Homing Missile System

This powerful and flexible Unity asset allows you to implement homing missiles that dynamically follow a designated target. Whether you're creating a fast-paced action game or adding some thrilling combat mechanics, this system will fit right into your project.

The system includes three main scripts:
- Homing Missile: The primary missile behavior script that handles movement, activation, and destruction.
- Homing Missile Pointer: A script for guiding the missile towards its target.
- Shoot Missile Example: A simple example to demonstrate how to launch a missile.

Let's walk through the setup process and how you can make the most of this system.

Why Use This Asset?

- Dynamic Targeting: The missile actively follows the designated target.
- Customizable Behavior: Adjust speed, activation time, damage, and more.
- Smooth Integration: Quickly integrate with your game using the provided example script.
- Visual Effects: Includes particle effects for smoke trails and explosions.

Getting Started

What You Need

Before you begin, make sure you have:
1. A Missile Prefab (provided in the asset).
2. A Target GameObject for the missile to follow.

Setting It Up

1. Import the Homing Missile System: Drag and drop the asset folder into your Unity project.
2. Prepare Your Missile Prefab: Attach the homing_missile script to the missile prefab. Ensure the missile prefab has a Rigidbody component.
3. Assign the Target Pointer: Add a child GameObject to the missile prefab. Attach the homing_missile_pointer script to this child.
4. Use the Example Script: Create a GameObject in your scene to act as the shooter (e.g., a player or enemy). Attach the shoot_missile_example script to this GameObject. Assign the missile prefab and target in the Inspector.
5. Launch the Missile: Call the shoot_missile method in the shoot_missile_example script when you want to launch a missile (e.g., via a button press or event).

Understanding the Scripts

homing_missile Script

This script controls the missile's behavior, including activation, movement, and destruction.
Key Fields:
- speed: The forward speed of the missile when fully active.
- downspeed: The speed during the initial descent.
- damage: The damage dealt upon collision.
- timebeforeactivition: Time before the missile becomes fully active.
- timebeforebursting: Time before the missile starts trailing smoke.
- timebeforedestruction: Time before the missile self-destructs.
- turnSpeed: Controls how quickly the missile turns towards its target.
- target: The GameObject the missile follows.
- shooter: The GameObject that launched the missile (to avoid self-collision).

Key Methods:
- usemissile(): Activates and launches the missile.
- DestroyMe(): Handles missile destruction, including visual effects.

homing_missile_pointer Script

This script ensures the missile is always oriented towards the target.
Key Fields:
- target: The GameObject the pointer tracks.

Key Behavior:
Continuously rotates the pointer to face the target using transform.LookAt.

shoot_missile_example Script

This script demonstrates how to launch a missile.
Key Fields:
- missile_prefab: The prefab of the missile to launch.
- target: The GameObject the missile will follow.

Key Method:
- shoot_missile(): Instantiates the missile prefab, assigns necessary parameters, and activates the missile.

Example Usage

Here's how you can launch a missile using the example script:
1. Attach the shoot_missile_example script to a GameObject in your scene (e.g., a player).
2. Assign the missile prefab and target in the Inspector.
3. Trigger the shoot_missile method, for instance:

```csharp
using UnityEngine;
using HomingMissile;

public class MissileLauncher : MonoBehaviour
{
   public shoot_missile_example missileLauncher;

   void Update()
   {
      if (Input.GetKeyDown(KeyCode.Space))
      {
         missileLauncher.shoot_missile();
      }
   }
}
```

Customization Ideas

- Adjust Speed and Damage: Tweak the missile's speed, damage, and other fields in the Inspector to suit your game's needs.
- Add Effects: Enhance the visuals by modifying the smoke or destruction effects.
- Dynamic Targeting: Change the target dynamically during gameplay for more advanced mechanics.

Troubleshooting

Here are some common issues and how to resolve them:
- Missile Doesn't Follow the Target: Ensure the target field in both the missile and pointer scripts is assigned correctly. Confirm the target GameObject is active in the scene.
- Visual Effects Missing: Check the smoke and destruction effect references in the homing_missile script.
- Missile Self-Destructs Immediately: Verify the timebeforeactivition, timebeforebursting, and timebeforedestruction values.

Need Help?

If you have questions or encounter issues, don't hesitate to reach out.