector3D left, Vector3D right)
```

Parameters

Type	Name	Description
------	------	-------------

Type	Name	Description
Vector3D	<i>left</i>	
Vector3D	<i>right</i>	

Returns

Type	Description
bool	

operator *(Vector3D, double)

向量与标量乘法运算符重载

Declaration

```
public static Vector3D operator *(Vector3D a, double scalar)
```

Parameters

Type	Name	Description
Vector3D	<i>a</i>	
double	<i>scalar</i>	

Returns

Type	Description
Vector3D	

operator -(Vector3D, Vector3D)

向量减法运算符重载

Declaration

```
public static Vector3D operator -(Vector3D a, Vector3D b)
```

Parameters

Type	Name	Description
Vector3D	<i>a</i>	
Vector3D	<i>b</i>	

Returns

Type	Description
Vector3D	

operator -(Vector3D)

向量反向

Declaration

```
public static Vector3D operator -(Vector3D a)
```

Parameters

Type	Name	Description
------	------	-------------

Type	Name	Description
Vector3D	<i>a</i>	

Returns

Type	Description
Vector3D	

Class InfraredWarnerConfig

红外告警器配置类，用于设置红外探测和告警系统的参数

Inheritance

↳ [object](#)
↳ InfraredWarnerConfig

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredWarnerConfig
```

Remarks

该类定义了红外告警器的关键参数：

- 设备标识
- 灵敏度阈值
- 警报持续时间
- 探测波长范围 这些参数决定了告警器对红外辐射的探测能力

Constructors

InfraredWarnerConfig()

初始化红外告警器配置的新实例

Declaration

```
public InfraredWarnerConfig()
```

Remarks

设置默认值：

- ID为空字符串
- 灵敏度阈值为0
- 警报持续时间为5秒
- 波长范围为0-0纳米

Properties

AlarmDuration

获取或设置警报持续时间

Declaration

```
public double AlarmDuration { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 定义了每次告警的持续时间

Id

获取或设置红外告警器的唯一标识符

Declaration

```
public string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和查找特定的告警器

SensitivityThreshold

获取或设置告警灵敏度阈值

Declaration

```
public double SensitivityThreshold { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特/平方米 当接收到的红外辐射强度超过此阈值时触发告警

WavelengthMax

获取或设置最大探测波长

Declaration

```
public double WavelengthMax { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最长红外波长

WavelengthMin

获取或设置最小探测波长

Declaration

```
public double WavelengthMin { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：纳米 定义了告警器可以探测的最短红外波长

Class EntityActivatedEvent

实体激活事件，表示仿真实体被激活

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [EntityActivatedEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class EntityActivatedEvent : SimulationEvent
```

Remarks

用于通知系统某个实体已进入活动状态 触发时机：实体被创建或重新激活时

Properties

ActivatedEntityId

获取或设置被激活实体的ID

Declaration

```
public string? ActivatedEntityId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识已被激活的实体

Class MillimeterWaveRadiometer

毫米波辐射计类，实现了目标辐射温度测量和目标识别功能

Inheritance

↳ [object](#)
↳ [Sensor](#)
↳ [MillimeterWaveRadiometer](#)

Implements

[ISensor](#)

Inherited Members

[Sensor.IsActive](#)
[Sensor.Position](#)
[Sensor.Orientation](#)
[Sensor.Activate\(\)](#)
[Sensor.Deactivate\(\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveRadiometer : Sensor, ISensor
```

Remarks

该类提供了毫米波辐射计的核心功能：

- 辐射温度测量
- 温差阈值检测
- 目标识别判断
- 实时状态更新 用于末敏子弹的目标探测和识别

Constructors

MillimeterWaveRadiometer(TerminalSensitiveSubmunition, double, double)

初始化毫米波辐射计的新实例

Declaration

```
public MillimeterWaveRadiometer(TerminalSensitiveSubmunition submunition, double wavelength, double fieldOfView)
```

Parameters

Type	Name	Description
TerminalSensitiveSubmunition	<i>submunition</i>	末敏子弹实例
double	<i>wavelength</i>	工作波段，单位：毫米
double	<i>fieldOfView</i>	视场角，单位：度

Remarks

构造过程：

- 设置工作参数
- 初始化温度记录
- 创建传感器数据
- 继承基类位置和姿态

Properties

DetectionTemperatureDifferenceThreshold

获取或设置辐射温度差检测阈值，单位：开尔文

Declaration

```
public double DetectionTemperatureDifferenceThreshold { get; set; }
```

Property Value

Type	Description
double	

Remarks

辐射计高温物体与低温物体的检测温差 默认值为100K 典型温差值：

- 金属目标与草地：170~230K
- 金属目标与砂石地：120~150K 辐射率说明：
- 草地辐射率：1.0
- 砂石地辐射率：0.83
- 金属辐射率：0 坦克辐射温度等于天空背景辐射的反射温度，天顶角30度时约为90K

Wavelength

获取或设置工作波段，单位：毫米

Declaration

```
public double Wavelength { get; set; }
```

Property Value

Type	Description
double	

Remarks

可选波段：3mm 或 8mm 影响辐射计的探测性能和大气衰减

Methods

GetSensorData()

获取毫米波辐射计的最新数据

Declaration

```
public override SensorData GetSensorData()
```

Returns

Type	Description
SensorData	包含探测结果的数据对象

Overrides

[Sensor.GetSensorData\(\)](#)

Remarks

返回数据：

- 目标探测状态
- 温度测量结果
- 时间戳信息

Update(double)

更新毫米波辐射计的工作状态

Declaration

```
public override void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒

Overrides

[Sensor.Update\(double\)](#)

Remarks

更新过程：

- 检查激活状态
- 测量辐射温度
- 计算温度差值
- 判断目标存在
- 更新历史温度

Implements

[ISensor](#)

[Show / Hide Table of Contents](#)

Class LaserIlluminationStartEvent

激光照射开始事件，表示激光定位器开始照射目标

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserIlluminationStartEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserIlluminationStartEvent : SimulationEvent
```

Remarks

用于激光半主动导引系统 触发时机：激光定位器开始照射目标时

Properties

LaserDesignatorId

获取或设置激光定位器的ID

Declaration

```
public string? LaserDesignatorId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识发出激光的定位器设备

TargetId

获取或设置被照射目标的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识被激光照射的目标实体

Class TankRadiationEvent

坦克辐射事件，表示坦克的多波段辐射特性

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ TankRadiationEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class TankRadiationEvent : SimulationEvent
```

Remarks

用于模拟坦克在不同波段的辐射特征 包含激光、毫米波和红外波段的信息

Properties

InfraredRadiationIntensity

获取或设置红外辐射强度

Declaration

```
public double InfraredRadiationIntensity { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特/平方米 表示坦克的红外辐射强度

LaserReflectionIntensity

获取或设置激光漫反射强度

Declaration

```
public double LaserReflectionIntensity { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：相对单位 表示坦克表面对激光的反射能力

MillimeterWaveRadiationTemperature

获取或设置毫米波辐射温度

Declaration

```
public double MillimeterWaveRadiationTemperature { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：开尔文 表示坦克在毫米波波段的辐射温度

MillimeterWaveReflectionIntensity

获取或设置毫米波反射强度

Declaration

```
public double MillimeterWaveReflectionIntensity { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：相对单位 表示坦克对毫米波雷达的反射特性

Class InfraredJammingEvent

红外干扰事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ InfraredJammingEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredJammingEvent : SimulationEvent
```

Properties

JammingPower

干扰功率(瓦特)

Declaration

```
public double JammingPower { get; set; }
```

Property Value

Type	Description
double	

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

WavelengthMax

最大波长(微米)

Declaration

```
public double WavelengthMax { get; set; }
```

Property Value

Type	Description
double	

WavelengthMin

最小波长(微米)

Declaration

```
public double WavelengthMin { get; set; }
```

Property Value

Type	Description
double	

Class InfraredSensorData

红外传感器数据类，包含红外探测的具体数据

Inheritance

↳ [object](#)
↳ [SensorData](#)
↳ InfraredSensorData

Inherited Members

[SensorData.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredSensorData : SensorData
```

Remarks

该类封装了红外传感器的测量结果：

- 目标温度信息
- 目标探测状态 用于红外目标的探测和跟踪

Properties

IsTargetDetected

获取或设置是否检测到目标

Declaration

```
public bool IsTargetDetected { get; set; }
```

Property Value

Type	Description
bool	

Remarks

true表示探测到有效目标 false表示未探测到目标 用于目标存在性判断

Temperature

获取或设置探测到的温度，单位：开尔文

Declaration

```
public double Temperature { get; set; }
```

Property Value

Type	Description
double	

Remarks

记录目标的红外辐射温度 用于目标特征识别和状态评估

Struct MissileRunningState

导弹运行状态信息结构体，包含导弹的完整状态数据

Inherited Members

[ValueType.Equals\(object\)](#)
[ValueType.GetHashCode\(\)](#)
[ValueType.ToString\(\)](#)
[object.Equals\(object, object\)](#)
[object.GetType\(\)](#)
[object.ReferenceEquals\(object, object\)](#)

Namespace: [ThreatSource.Missile](#)

Assembly: ThreatSource.dll

Syntax

```
public struct MissileRunningState
```

Remarks

该结构体用于：

- 状态监控和记录
- 数据分析和诊断
- 仿真状态同步
- 导弹行为控制

Properties

EngineBurnTime

获取或设置发动机的当前燃烧时间

Declaration

```
public double EngineBurnTime { readonly get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 记录发动机工作的累计时间 用于控制推力变化和燃料消耗

FlightDistance

获取或设置导弹的当前飞行距离

Declaration

```
public double FlightDistance { readonly get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米 从发射点到当前位置的累计飞行距离

FlightTime

获取或设置导弹的当前飞行时间

Declaration

```
public double FlightTime { readonly get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 从发射时刻开始计时

Id

获取或设置导弹的唯一标识符

Declaration

```
public string Id { readonly get; set; }
```

Property Value

Type	Description
string	

Remarks

在仿真系统中必须唯一 用于标识和追踪特定的导弹

IsActive

获取或设置导弹是否处于活动状态

Declaration

```
public bool IsActive { readonly get; set; }
```

Property Value

Type	Description
bool	

Remarks

true表示导弹正常工作 false表示导弹已停止工作（销毁或自毁）

IsGuidance

获取或设置导弹是否处于制导状态

Declaration

```
public bool IsGuidance { readonly get; set; }
```

Property Value

Type	Description
bool	

Remarks

true表示导弹当前在制导 false表示导弹处于非制导状态

LostGuidanceTime

获取或设置导弹失去制导的持续时间

Declaration

```
public double LostGuidanceTime { readonly get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：秒 记录导弹处于非制导状态的累计时间 用于判断是否需要触发自毁

Orientation

获取或设置导弹的当前朝向

Declaration

```
public Orientation Orientation { readonly get; set; }
```

Property Value

Type	Description
Orientation	

Remarks

使用欧拉角表示导弹的姿态 包含偏航角、俯仰角和滚转角

Position

获取或设置导弹的当前位置

Declaration

```
public Vector3D Position { readonly get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

使用三维向量表示空间位置 坐标系：右手坐标系，X向右，Y向上，Z向前

Speed

获取或设置导弹的当前速度大小

Declaration

```
public double Speed { readonly get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米/秒 表示导弹运动的标量速度

Type

获取或设置导弹的类型

Declaration

```
public MissileType Type { readonly get; set; }
```

Property Value

Type	Description
MissileType	

Remarks

用于区分不同种类的导弹 影响导弹的行为特征和性能参数

Velocity

获取或设置导弹的当前速度向量

Declaration

```
public Vector3D Velocity { readonly get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

包含速度的大小和方向信息 单位：米/秒

[Show / Hide Table of Contents](#)

Class SimulationElement

仿真元素的抽象基类，所有仿真中的实体都继承自此类

Inheritance

↳ [object](#)

- ↳ [SimulationElement](#)
 - ↳ [InfraredTracker](#)
 - ↳ [LaserBeamRider](#)
 - ↳ [LaserDesignator](#)
 - ↳ [BaseMissile](#)
 - ↳ [Tank](#)

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public abstract class SimulationElement
```

Remarks

该类提供了仿真实体的基本功能：

- 位置和运动状态管理
- 生命周期管理（激活/停用）
- 事件发布机制
- 状态更新机制 所有具体的仿真实体（如导弹、目标等）都应继承此类并实现其抽象方法。

Constructors

SimulationElement(string, Vector3D, Orientation, double, ISimulationManager)

初始化仿真元素的新实例

Declaration

```
protected SimulationElement(string id, Vector3D position, Orientation orientation, double speed, ISimulationManager simulationManager)
```

Parameters

Type	Name	Description
string	<i>id</i>	仿真元素的唯一标识符
Vector3D	<i>position</i>	初始位置坐标
Orientation	<i>orientation</i>	初始朝向角度
double	<i>speed</i>	初始速度大小，单位：米/秒
ISimulationManager	<i>simulationManager</i>	仿真管理器实例

Remarks

构造函数设置实体的初始状态：

- 根据ID、位置、朝向和速度初始化实体
- 计算初始速度向量
- 设置初始状态为未激活

Properties

Id

获取或设置仿真元素的唯一标识符

Declaration

```
public virtual string Id { get; set; }
```

Property Value

Type	Description
string	

Remarks

ID在仿真系统中必须唯一，用于标识和查找实体

IsActive

获取仿真元素是否处于活动状态

Declaration

```
public virtual bool IsActive { get; protected set; }
```

Property Value

Type	Description
bool	

Remarks

true表示实体当前处于活动状态，可以参与仿真 false表示实体已停用，不再参与仿真计算

Orientation

获取或设置仿真元素的当前朝向

Declaration

```
public virtual Orientation Orientation { get; set; }
```

Property Value

Type	Description
Orientation	

Remarks

使用欧拉角表示实体的朝向 包含偏航角、俯仰角和滚转角

Position

获取或设置仿真元素的当前位置

Declaration

```
public virtual Vector3D Position { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

使用三维向量表示实体在空间中的位置 坐标系：右手坐标系，X向右，Y向上，Z向前

SimulationManager

获取或设置仿真管理器的引用

Declaration

```
protected ISimulationManager SimulationManager { get; set; }
```

Property Value

Type	Description
ISimulationManager	

Remarks

用于访问仿真系统的核心功能，如事件发布、实体查询等 在构造函数中初始化，整个生命周期内保持不变

Speed

获取或设置仿真元素的当前速度大小

Declaration

```
public virtual double Speed { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：米/秒 此属性仅表示速度的标量值，方向信息需要结合Velocity属性

Velocity

获取或设置仿真元素的当前速度向量

Declaration


```
public virtual Vector3D Velocity { get; set; }
```

Property Value

Type	Description
Vector3D	

Remarks

单位：米/秒 包含速度的大小和方向信息 向量的模等于Speed属性值

Methods

Activate()

激活仿真元素，使其参与仿真

Declaration

```
public virtual void Activate()
```

Remarks

激活操作会：

- 将IsActive设置为true
- 发布实体激活事件
- 允许实体参与仿真计算

Deactivate()

停用仿真元素，将其从仿真中移除

Declaration

```
public virtual void Deactivate()
```

Remarks

停用操作会：

- 将IsActive设置为false
- 发布实体停用事件
- 停止实体参与仿真计算

GetStatus()

获取仿真元素的状态信息

Declaration

```
public virtual string GetStatus()
```

Returns

Type	Description
string	包含实体当前状态的字符串描述

Remarks

返回的字符串包含：

- 实体类型

- 实体ID
- 当前位置
- 当前朝向
- 活动状态

PublishEvent(SimulationEvent)

发布仿真事件到事件系统

Declaration

```
protected void PublishEvent(SimulationEvent evt)
```

Parameters

Type	Name	Description
SimulationEvent	<i>evt</i>	要发布的事件实例

Remarks

在发布事件前会：

- 设置事件的发送者ID
- 添加时间戳
- 通过仿真管理器发布到事件系统

Update(double)

更新仿真元素的状态

Declaration

```
public abstract void Update(double deltaTime)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	时间步长，单位：秒

Remarks

抽象方法，需要在派生类中实现具体的更新逻辑：

- 更新位置和朝向
- 计算新的速度和加速度
- 处理碰撞和其他物理效果
- 触发相关事件

[Show / Hide Table of Contents](#)

Class TargetDestroyedEvent

目标被摧毁事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [TargetDestroyedEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class TargetDestroyedEvent : SimulationEvent
```

Properties

TargetId

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Class LaserIlluminationUpdateEvent

激光照射更新事件，表示激光照射状态的更新

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserIlluminationUpdateEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class LaserIlluminationUpdateEvent : SimulationEvent
```

Remarks

用于实时更新激光照射的状态 在照射过程中周期性触发

Properties

LaserDesignatorId

获取或设置激光定位器的ID

Declaration

```
public string? LaserDesignatorId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识正在照射的定位器设备

TargetId

获取或设置被照射目标的ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Remarks

标识正在被照射的目标实体

Class RadiometerSensorData

辐射计传感器数据类，包含毫米波辐射探测的具体数据

Inheritance

↳ [object](#)
↳ [SensorData](#)
↳ RadiometerSensorData

Inherited Members

[SensorData.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Sensor](#)

Assembly: ThreatSource.dll

Syntax

```
public class RadiometerSensorData : SensorData
```

Remarks

该类封装了毫米波辐射计的测量结果：

- 辐射强度信息
- 目标探测状态 用于目标特征识别和态势评估

Properties

IsTargetDetected

获取或设置是否检测到目标

Declaration

```
public bool IsTargetDetected { get; set; }
```

Property Value

Type	Description
bool	

Remarks

true表示探测到有效目标 false表示未探测到目标 用于目标存在性判断

RadiationIntensity

获取或设置探测到的辐射强度，单位： 瓦特/平方米

Declaration

```
public double RadiationIntensity { get; set; }
```

Property Value

Type	Description
double	

Remarks

记录目标的毫米波辐射强度 用于目标特征分析和识别判断

[Show / Hide Table of Contents](#)

Interface IGuidanceSystem

制导系统接口，定义了所有制导系统的通用功能

Namespace: [ThreatSource.Guidance](#)

Assembly: ThreatSource.dll

Syntax

```
public interface IGuidanceSystem
```

Remarks

该接口提供了制导系统的基本功能规范：

- 制导状态判断
- 制导信息更新
- 制导加速度计算 是所有具体制导系统实现的基础

Properties

HasGuidance

获取是否有有效的制导信息

Declaration

```
bool HasGuidance { get; }
```

Property Value

Type	Description
bool	

Remarks

true表示当前有可用的制导信息 false表示无法获得有效制导 用于判断制导系统的工作状态

Methods

GetGuidanceAcceleration()

获取制导加速度指令

Declaration

```
Vector3D GetGuidanceAcceleration()
```


Returns

Type	Description
Vector3D	三维制导加速度向量，单位：米/平方秒

Remarks

返回数据：

- X分量：横向制导加速度
- Y分量：垂直制导加速度
- Z分量：纵向制导加速度 用于导弹的轨迹控制

Update(double, Vector3D, Vector3D)

更新制导系统的状态和计算结果

Declaration

```
void Update(double deltaTime, Vector3D missilePosition, Vector3D missileVelocity)
```

Parameters

Type	Name	Description
double	<i>deltaTime</i>	自上次更新以来的时间间隔，单位：秒
Vector3D	<i>missilePosition</i>	导弹当前位置，单位：米
Vector3D	<i>missileVelocity</i>	导弹当前速度，单位：米/秒

Remarks

更新过程：

- 获取最新目标信息
- 计算制导参数
- 更新制导状态
- 生成制导指令

Class TestSimulationAdapter

用于测试的仿真环境适配器

Inheritance

↳ [object](#)

↳ TestSimulationAdapter

Implements

[ISimulationAdapter](#)

Inherited Members

[object.Equals\(object\)](#)

[object.Equals\(object, object\)](#)

[object.GetHashCode\(\)](#)

[object.GetType\(\)](#)

[object.MemberwiseClone\(\)](#)

[object.ReferenceEquals\(object, object\)](#)

[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation.Testing](#)

Assembly: ThreatSource.dll

Syntax

```
public class TestSimulationAdapter : ISimulationAdapter
```

Constructors

TestSimulationAdapter(ISimulationManager)

Declaration

```
public TestSimulationAdapter(ISimulationManager simulationManager)
```

Parameters

Type	Name	Description
ISimulationManager	<i>simulationManager</i>	

Methods

AddTestEntity(string, object)

Declaration

```
public void AddTestEntity(string id, object entity)
```

Parameters

Type	Name	Description
string	<i>id</i>	
object	<i>entity</i>	

ClearEvents()

Declaration

```
public void ClearEvents()
```

ClearTestEntities()

Declaration

```
public void ClearTestEntities()
```

GetEntity(string)

从外部仿真环境获取实体

Declaration

```
public object? GetEntity(string id)
```

Parameters

Type	Name	Description
string	<i>id</i>	实体ID

Returns

Type	Description
object	实体对象，如果不存在则返回null

GetPublishedEvents()

Declaration

```
public IReadOnlyList<object> GetPublishedEvents()
```

Returns

Type	Description
IReadOnlyList<object>	

GetReceivedEvents()

Declaration

```
public IReadOnlyList<object> GetReceivedEvents()
```

Returns

Type	Description
IReadOnlyList<object>	

PublishToExternalSimulation<T>(T)

发布事件到第三方仿真环境

Declaration

```
public void PublishToExternalSimulation<T>(T evt)
```

Parameters

Type	Name	Description
T	<i>evt</i>	要发布的事件

Type Parameters

Name	Description
<i>T</i>	事件类型

Remarks

用于将威胁源仿真库中的事件同步到外部仿真环境

ReceiveFromExternalSimulation<T>(T)

从第三方仿真环境接收事件

Declaration

```
public void ReceiveFromExternalSimulation<T>(T evt)
```

Parameters

Type	Name	Description
T	<i>evt</i>	接收到的事件

Type Parameters

Name	Description
<i>T</i>	事件类型

Remarks

用于接收外部仿真环境的事件并在威胁源仿真库中处理

Implements

[ISimulationAdapter](#)

Class MillimeterWaveGuidanceSystem

毫米波导引头系统类，实现了基于毫米波雷达的目标探测和跟踪功能

Inheritance

↳ [object](#)
↳ [BasicGuidanceSystem](#)
↳ [MillimeterWaveGuidanceSystem](#)

Implements

[IGuidanceSystem](#)

Inherited Members

[BasicGuidanceSystem.HasGuidance](#)
[BasicGuidanceSystem.MaxAcceleration](#)
[BasicGuidanceSystem.ProportionalNavigationCoefficient](#)
[BasicGuidanceSystem.Position](#)
[BasicGuidanceSystem.Velocity](#)
[BasicGuidanceSystem.GuidanceAcceleration](#)
[BasicGuidanceSystem.GetGuidanceAcceleration\(\)](#)
[BasicGuidanceSystem.CalculateGuidanceAcceleration\(double\)](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Guidance](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveGuidanceSystem : BasicGuidanceSystem, IGuidanceSystem
```

Remarks

该类提供了毫米波导引头系统的核心功能：

- 目标探测和识别
- 信噪比计算
- 抗干扰处理
- 比例导引控制 用于实现全天候制导打击

Constructors

MillimeterWaveGuidanceSystem(double, double, ISimulationManager)

初始化毫米波导引头系统的新实例

Declaration

```
public MillimeterWaveGuidanceSystem(double maxAcceleration, double guidanceCoefficient, I
SimulationManager simulationManager)
```

Parameters

Type	Name	Description
double	maxAcceleration	最大加速度，单位：米/平方秒
double	guidanceCoefficient	制导系数
ISimulationManager	simulationManager	仿真管理器实例

Remarks

构造过程：

- 初始化基类参数
- 设置仿真管理器
- 初始化目标位置
- 注册干扰事件处理

Properties

SimulationManager

获取或设置仿真管理器实例

Declaration

```
public ISimulationManager SimulationManager { get; set; }
```

Property Value

Type	Description
ISimulationManager	

Remarks

用于获取场景中的目标信息 实现目标探测功能

Methods

GetStatus()

获取制导系统的详细状态信息

Declaration

```
public override string GetStatus()
```

Returns

Type	Description
string	包含完整状态参数的字符串

Overrides

BasicGuidanceSystem.GetStatus()

Remarks

返回信息：

- 基本状态信息

- 目标位置 用于系统监控和调试

Update(double, Vector3D, Vector3D)

更新制导系统的状态和计算结果

Declaration

```
public override void Update(double deltaTime, Vector3D missilePosition, Vector3D missileVelocity)
```

Parameters

Type	Name	Description
double	deltaTime	自上次更新以来的时间间隔，单位：秒
Vector3D	missilePosition	导弹当前位置，单位：米
Vector3D	missileVelocity	导弹当前速度，单位：米/秒

Overrides

[BasicGuidanceSystem.Update\(double, Vector3D, Vector3D\)](#)

Remarks

更新过程：

- 探测目标位置
- 计算目标速度
- 生成制导指令
- 限制最大加速度

Implements

[IGuidanceSystem](#)

Class LaserBeamStopEvent

激光波束停止事件，表示激光波束制导结束

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ [LaserBeamStopEvent](#)

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: [ThreatSource.dll](#)

Syntax

```
public class LaserBeamStopEvent : SimulationEvent
```

Remarks

用于通知系统激光波束制导已结束 触发时机：停止发射制导激光波束时

Properties

LaserBeamRiderId

获取或设置激光波束制导器的ID

Declaration

```
public string? LaserBeamRiderId { get; set; }
```

Property Value

Type	Description
string	



[Show / Hide Table of Contents](#)

Namespace ThreatSource.Simulation.Testing

Classes

TestSimulationAdapter

用于测试的仿真环境适配器

[Show / Hide Table of Contents](#)

Class UltravioletWarnerAlarmEvent

紫外告警器警报事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ UltravioletWarnerAlarmEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class UltravioletWarnerAlarmEvent : SimulationEvent
```

Properties

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

[Show / Hide Table of Contents](#)

Class InfraredGuidanceMissileLightEvent

红外指令制导导弹点亮红外热源事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ InfraredGuidanceMissileLightEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class InfraredGuidanceMissileLightEvent : SimulationEvent
```

Remarks

用于模拟导弹尾部红外热源的激活 触发时机：导弹发动机点火时

Properties

RadiationPower

获取或设置红外热源辐射功率

Declaration

```
public double RadiationPower { get; set; }
```

Property Value

Type	Description
double	

Remarks

单位：瓦特 表示导弹尾焰的红外辐射强度

Class SimulationManager

仿真管理器的默认实现，提供仿真系统的核心功能

Inheritance

↳ [object](#)
↳ [SimulationManager](#)

Implements

[ISimulationManager](#)

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class SimulationManager : ISimulationManager
```

Remarks

该类实现了ISimulationManager接口，提供：

- 事件系统：支持发布/订阅模式的事件处理
- 实体管理：实体的注册、注销和查询
- 第三方集成：支持与外部仿真环境的对接

线程安全说明：

- 实体管理相关操作使用锁机制确保线程安全
- 事件系统的操作不保证线程安全，应在单线程环境中使用

Methods

GetAllEntities()

获取仿真系统中的所有实体

Declaration

```
public IReadOnlyList<object> GetAllEntities()
```

Returns

Type	Description
<code>ReadOnlyList<object></code>	所有实体的列表

Remarks

只返回本地注册的实体，不包括外部系统的实体 返回的列表是原始集合的副本，可以安全地遍历

线程安全：使用锁保护查询操作

GetEntitiesByType<T>()

获取特定类型的所有实体

Declaration

```
public IReadOnlyList<T> GetEntitiesByType<T>() where T : class
```

Returns

Type	Description
<code>ReadOnlyList<T></code>	指定类型的实体列表

Type Parameters

Name	Description
<i>T</i>	实体类型

Remarks

只返回本地注册的实体，不包括外部系统的实体 返回的列表是原始集合的副本，可以安全地遍历

线程安全：使用锁保护查询操作

GetEntityById(string)

根据ID获取实体

Declaration

```
public object? GetEntityById(string id)
```

Parameters

Type	Name	Description
<code>string</code>	<i>id</i>	实体ID

Returns

Type	Description
<code>object</code>	实体对象，如果不存在则返回null

Remarks

查找顺序：

- 1. 在本地实体集合中查找
- 2. 如果未找到且配置了外部适配器，尝试从外部系统获取

线程安全：使用锁保护查询操作

GetSimulationAdapter()

获取当前的仿真环境适配器

Declaration

```
public ISimulationAdapter? GetSimulationAdapter()
```

Returns

Type	Description
ISimulationAdapter	当前配置的适配器实例，如果未配置则返回null

PublishEvent<T>(T)

发布事件到仿真系统

Declaration

```
public void PublishEvent<T>(T evt)
```

Parameters

Type	Name	Description
T	<i>evt</i>	要发布的事件实例

Type Parameters

Name	Description
<i>T</i>	事件类型

Remarks

发布过程：

1. 查找并调用所有注册的事件处理器
2. 如果配置了外部仿真适配器，同时发布到外部系统

RegisterEntity(string, object)

注册实体到仿真系统

Declaration

```
public bool RegisterEntity(string id, object entity)
```

Parameters

Type	Name	Description
string	<i>id</i>	实体ID
object	<i>entity</i>	实体对象

Returns

Type	Description
bool	注册是否成功

Remarks

注册失败的情况：

- ID为空或实体为null
- ID已被其他实体使用

线程安全：使用锁保护注册操作

SetSimulationAdapter(ISimulationAdapter)

设置第三方仿真环境适配器

Declaration

```
public void SetSimulationAdapter(ISimulationAdapter adapter)
```

Parameters

Type	Name	Description
ISimulationAdapter	<i>adapter</i>	要设置的适配器实例

Remarks

设置适配器后，仿真系统将：

- 在发布事件时同步到外部系统
- 在查询实体时同时查询外部系统

Exceptions

Type	Condition
ArgumentNullException	适配器参数为null时抛出

SubscribeToEvent<T>(Action<T>)

订阅特定类型的事件

Declaration

```
public void SubscribeToEvent<T>(Action<T> handler)
```

Parameters

Type	Name	Description
Action<T>	<i>handler</i>	事件处理函数

Type Parameters

Name	Description
<i>T</i>	事件类型

Remarks

如果是首次订阅该类型事件，会创建新的处理器列表 同一个处理器可以多次订阅，会被多次调用

UnregisterEntity(string)

从仿真系统注销实体

Declaration

```
public bool UnregisterEntity(string id)
```

Parameters

Type	Name	Description
string	<i>id</i>	要注销的实体ID

Returns

Type	Description
bool	注销是否成功

Remarks

如果实体不存在，返回false 线程安全：使用锁保护注销操作

UnsubscribeFromEvent<T>(Action<T>)

取消订阅特定类型的事件

Declaration

```
public void UnsubscribeFromEvent<T>(Action<T> handler)
```

Parameters

Type	Name	Description
Action<T>	handler	要取消的事件处理函数

Type Parameters

Name	Description
T	事件类型

Remarks

如果取消订阅后该类型没有其他处理器，会删除整个处理器列表 如果处理器不存在，操作会被忽略

Implements

[ISimulationManager](#)

[Show / Hide Table of Contents](#)

Class MillimeterWaveWarnerAlarmStopEvent

毫米波告警器警报停止事件

Inheritance

↳ [object](#)
↳ [SimulationEvent](#)
↳ MillimeterWaveWarnerAlarmStopEvent

Inherited Members

[SimulationEvent.SenderId](#)
[SimulationEvent.Timestamp](#)
[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [ThreatSource.Simulation](#)

Assembly: ThreatSource.dll

Syntax

```
public class MillimeterWaveWarnerAlarmStopEvent : SimulationEvent
```

Properties

TargetId

目标ID

Declaration

```
public string? TargetId { get; set; }
```

Property Value

Type	Description
string	

Namespace ThreatSource.Utils

Classes

MotionAlgorithm

运动算法静态类，提供各种运动计算方法

Vector3D

表示三维空间中的向量，提供基本的向量运算功能

Structs

Orientation

表示三维空间中的方向，使用欧拉角表示

Vector2D

表示二维平面上的向量

[Show / Hide Table of Contents](#)

Enum MissileType

导弹类型枚举，定义了系统支持的所有导弹类型

Namespace: [ThreatSource.Missile](#)

Assembly: ThreatSource.dll

Syntax

```
public enum MissileType
```

Remarks

包含以下导弹类型：

- 标准导弹：基础型号导弹
- 激光半主动制导：利用目标反射的激光能量进行制导
- 激光驾束制导：沿着激光束路径飞行
- 红外指令制导：通过红外信号接收指令进行制导
- 红外成像末制导：末段使用红外成像进行制导
- 毫米波末制导：末段使用毫米波雷达进行制导
- 末敏弹：末段敏感引信的导弹

Fields

Name	Description
InfraredCommandGuidance	
InfraredImagingTerminalGuidance	
LaserBeamRiderGuidance	
LaserSemiActiveGuidance	
MillimeterWaveTerminalGuidance	
StandardMissile	
TerminalSensitiveMissile	