Namespace NWH.Common

Classes

MathUtility (NWH.Common.MathUtility.html)

Mathematical utility functions for common calculations.

Class MathUtility

Mathematical utility functions for common calculations.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ MathUtility

Namespace: NWH (NWH.html).Common (NWH.Common.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class MathUtility
```

Methods

ClampWithRemainder(ref float, in float, out float)

Clamps a value to a range and outputs how much it exceeded the range. Useful for clamping values while preserving overflow information.

Declaration

```
public static void ClampWithRemainder(ref float x, in float range, out float remainder)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*x*	Value to clamp (will be modified).
float (https://learn.microsoft.com/dotnet/api/system.single)	*range*	Range limit (value will be clamped to [-range, +range]).
float (https://learn.microsoft.com/dotnet/api/system.single)	*remainder*	Amount by which x exceeded the range (output).

Namespace NWH.Common.AssetInfo

Classes

AssetInfo (NWH.Common.AssetInfo.AssetInfo.html)

ScriptableObject containing metadata and URLs for an NWH asset. Used by the welcome window and asset information systems.

Class AssetInfo

ScriptableObject containing metadata and URLs for an NWH asset. Used by the welcome window and asset information systems.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ ScriptableObject (https://docs.unity3d.com/ScriptReference/ScriptableObject.html)
      ↳ AssetInfo

Namespace: NWH (NWH.html).Common (NWH.Common.html).AssetInfo (NWH.Common.AssetInfo.html)

Assembly: NWH.DWP2.dll

Syntax

```
[CreateAssetMenu(fileName = "AssetInfo", menuName = "NWH/AssetInfo", order = 0)]
public class AssetInfo : ScriptableObject
```

Fields

assetName

Display name of the asset.

Declaration

```
public string assetName
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

assetURL

Unity Asset Store URL for this asset.

Declaration

```
public string assetURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

changelogURL

URL to the changelog documentation page.

Declaration

```
public string changelogURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

discordURL

Discord server invite link for support and community.

Declaration

```
public string discordURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

documentationURL

URL to the main documentation page.

Declaration

```
public string documentationURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

emailURL

Support email contact link.

Declaration

```
public string emailURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

forumURL

Unity Forum thread URL for this asset.

Declaration

```
public string forumURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

publisherURL

NWH publisher page URL on Unity Asset Store.

Declaration

```
public string publisherURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

quickStartURL

URL to quick start guide documentation.

Declaration

```
public string quickStartURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

recentUpdates

Recent updates/changes in the current version (3-5 bullet points).

Declaration

```
[TextArea(3, 10)]
public string[] recentUpdates
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)[]	

upgradeNotesURL

URL to upgrade notes between versions.

Declaration

```
public string upgradeNotesURL
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

version

Current version string of the asset.

Declaration

```
public string version
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

Namespace NWH.Common.Cameras

Classes

CameraChanger (NWH.Common.Cameras.CameraChanger.html)

Switches between the camera objects that are children to this object and contain camera tag, in order they appear in the hierarchy or in order they are added to the vehicle cameras list.

CameraInsideVehicle (NWH.Common.Cameras.CameraInsideVehicle.html)

Empty component that should be attached to the cameras that are inside the vehicle if interior sound change is to be used.

CameraMouseDrag (NWH.Common.Cameras.CameraMouseDrag.html)

Camera that can be dragged with the mouse.

VehicleCamera (NWH.Common.Cameras.VehicleCamera.html)

Base class for vehicle camera implementations with automatic target detection.

Enums

CameraMouseDrag.POVType (NWH.Common.Cameras.CameraMouseDrag.POVType.html)

Class CameraChanger

Switches between the camera objects that are children to this object and contain camera tag, in order they appear in the hierarchy or in order they are added to the vehicle cameras list.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ CameraChanger

Namespace: NWH (NWH.html).Common (NWH.Common.html).Cameras (NWH.Common.Cameras.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DefaultExecutionOrder(20)]
public class CameraChanger : MonoBehaviour
```

Fields

autoFindCameras

If true vehicleCameras list will be filled through cameraTag.

Declaration

```
[Tooltip("    If true vehicleCameras list will be filled through cameraTag.")]
public bool autoFindCameras
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

cameras

List of cameras that the changer will cycle through. Leave empty if you want cameras to be automatically detected. To be detected cameras need to have camera tag and be children of the object this script is attached to.

Declaration

```
[FormerlySerializedAs("vehicleCameras")]
[Tooltip("List of cameras that the changer will cycle through. Leave empty if you want camer
as to be automatically detected. To be detected cameras need to have camera tag and be child
ren of the object this script is attached to.")]
public List<GameObject> cameras
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< GameObject (https://docs.unity3d.com/ScriptReference/GameObject.html)>	

currentCameraIndex

Index of the camera from vehicle cameras list that will be active first.

Declaration

```
[Tooltip("    Index of the camera from vehicle cameras list that will be active first.")]
public int currentCameraIndex
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

Methods

NextCamera()

Activates the next camera in the list, cycling back to the first camera when reaching the end. Automatically disables all other cameras and their AudioListeners.

Declaration

```
public void NextCamera()
```

PreviousCamera()

Activates the previous camera in the list, cycling back to the last camera when reaching the beginning. Automatically disables all other cameras and their AudioListeners.

Declaration

```
public void PreviousCamera()
```

Class CameraInsideVehicle

Empty component that should be attached to the cameras that are inside the vehicle if interior sound change is to be used.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ CameraInsideVehicle

Namespace: NWH (NWH.html).Common (NWH.Common.html).Cameras (NWH.Common.Cameras.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class CameraInsideVehicle : MonoBehaviour
```

Fields

isInsideVehicle

Is the camera inside vehicle?

Declaration

```
[Tooltip("    Is the camera inside vehicle?")]
public bool isInsideVehicle
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Class CameraMouseDrag

Camera that can be dragged with the mouse.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ VehicleCamera (NWH.Common.Cameras.VehicleCamera.html)
            ↳ CameraMouseDrag

Inherited Members

VehicleCamera.target (NWH.Common.Cameras.VehicleCamera.html#NWH_Common_Cameras_VehicleCamera_target)
VehicleCamera.Awake() (NWH.Common.Cameras.VehicleCamera.html#NWH_Common_Cameras_VehicleCamera_Awake)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Cameras (NWH.Common.Cameras.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class CameraMouseDrag : VehicleCamera
```

Fields

allowPanning

Can the camera be panned by the user?

Declaration

```
[Tooltip("    Can the camera be panned by the user?")]
public bool allowPanning
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

allowRotation

Can the camera be rotated by the user?

Declaration

```
[Tooltip("    Can the camera be rotated by the user?")]
public bool allowRotation
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

distance

Distance from target at which camera will be positioned. Might vary depending on smoothing.

Declaration

```
[Range(0, 100)]
[Tooltip("    Distance from target at which camera will be positioned. Might vary depending
on smoothing.")]
public float distance
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

followTargetPitchAndYaw

If true the camera will rotate with the vehicle along the X and Y axis.

Declaration

```
[FormerlySerializedAs("followTargetsRotation")]
[Tooltip("    If true the camera will rotate with the vehicle along the X and Y axis.")]
public bool followTargetPitchAndYaw
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

followTargetRoll

If true the camera will rotate with the vehicle along the Z axis.

Declaration

```
[Tooltip("    If true the camera will rotate with the vehicle along the Z axis.")]
public bool followTargetRoll
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

initXRotation

Initial rotation around the X axis (up/down)

Declaration

```
[Tooltip("    Initial rotation around the X axis (up/down)")]
public float initXRotation
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

initYRotation

Initial rotation around the Y axis (left/right)

Declaration

```
[Tooltip("    Initial rotation around the Y axis (left/right)")]
public float initYRotation
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxDistance

Maximum distance that will be reached when zooming out.

Declaration

```
[Range(0, 100)]
[Tooltip("    Maximum distance that will be reached when zooming out.")]
public float maxDistance
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

minDistance

Minimum distance that will be reached when zooming in.

Declaration

```
[Range(0, 100)]
[Tooltip("    Minimum distance that will be reached when zooming in.")]
public float minDistance
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

panningSensitivity

Sensitivity of panning input.

Declaration

```
[Tooltip("    Sensitivity of panning input.")]
public Vector2 panningSensitivity
```

Field Value

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

povType

Camera POV type. First person camera will invert controls. Zoom is not available in 1st person.

Declaration

```
[Tooltip("Camera POV type. First person camera will invert controls.\r\nZoom is not availabl
e in 1st person.")]
public CameraMouseDrag.POVType povType
```

Field Value

Type	Description
CameraMouseDrag (NWH.Common.Cameras.CameraMouseDrag.html). POVType (NWH.Common.Cameras.CameraMouseDrag.POVType.html)	

rotationSensitivity

Sensitivity of rotation input.

Declaration

```
[Tooltip("    Sensitivity of rotation input.")]
public Vector2 rotationSensitivity
```

Field Value

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

rotationSmoothing

Smoothing of the camera rotation.

Declaration

```
[Range(0, 1)]
[Tooltip("    Smoothing of the camera rotation.")]
public float rotationSmoothing
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

shakeAxisIntensity

Movement intensity per axis. Set to 0 to disable movement on that axis or negative to reverse it.

Declaration

```
[Tooltip("    Movement intensity per axis. Set to 0 to disable movement on that axis or nega
tive to reverse it.")]
public Vector3 shakeAxisIntensity
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

shakeIntensity

How much will the head move around for the given g-force.

Declaration

```
[Range(0, 1)]
[Tooltip("    How much will the head move around for the given g-force.")]
public float shakeIntensity
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

shakeMaxOffset

Maximum head movement from the initial position.

Declaration

```
[Range(0, 1)]
[Tooltip("    Maximum head movement from the initial position.")]
public float shakeMaxOffset
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

shakeSmoothing

Smoothing of the head movement.

Declaration

```
[Range(0, 1)]
[Tooltip("    Smoothing of the head movement.")]
public float shakeSmoothing
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

targetPositionOffset

Look position offset from the target center.

Declaration

```
[Tooltip("    Look position offset from the target center.")]
public Vector3 targetPositionOffset
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

useShake

Should camera movement on acceleration be used?

Declaration

```
[Tooltip("Should camera movement on acceleration be used?")]
public bool useShake
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

verticalMaxAngle

Maximum vertical angle the camera can achieve.

Declaration

```
[Range(-90, 90)]
[Tooltip("    Maximum vertical angle the camera can achieve.")]
public float verticalMaxAngle
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

verticalMinAngle

Minimum vertical angle the camera can achieve.

Declaration

```
[Range(-90, 90)]
[Tooltip("    Minimum vertical angle the camera can achieve.")]
public float verticalMinAngle
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

zoomSensitivity

Sensitivity of the middle mouse button / wheel.

Declaration

```
[Range(0, 15)]
[Tooltip("    Sensitivity of the middle mouse button / wheel.")]
public float zoomSensitivity
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

ClampAngle(float, float, float)

Declaration

```
public float ClampAngle(float angle, float min, float max)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*angle*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*min*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*max*	

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

OnDrawGizmosSelected()

Declaration

```
public void OnDrawGizmosSelected()
```

Enum CameraMouseDrag.POVType

Namespace: NWH (NWH.html).Common (NWH.Common.html).Cameras (NWH.Common.Cameras.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum CameraMouseDrag.POVType
```

Fields

Name	Description
FirstPerson	First-person camera view from inside the vehicle.
ThirdPerson	Third-person camera view from outside the vehicle.

Class VehicleCamera

Base class for vehicle camera implementations with automatic target detection.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ VehicleCamera
            ↳ CameraMouseDrag (NWH.Common.Cameras.CameraMouseDrag.html)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Cameras (NWH.Common.Cameras.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class VehicleCamera : MonoBehaviour
```

Fields

target

Transform to track. Auto-detects parent Rigidbody if not assigned.

Declaration

```
[Tooltip("Transform that this script is targeting. Can be left empty if head movement is not
being used.")]
public Transform target
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

Methods

Awake()

Declaration

```
public virtual void Awake()
```

See Also

CameraChanger (NWH.Common.Cameras.CameraChanger.html)
CameraInsideVehicle (NWH.Common.Cameras.CameraInsideVehicle.html)
CameraMouseDrag (NWH.Common.Cameras.CameraMouseDrag.html)

Namespace NWH.Common.CoM

Classes

MassAffector (NWH.Common.CoM.MassAffector.html)

Simple mass affector implementation that contributes a fixed mass at its transform position to the vehicle's center of mass calculations.

VariableCenterOfMass (NWH.Common.CoM.VariableCenterOfMass.html)

Dynamic center of mass and inertia calculation system that updates Rigidbody properties based on attached mass affectors like fuel tanks, cargo loads, and passengers.

Interfaces

IMassAffector (NWH.Common.CoM.IMassAffector.html)

Interface for objects that contribute mass and affect vehicle center of mass calculations. Implemented by fuel tanks, cargo systems, and other variable mass components.

Interface IMassAffector

Interface for objects that contribute mass and affect vehicle center of mass calculations. Implemented by fuel tanks, cargo systems, and other variable mass components.

Namespace: NWH (NWH.html).Common (NWH.Common.html).CoM (NWH.Common.CoM.html)

Assembly: NWH.DWP2.dll

Syntax

```
public interface IMassAffector
```

Remarks

Mass affectors allow dynamic vehicle physics by contributing their mass and position to the overall center of mass calculation. As fuel depletes or cargo loads change, the vehicle's handling characteristics update automatically.

Methods

GetMass()

Current mass of this affector in kilograms. Should return variable values for fuel tanks, cargo, etc.

Declaration

```
float GetMass()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Mass in kg

GetTransform()

Returns transform of the mass affector.

Declaration

```
Transform GetTransform()
```

Returns

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

GetWorldCenterOfMass()

World position of this affector's center of mass. Used for weighted center of mass calculations.

Declaration

```
Vector3 GetWorldCenterOfMass()
```

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Class MassAffector

Simple mass affector implementation that contributes a fixed mass at its transform position to the vehicle's center of mass calculations.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ MassAffector

Implements

IMassAffector (NWH.Common.CoM.IMassAffector.html)

Namespace: NWH (NWH.html).Common (NWH.Common.html).CoM (NWH.Common.CoM.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class MassAffector : MonoBehaviour, IMassAffector
```

Remarks

Use this component for static mass contributions like passengers, cargo, or equipment. For dynamic masses like fuel tanks, create a custom IMassAffector implementation that returns varying mass values.

Fields

mass

Mass contribution of this affector in kilograms.

Declaration

```
public float mass
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

GetMass()

Returns the mass of this affector.

Declaration

```
public float GetMass()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Mass in kilograms.

GetTransform()

Returns the transform of this mass affector.

Declaration

```
public Transform GetTransform()
```

Returns

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

GetWorldCenterOfMass()

Returns the world position of this mass affector's center of mass.

Declaration

```
public Vector3 GetWorldCenterOfMass()
```

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Implements

IMassAffector (NWH.Common.CoM.IMassAffector.html)

Class VariableCenterOfMass

Dynamic center of mass and inertia calculation system that updates Rigidbody properties based on attached mass affectors like fuel tanks, cargo loads, and passengers.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ VariableCenterOfMass

Namespace: NWH (NWH.html).Common (NWH.Common.html).CoM (NWH.Common.CoM.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DisallowMultipleComponent]
[DefaultExecutionOrder(-1000)]
[RequireComponent(typeof(Rigidbody))]
public class VariableCenterOfMass : MonoBehaviour
```

Remarks

VariableCenterOfMass enables realistic vehicle physics behavior by automatically adjusting center of mass and inertia tensor as vehicle loading changes. This affects handling characteristics, stability, and acceleration response without requiring complex rigidbody hierarchies.

The system calculates total mass, weighted center of mass position, and inertia contributions from all IMassAffector components. Changes in fuel level, cargo loading, or passenger weight immediately affect vehicle dynamics, creating realistic weight distribution effects.

Critical for vehicle realism: Front-heavy vehicles understeer more, rear-heavy vehicles may oversteer, and high center of mass increases rollover tendency. The system updates these characteristics dynamically based on actual mass distribution.

Fields

affectors

Objects attached or part of the vehicle affecting its center of mass and inertia.

Declaration

```
[NonSerialized]
public IMassAffector[] affectors
```

Field Value

Type	Description
IMassAffector (NWH.Common.CoM.IMassAffector.html)[]	

baseMass

Base mass of the object, without IMassAffectors.

Declaration

```
[Tooltip("Base mass of the object, without IMassAffectors.")]
public float baseMass
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

centerOfMass

Center of mass of the object. Auto calculated. To adjust center of mass use centerOfMassOffset.

Declaration

```
[Tooltip("Center of mass of the rigidbody. Needs to be readjusted when new colliders are add
ed.")]
public Vector3 centerOfMass
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

combinedCenterOfMass

Combined center of mass, including the Rigidbody and any IMassAffectors.

Declaration

```
public Vector3 combinedCenterOfMass
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

combinedInertiaTensor

Total inertia tensor. Includes Rigidbody and IMassAffectors.

Declaration

```
public Vector3 combinedInertiaTensor
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

combinedMass

Total mass of the object with masses of IMassAffectors counted in.

Declaration

```
[Tooltip("Total mass of the object with masses of IMassAffectors counted in.")]
public float combinedMass
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

dimensions

Object dimensions in [m]. X - width, Y - height, Z - length. It is important to set the correct dimensions or otherwise inertia might be calculated incorrectly.

Declaration

```
[Tooltip("Object dimensions in [m]. X - width, Y - height, Z - length.\r\nIt is important to
set the correct dimensions or otherwise inertia might be calculated incorrectly.")]
public Vector3 dimensions
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

inertiaTensor

Vector by which the inertia tensor of the rigidbody will be scaled on Start(). Due to the uniform density of the rigidbodies, versus the very non-uniform density of a vehicle, inertia can feel off. Use this to adjust inertia tensor values.

Declaration

```
[Tooltip("    Vector by which the inertia tensor of the rigidbody will be scaled on Start
().\r\n    Due to the unform density of the rigidbodies, versus the very non-uniform density
of a vehicle, inertia can feel\r\n    off.\r\n    Use this to adjust inertia tensor value
s.")]
public Vector3 inertiaTensor
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

isDirty

When true, properties will be recalculated in the next FixedUpdate. Call MarkDirty() when mass affectors change to trigger update.

Declaration

```
[Tooltip("When true, properties will be recalculated in the next FixedUpdate. Automatically
managed.")]
public bool isDirty
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

useDefaultCenterOfMass

When enabled the Unity-calculated center of mass will be used.

Declaration

```
[Tooltip("When enabled the Unity-calculated center of mass will be used.")]
public bool useDefaultCenterOfMass
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

useDefaultInertia

When true inertia settings will be ignored and default Rigidbody inertia tensor will be used.

Declaration

```
[Tooltip("When true inertia settings will be ignored and default Rigidbody inertia tensor wi
ll be used.")]
public bool useDefaultInertia
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

useDefaultMass

Should the default Rigidbody mass be used?

Declaration

```
public bool useDefaultMass
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

useMassAffectors

If true, the script will search for any IMassAffectors attached as a child (recursively) of this script and use them when calculating mass, center of mass and inertia tensor.

Declaration

```
public bool useMassAffectors
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

CalculateInertia(Vector3, float)

Calculates inertia tensor for a cuboid with given dimensions and mass. Uses parallel axis theorem for rectangular prism approximation.

Declaration

```
public static Vector3 CalculateInertia(Vector3 dimensions, float mass)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*dimensions*	Object dimensions in meters (width, height, length)
float (https://learn.microsoft.com/dotnet/api/system.single)	*mass*	Total mass in kilograms

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Inertia tensor components (Ix, Iy, Iz) in kg·m²

CalculateInertiaTensorOffset(Vector3)

Calculates the inertia tensor of the Rigidbody and attached mass affectors.

Declaration

```
public Vector3 CalculateInertiaTensorOffset(Vector3 dimensions)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*dimensions*	

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

CalculateMass()

Calculates the mass of the Rigidbody and attached mass affectors.

Declaration

```
public float CalculateMass()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

CalculateRelativeCenterOfMassOffset()

Calculates the center of mass of the Rigidbody and attached mass affectors.

Declaration

```
public Vector3 CalculateRelativeCenterOfMassOffset()
```

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

GetMassAffectors()

Updates list of IMassAffectors attached to this object. Call after IMassAffector has been added or removed from the object.

Declaration

```
public IMassAffector[] GetMassAffectors()
```

Returns

Type	Description
IMassAffector (NWH.Common.CoM.IMassAffector.html)[]	

GetWorldCenterOfMass()

Gets the combined center of mass position in world space coordinates.

Declaration

```
public Vector3 GetWorldCenterOfMass()
```

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	World space position of the center of mass

MarkDirty()

Mark properties as needing recalculation. Call this when mass affectors change (fuel consumption, cargo loading, etc.).

Declaration

```
public void MarkDirty()
```

UpdateAllProperties()

Recalculates all Rigidbody properties (mass, center of mass, and inertia) based on current settings and affectors. Called automatically when isDirty flag is set.

Declaration

```
public void UpdateAllProperties()
```

UpdateCoM()

Calculates and applies the CoM to the Rigidbody.

Declaration

```
public void UpdateCoM()
```

UpdateInertia(bool)

Calculates and applies the inertia tensor to the Rigidbody.

Declaration

```
public void UpdateInertia(bool applyUnchanged = false)
```

Parameters

Type	Name	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*applyUnchanged*	

UpdateMass()

Calculates and applies the total mass to the Rigidbody. Includes mass from affectors if useMassAffectors is enabled.

Declaration

```
public void UpdateMass()
```

See Also

IMassAffector (NWH.Common.CoM.IMassAffector.html)
MassAffector (NWH.Common.CoM.MassAffector.html)
Vehicle (NWH.Common.Vehicles.Vehicle.html)

Namespace NWH.Common.Demo

Classes

DemoCameraNameDisplay (NWH.Common.Demo.DemoCameraNameDisplay.html)

Displays the name of the currently active main camera in a Text component. Updates every 0.1 seconds.

DemoDtSetter (NWH.Common.Demo.DemoDtSetter.html)

Sets Time.fixedDeltaTime to a specific value for demo scenes. Default is 0.008333s (120Hz) for optimal physics performance.

DemoOscillator (NWH.Common.Demo.DemoOscillator.html)

Moves a Rigidbody in a sinusoidal oscillation pattern. Useful for creating moving platforms or obstacles in demo scenes.

DemoRotator (NWH.Common.Demo.DemoRotator.html)

Continuously rotates a Rigidbody at a constant rate. Useful for rotating platforms or visual elements in demo scenes.

DemoVehicleNameDisplay (NWH.Common.Demo.DemoVehicleNameDisplay.html)

Displays the name and type of the currently active vehicle in a Text component. Updates every 0.1 seconds.

DemoWelcomeMessage (NWH.Common.Demo.DemoWelcomeMessage.html)

Controls the display of a welcome message panel in demo scenes. Shows the message when running outside the editor.

DragObject (NWH.Common.Demo.DragObject.html)

Simple script that drags Rigidbody behind the mouse cursor when MMB is held down.

FpsToText (NWH.Common.Demo.FpsToText.html)

Displays the current framerate in a Text component with optional color coding. Supports both instantaneous and averaged FPS measurements.

RigidbodyFPSController (NWH.Common.Demo.RigidbodyFPSController.html)

Simple first-person controller using physics-based movement. Useful for testing and navigating demo scenes on foot.

Class DemoCameraNameDisplay

Displays the name of the currently active main camera in a Text component. Updates every 0.1 seconds.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ DemoCameraNameDisplay

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(Text))]
public class DemoCameraNameDisplay : MonoBehaviour
```

Class DemoDtSetter

Sets Time.fixedDeltaTime to a specific value for demo scenes. Default is 0.008333s (120Hz) for optimal physics performance.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ DemoDtSetter

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DefaultExecutionOrder(-500)]
public class DemoDtSetter : MonoBehaviour
```

Fields

fixedDeltaTime

Target physics update rate in seconds. Default 0.008333s equals 120Hz.

Declaration

```
public float fixedDeltaTime
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Class DemoOscillator

Moves a Rigidbody in a sinusoidal oscillation pattern. Useful for creating moving platforms or obstacles in demo scenes.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ DemoOscillator

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DemoOscillator : MonoBehaviour
```

Fields

speed

Speed of the oscillation in Hz. Higher values result in faster movement.

Declaration

```
public float speed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

travel

Maximum displacement from the starting position in each axis.

Declaration

```
public Vector3 travel
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Class DemoRotator

Continuously rotates a Rigidbody at a constant rate. Useful for rotating platforms or visual elements in demo scenes.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ DemoRotator

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DemoRotator : MonoBehaviour
```

Fields

rotation

Rotation speed in degrees per second for each axis.

Declaration

```
public Vector3 rotation
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Class DemoVehicleNameDisplay

Displays the name and type of the currently active vehicle in a Text component. Updates every 0.1 seconds.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ DemoVehicleNameDisplay

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(Text))]
public class DemoVehicleNameDisplay : MonoBehaviour
```

Class DemoWelcomeMessage

Controls the display of a welcome message panel in demo scenes. Shows the message when running outside the editor.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ DemoWelcomeMessage

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DemoWelcomeMessage : MonoBehaviour
```

Fields

closeButton

Button used to close the welcome message panel.

Declaration

```
public Button closeButton
```

Field Value

Type	Description
Button	

welcomeMessageGO

GameObject containing the welcome message UI.

Declaration

```
public GameObject welcomeMessageGO
```

Field Value

Type	Description
GameObject (https://docs.unity3d.com/ScriptReference/GameObject.html)	

Class DragObject

Simple script that drags Rigidbody behind the mouse cursor when MMB is held down.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ DragObject

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DragObject : MonoBehaviour
```

Class FpsToText

Displays the current framerate in a Text component with optional color coding. Supports both instantaneous and averaged FPS measurements.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ FpsToText

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(Text))]
public class FpsToText : MonoBehaviour
```

Fields

bad

Text color when framerate is below badBelow threshold.

Declaration

```
public Color bad
```

Field Value

Type	Description
Color (https://docs.unity3d.com/ScriptReference/Color.html)	

badBelow

FPS threshold below which the color changes to bad (red).

Declaration

```
public int badBelow
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

forceIntResult

Round FPS to nearest integer for cleaner display.

Declaration

```
public bool forceIntResult
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

good

Text color when framerate is above okayBelow threshold.

Declaration

```
public Color good
```

Field Value

Type	Description
Color (https://docs.unity3d.com/ScriptReference/Color.html)	

groupSampling

Use averaging over multiple samples instead of single frame measurement.

Declaration

```
public bool groupSampling
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

okay

Text color when framerate is between badBelow and okayBelow.

Declaration

```
public Color okay
```

Field Value

Type	Description
Color (https://docs.unity3d.com/ScriptReference/Color.html)	

okayBelow

FPS threshold below which the color changes to okay (yellow).

Declaration

```
public int okayBelow
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

sampleSize

Number of samples to average when groupSampling is enabled.

Declaration

```
public int sampleSize
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

smoothed

Use Time.smoothDeltaTime instead of Time.deltaTime for calculations.

Declaration

```
public bool smoothed
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

updateTextEvery

Update text display every N frames. 1 = every frame.

Declaration

```
public int updateTextEvery
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

useColors

Enable color coding based on framerate thresholds.

Declaration

```
public bool useColors
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

useSystemTick

Use Environment.TickCount instead of Time.deltaTime for calculations.

Declaration

```
public bool useSystemTick
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

GetSystemFramerate()

Declaration

```
protected virtual int GetSystemFramerate()
```

Returns

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

Group()

Declaration

```
protected virtual void Group()
```

Reset()

Declaration

```
protected virtual void Reset()
```

SingleFrame()

Declaration

```
protected virtual void SingleFrame()
```

Start()

Declaration

```
protected virtual void Start()
```

Update()

Declaration

```
protected virtual void Update()
```

Class RigidbodyFPSController

Simple first-person controller using physics-based movement. Useful for testing and navigating demo scenes on foot.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ RigidbodyFPSController

Namespace: NWH (NWH.html).Common (NWH.Common.html).Demo (NWH.Common.Demo.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(Rigidbody))]
[RequireComponent(typeof(CapsuleCollider))]
public class RigidbodyFPSController : MonoBehaviour
```

Remarks

Based on Unity Community Wiki example. Uses Rigidbody for physics-accurate movement with mouse-look camera control.

Fields

gravity

Downward acceleration force in m/s^2.

Declaration

```
public float gravity
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

jumpHeight

Maximum height of jumps in meters.

Declaration

```
public float jumpHeight
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxVelocityChange

Maximum velocity change per fixed update, controls acceleration responsiveness.

Declaration

```
public float maxVelocityChange
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maximumY

Maximum upward look angle in degrees.

Declaration

```
public float maximumY
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

minimumY

Maximum downward look angle in degrees.

Declaration

```
public float minimumY
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

sensitivityX

Horizontal mouse look sensitivity.

Declaration

```
public float sensitivityX
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

sensitivityY

Vertical mouse look sensitivity.

Declaration

```
public float sensitivityY
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

speed

Movement speed in meters per second.

Declaration

```
public float speed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Namespace NWH.Common.Input

Classes

InputManagerSceneInputProvider (NWH.Common.Input.InputManagerSceneInputProvider.html)

Scene input provider using Unity's legacy Input Manager system. Requires input axes and buttons to be configured in Project Settings > Input Manager.

InputProvider (NWH.Common.Input.InputProvider.html)

Base class from which all input providers inherit.

InputSystemSceneInputProvider (NWH.Common.Input.InputSystemSceneInputProvider.html)

Unity Input System implementation of scene input provider. Handles camera controls and scene navigation using the new Input System.

InputUtils (NWH.Common.Input.InputUtils.html)

Utility methods for safe input retrieval with automatic fallback to default keys. Prevents errors when Input Manager bindings are missing.

MobileInputButton (NWH.Common.Input.MobileInputButton.html)

Extended Unity UI Button with state tracking for mobile input handling. Provides hasBeenClicked and isPressed flags for easier input polling.

MobileSceneInputProvider (NWH.Common.Input.MobileSceneInputProvider.html)

Scene input provider for mobile platforms using on-screen UI buttons. Requires MobileInputButton components assigned to changeCameraButton and changeVehicleButton fields.

SceneInputActions (NWH.Common.Input.SceneInputActions.html)

Provides programmatic access to UnityEngine.InputSystem.InputActionAsset, UnityEngine.InputSystem.InputActionMap, UnityEngine.InputSystem.InputAction and UnityEngine.InputSystem.InputControlScheme instances defined in asset "Packages/com.nwh.common/Runtime/Input/InputSystem/SceneInputActions.inputactions".

SceneInputProviderBase (NWH.Common.Input.SceneInputProviderBase.html)

InputProvider for scene and camera related behavior.

Structs

SceneInputActions.CameraControlsActions (NWH.Common.Input.SceneInputActions.CameraControlsActions.html)

Provides access to input actions defined in input action map "CameraControls".

SceneInputActions.SceneControlsActions (NWH.Common.Input.SceneInputActions.SceneControlsActions.html)

Provides access to input actions defined in input action map "SceneControls".

Interfaces

SceneInputActions.ICameraControlsActions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html)

Interface to implement callback methods for all input action callbacks associated with input actions defined by "CameraControls" which allows adding and removing callbacks.

SceneInputActions.ISceneControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html)

Interface to implement callback methods for all input action callbacks associated with input actions defined by "SceneControls" which allows adding and removing callbacks.

Class InputManagerSceneInputProvider

Scene input provider using Unity's legacy Input Manager system. Requires input axes and buttons to be configured in Project Settings > Input Manager.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ SceneInputProviderBase (NWH.Common.Input.SceneInputProviderBase.html)
              ↳ InputManagerSceneInputProvider

Inherited Members

SceneInputProviderBase.requireCameraPanningModifier (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_requireCameraPanningModifier)
SceneInputProviderBase.requireCameraRotationModifier (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_requireCameraRotationModifier)
InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.Awake() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Awake)
InputProvider.OnDestroy() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)
InputProvider.CombinedInput<T>(Func<T, int>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class InputManagerSceneInputProvider : SceneInputProviderBase
```

Methods

CameraPanning()

Returns camera panning input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public override Vector2 CameraPanning()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CameraPanning() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraPanning)

CameraPanningModifier()

Returns true when the camera panning modifier button is held. If requireCameraPanningModifier is false, always returns true.

Declaration

```
public override bool CameraPanningModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.CameraPanningModifier() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraPanningModifier)

CameraRotation()

Returns camera rotation input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public override Vector2 CameraRotation()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CameraRotation() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraRotation)

CameraRotationModifier()

Returns true when the camera rotation modifier button is held. If requireCameraRotationModifier is false, always returns true.

Declaration

```
public override bool CameraRotationModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.CameraRotationModifier() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraRotationModifier)

CameraZoom()

Returns camera zoom input value. Positive = zoom in, negative = zoom out.

Declaration

```
public override float CameraZoom()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Overrides

SceneInputProviderBase.CameraZoom() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraZoom)

ChangeCamera()

Returns true when the change camera button is pressed.

Declaration

```
public override bool ChangeCamera()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ChangeCamera() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ChangeCamera)

ChangeVehicle()

Returns true when the change vehicle button is pressed.

Declaration

```
public override bool ChangeVehicle()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ChangeVehicle() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ChangeVehicle)

CharacterMovement()

Returns character movement input as a Vector2 (x = horizontal, y = forward/back).

Declaration

```
public override Vector2 CharacterMovement()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CharacterMovement() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CharacterMovement)

ToggleGUI()

Returns true when the toggle GUI button is pressed.

Declaration

```
public override bool ToggleGUI()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ToggleGUI()
(NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ToggleGUI)

Class InputProvider

Base class from which all input providers inherit.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider
            ↳ SceneInputProviderBase (NWH.Common.Input.SceneInputProviderBase.html)
            ↳ ShipInputProvider (NWH.DWP2.ShipController.ShipInputProvider.html)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public abstract class InputProvider : MonoBehaviour
```

Fields

Instances

List of all InputProviders in the scene.

Declaration

```
public static List<InputProvider> Instances
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< InputProvider (NWH.Common.Input.InputProvider.html)>	

Methods

Awake()

Declaration

```
public virtual void Awake()
```

CombinedInput<T>(Func<T, bool>)

Returns combined input of all InputProviders present in the scene. Result will be positive if any InputProvider has the selected input set to true. T is a type of InputProvider that the input will be retrieved from.

Declaration

```
public static bool CombinedInput<T>(Func<T, bool> selector) where T : InputProvider
```

Parameters

Type	Name	Description
Func (https://learn.microsoft.com/dotnet/api/system.func-2)<T, bool (https://learn.microsoft.com/dotnet/api/system.boolean)>	*selector*	

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Type Parameters

Name	Description
T	

CombinedInput<T>(Func<T, int>)

Returns combined input of all InputProviders present in the scene. Result will be a sum of all inputs of the selected type. T is a type of InputProvider that the input will be retrieved from.

Declaration

```
public static int CombinedInput<T>(Func<T, int> selector) where T : InputProvider
```

Parameters

Type	Name	Description
Func (https://learn.microsoft.com/dotnet/api/system.func-2)<T, int (https://learn.microsoft.com/dotnet/api/system.int32)>	*selector*	

Returns

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

Type Parameters

Name	Description
T	

CombinedInput<T>(Func<T, float>)

Returns combined input of all InputProviders present in the scene. Result will be a sum of all inputs of the selected type. T is a type of InputProvider that the input will be retrieved from.

Declaration

```
public static float CombinedInput<T>(Func<T, float> selector) where T : InputProvider
```

Parameters

Type	Name	Description
Func (https://learn.microsoft.com/dotnet/api/system.func-2)<T, float (https://learn.microsoft.com/dotnet/api/system.single)>	*selector*	

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Type Parameters

Name	Description
T	

CombinedInput<T>(Func<T, Vector2>)

Returns combined input of all InputProviders present in the scene. Result will be a sum of all inputs of the selected type. T is a type of InputProvider that the input will be retrieved from.

Declaration

```
public static Vector2 CombinedInput<T>(Func<T, Vector2> selector) where T : InputProvider
```

Parameters

Type	Name	Description
Func (https://learn.microsoft.com/dotnet/api/system.func-2)<T, Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)>	*selector*	

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Type Parameters

Name	Description
T	

OnDestroy()

Declaration

```
public virtual void OnDestroy()
```

Class InputSystemSceneInputProvider

Unity Input System implementation of scene input provider. Handles camera controls and scene navigation using the new Input System.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ SceneInputProviderBase (NWH.Common.Input.SceneInputProviderBase.html)
              ↳ InputSystemSceneInputProvider

Inherited Members

SceneInputProviderBase.requireCameraPanningModifier (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_requireCameraPanningModifier)
SceneInputProviderBase.requireCameraRotationModifier (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_requireCameraRotationModifier)
InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.OnDestroy() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)
InputProvider.CombinedInput<T>(Func<T, int>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class InputSystemSceneInputProvider : SceneInputProviderBase
```

Fields

sceneInputActions

Declaration

```
public SceneInputActions sceneInputActions
```

Field Value

Type	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html)	

Methods

Awake()

Declaration

```
public override void Awake()
```

Overrides

InputProvider.Awake() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Awake)

CameraPanning()

Returns camera panning input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public override Vector2 CameraPanning()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CameraPanning() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraPanning)

CameraPanningModifier()

Returns true when the camera panning modifier button is held. If requireCameraPanningModifier is false, always returns true.

Declaration

```
public override bool CameraPanningModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.CameraPanningModifier() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraPanningModifier)

CameraRotation()

Returns camera rotation input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public override Vector2 CameraRotation()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CameraRotation() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraRotation)

CameraRotationModifier()

Returns true when the camera rotation modifier button is held. If requireCameraRotationModifier is false, always returns true.

Declaration

```
public override bool CameraRotationModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.CameraRotationModifier() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraRotati

onModifier)

CameraZoom()

Returns camera zoom input value. Positive = zoom in, negative = zoom out.

Declaration

```
public override float CameraZoom()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Overrides

SceneInputProviderBase.CameraZoom() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraZoom)

ChangeCamera()

Returns true when the change camera button is pressed.

Declaration

```
public override bool ChangeCamera()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ChangeCamera() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ChangeCamera)

ChangeVehicle()

Returns true when the change vehicle button is pressed.

Declaration

```
public override bool ChangeVehicle()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ChangeVehicle() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ChangeVehicle)

CharacterMovement()

Returns character movement input as a Vector2 (x = horizontal, y = forward/back).

Declaration

```
public override Vector2 CharacterMovement()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CharacterMovement() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CharacterMovement)

ToggleGUI()

Returns true when the toggle GUI button is pressed.

Declaration

```
public override bool ToggleGUI()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ToggleGUI() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ToggleGUI)

Class InputUtils

Utility methods for safe input retrieval with automatic fallback to default keys. Prevents errors when Input Manager bindings are missing.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ InputUtils

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class InputUtils
```

Methods

TryGetAxis(string, bool)

Attempts to retrieve axis value from Input Manager, returns 0 if binding is missing.

Declaration

```
public static float TryGetAxis(string axisName, bool showWarning = true)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*axisName*	Input Manager axis name to query.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*showWarning*	Display warning message when axis is missing.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Axis value between -1 and 1, or 0 if binding is missing.

TryGetAxisRaw(string, bool)

Attempts to retrieve raw axis value from Input Manager, returns 0 if binding is missing. Raw axes return only -1, 0, or 1 without smoothing.

Declaration

```
public static float TryGetAxisRaw(string axisName, bool showWarning = true)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*axisName*	Input Manager axis name to query.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*showWarning*	Display warning message when axis is missing.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Raw axis value (-1, 0, or 1), or 0 if binding is missing.

TryGetButton(string, KeyCode, bool)

Attempts to retrieve button state from Input Manager, falls back to KeyCode if binding is missing.

Declaration

```
public static bool TryGetButton(string buttonName, KeyCode altKey, bool showWarning = true)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*buttonName*	Input Manager button name to query.
KeyCode	*altKey*	Fallback KeyCode to use if binding is missing.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*showWarning*	Display warning message when falling back to default key.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if button is currently held down.

TryGetButtonDown(string, KeyCode, bool)

Attempts to retrieve button press from Input Manager, falls back to KeyCode if binding is missing.

Declaration

```
public static bool TryGetButtonDown(string buttonName, KeyCode altKey, bool showWarning = tr
ue)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*buttonName*	Input Manager button name to query.
KeyCode	*altKey*	Fallback KeyCode to use if binding is missing.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*showWarning*	Display warning message when falling back to default key.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True on the frame the button was pressed.

Class MobileInputButton

Extended Unity UI Button with state tracking for mobile input handling. Provides hasBeenClicked and isPressed flags for easier input polling.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ UIBehaviour
            ↳ Selectable
              ↳ Button
                ↳ MobileInputButton

Implements

IMoveHandler
IPointerDownHandler
IPointerUpHandler
IPointerEnterHandler
IPointerExitHandler
ISelectHandler
IDeselectHandler
IPointerClickHandler
ISubmitHandler
IEventSystemHandler

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DefaultExecutionOrder(1000)]
public class MobileInputButton : Button, IMoveHandler, IPointerDownHandler, IPointerUpHandle
r, IPointerEnterHandler, IPointerExitHandler, ISelectHandler, IDeselectHandler, IPointerClic
kHandler, ISubmitHandler, IEventSystemHandler
```

Fields

hasBeenClicked

True for one frame after the button is clicked. Automatically resets to false.

Declaration

```
public bool hasBeenClicked
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

isPressed

True while the button is being held down. Updates every frame.

Declaration

```
public bool isPressed
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

OnPointerDown(PointerEventData)

Declaration

```
public override void OnPointerDown(PointerEventData eventData)
```

Parameters

Type	Name	Description
PointerEventData	*eventData*	

Overrides

UnityEngine.UI.Selectable.OnPointerDown(UnityEngine.EventSystems.PointerEventData)

Implements

UnityEngine.EventSystems.IMoveHandler
UnityEngine.EventSystems.IPointerDownHandler
UnityEngine.EventSystems.IPointerUpHandler
UnityEngine.EventSystems.IPointerEnterHandler
UnityEngine.EventSystems.IPointerExitHandler
UnityEngine.EventSystems.ISelectHandler
UnityEngine.EventSystems.IDeselectHandler
UnityEngine.EventSystems.IPointerClickHandler
UnityEngine.EventSystems.ISubmitHandler
UnityEngine.EventSystems.IEventSystemHandler

Class MobileSceneInputProvider

Scene input provider for mobile platforms using on-screen UI buttons. Requires MobileInputButton components assigned to changeCameraButton and changeVehicleButton fields.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ SceneInputProviderBase (NWH.Common.Input.SceneInputProviderBase.html)
              ↳ MobileSceneInputProvider

Inherited Members

SceneInputProviderBase.requireCameraPanningModifier (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_requireCameraPanningModifier)
SceneInputProviderBase.requireCameraRotationModifier (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_requireCameraRotationModifier)
InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.Awake() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Awake)
InputProvider.OnDestroy() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)
InputProvider.CombinedInput<T>(Func<T, int>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class MobileSceneInputProvider : SceneInputProviderBase
```

Fields

changeCameraButton

UI button for changing camera. Should reference a MobileInputButton in the scene.

Declaration

```
public MobileInputButton changeCameraButton
```

Field Value

Type	Description
MobileInputButton (NWH.Common.Input.MobileInputButton.html)	

changeVehicleButton

UI button for changing vehicle. Should reference a MobileInputButton in the scene.

Declaration

```
public MobileInputButton changeVehicleButton
```

Field Value

Type	Description
MobileInputButton (NWH.Common.Input.MobileInputButton.html)	

Methods

CameraPanning()

Returns camera panning input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public override Vector2 CameraPanning()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CameraPanning()
(NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraPanning)

g)

CameraPanningModifier()

Returns true when the camera panning modifier button is held. If requireCameraPanningModifier is false, always returns true.

Declaration

```
public override bool CameraPanningModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.CameraPanningModifier() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraPanningModifier)

CameraRotation()

Returns camera rotation input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public override Vector2 CameraRotation()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CameraRotation() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraRotation)

CameraRotationModifier()

Returns true when the camera rotation modifier button is held. If requireCameraRotationModifier is false, always returns true.

Declaration

```
public override bool CameraRotationModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.CameraRotationModifier() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraRotationModifier)

CameraZoom()

Returns camera zoom input value. Positive = zoom in, negative = zoom out.

Declaration

```
public override float CameraZoom()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Overrides

SceneInputProviderBase.CameraZoom() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CameraZoom)

ChangeCamera()

Returns true when the change camera button is pressed.

Declaration

```
public override bool ChangeCamera()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ChangeCamera() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ChangeCamera)

ChangeVehicle()

Returns true when the change vehicle button is pressed.

Declaration

```
public override bool ChangeVehicle()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ChangeVehicle() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ChangeVehicle)

CharacterMovement()

Returns character movement input as a Vector2 (x = horizontal, y = forward/back).

Declaration

```
public override Vector2 CharacterMovement()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

Overrides

SceneInputProviderBase.CharacterMovement() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_CharacterMovement)

ToggleGUI()

Returns true when the toggle GUI button is pressed.

Declaration

```
public override bool ToggleGUI()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Overrides

SceneInputProviderBase.ToggleGUI() (NWH.Common.Input.SceneInputProviderBase.html#NWH_Common_Input_SceneInputProviderBase_ToggleGUI)

Class SceneInputActions

Provides programmatic access to UnityEngine.InputSystem.InputActionAsset, UnityEngine.InputSystem.InputActionMap, UnityEngine.InputSystem.InputAction and UnityEngine.InputSystem.InputControlScheme instances defined in asset "Packages/com.nwh.common/Runtime/Input/InputSystem/SceneInputActions.inputactions".

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ SceneInputActions

Implements

IInputActionCollection2
IInputActionCollection
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<InputAction>
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.ienumerable)
IDisposable (https://learn.microsoft.com/dotnet/api/system.idisposable)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class SceneInputActions : IInputActionCollection2, IInputActionCollection, IEnumerabl
e<InputAction>, IEnumerable, IDisposable
```

Remarks

This class is source generated and any manual edits will be discarded if the associated asset is reimported or modified.

Examples

```
using namespace UnityEngine;
using UnityEngine.InputSystem;

// Example of using an InputActionMap named "Player" from a UnityEngine.MonoBehaviour implem
enting callback interface.
public class Example : MonoBehaviour, MyActions.IPlayerActions
{
    private MyActions_Actions m_Actions;                  // Source code representation of a
sset.
    private MyActions_Actions.PlayerActions m_Player;     // Source code representation of a
ction map.

    void Awake()
    {
        m_Actions = new MyActions_Actions();              // Create asset object.
        m_Player = m_Actions.Player;                      // Extract action map object.
        m_Player.AddCallbacks(this);                      // Register callback interface IPl
ayerActions.
    }

    void OnDestroy()
    {
        m_Actions.Dispose();                              // Destroy asset object.
    }

    void OnEnable()
    {
        m_Player.Enable();                                // Enable all actions within map.
    }

    void OnDisable()
    {
        m_Player.Disable();                               // Disable all actions within map.
    }

    #region Interface implementation of MyActions.IPlayerActions

    // Invoked when "Move" action is either started, performed or canceled.
    public void OnMove(InputAction.CallbackContext context)
    {
        Debug.Log($"OnMove: {context.ReadValue<Vector2>()}");
    }

    // Invoked when "Attack" action is either started, performed or canceled.
    public void OnAttack(InputAction.CallbackContext context)
    {
        Debug.Log($"OnAttack: {context.ReadValue<float>()}");
    }

    #endregion
}
```

Constructors

SceneInputActions()

Constructs a new instance.

Declaration

```
public SceneInputActions()
```

Properties

CameraControls

Provides a new SceneInputActions.CameraControlsActions (NWH.Common.Input.SceneInputActions.CameraControlsActions.html) instance referencing this action map.

Declaration

```
public SceneInputActions.CameraControlsActions CameraControls { get; }
```

Property Value

Type	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). CameraControlsActions (NWH.Common.Input.SceneInputActions.CameraControlsActions.html)	

SceneControls

Provides a new SceneInputActions.SceneControlsActions (NWH.Common.Input.SceneInputActions.SceneControlsActions.html) instance referencing this action map.

Declaration

```
public SceneInputActions.SceneControlsActions SceneControls { get; }
```

Property Value

Type	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). SceneControlsActions (NWH.Common.Input.SceneInputActions.SceneControlsActions.html)	

asset

Provides access to the underlying asset instance.

Declaration

```
public InputActionAsset asset { get; }
```

Property Value

Type	Description
InputActionAsset	

bindingMask

Declaration

```
public InputBinding? bindingMask { get; set; }
```

Property Value

Type	Description
InputBinding?	

bindings

Declaration

```
public IEnumerable<InputBinding> bindings { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1) <InputBinding>	

controlSchemes

Declaration

```
public ReadOnlyArray<InputControlScheme> controlSchemes { get; }
```

Property Value

Type	Description
ReadOnlyArray<InputControlScheme>	

devices

Declaration

```
public ReadOnlyArray<InputDevice>? devices { get; set; }
```

Property Value

Type	Description
ReadOnlyArray<InputDevice>?	

Methods

Contains(InputAction)

Declaration

```
public bool Contains(InputAction action)
```

Parameters

Type	Name	Description
InputAction	*action*	

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Disable()

Declaration

```
public void Disable()
```

Dispose()

Destroys this asset and all associated UnityEngine.InputSystem.InputAction instances.

Declaration

```
public void Dispose()
```

Enable()

Declaration

```
public void Enable()
```

~SceneInputActions()

Declaration

```
protected ~SceneInputActions()
```

FindAction(string, bool)

Declaration

```
public InputAction FindAction(string actionNameOrId, bool throwIfNotFound = false)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*actionNameOrId*	
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*throwIfNotFound*	

Returns

Type	Description
InputAction	

FindBinding(InputBinding, out InputAction)

Declaration

```
public int FindBinding(InputBinding bindingMask, out InputAction action)
```

Parameters

Type	Name	Description
InputBinding	*bindingMask*	
InputAction	*action*	

Returns

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

GetEnumerator()

Returns an enumerator that iterates through the collection.

Declaration

```
public IEnumerator<InputAction> GetEnumerator()
```

Returns

Type	Description
IEnumerator (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerator-1) <InputAction>	An enumerator that can be used to iterate through the collection.

Implements

UnityEngine.InputSystem.IInputActionCollection2
UnityEngine.InputSystem.IInputActionCollection
IEnumerable<T> (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.ienumerable)
IDisposable (https://learn.microsoft.com/dotnet/api/system.idisposable)

Struct SceneInputActions.CameraControlsActions

Provides access to input actions defined in input action map "CameraControls".

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public struct SceneInputActions.CameraControlsActions
```

Constructors

CameraControlsActions(SceneInputActions)

Construct a new instance of the input action map wrapper class.

Declaration

```
public CameraControlsActions(SceneInputActions wrapper)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html)	*wrapper*	

Properties

CameraPanning

Provides access to the underlying input action "CameraControls/CameraPanning".

Declaration

```
public InputAction CameraPanning { get; }
```

Property Value

Type	Description
InputAction	

CameraPanningModifier

Provides access to the underlying input action "CameraControls/CameraPanningModifier".

Declaration

```
public InputAction CameraPanningModifier { get; }
```

Property Value

Type	Description
InputAction	

CameraRotation

Provides access to the underlying input action "CameraControls/CameraRotation".

Declaration

```
public InputAction CameraRotation { get; }
```

Property Value

Type	Description
InputAction	

CameraRotationModifier

Provides access to the underlying input action "CameraControls/CameraRotationModifier".

Declaration

```
public InputAction CameraRotationModifier { get; }
```

Property Value

Type	Description
InputAction	

CameraZoom

Provides access to the underlying input action "CameraControls/CameraZoom".

Declaration

```
public InputAction CameraZoom { get; }
```

Property Value

Type	Description
InputAction	

ChangeCamera

Provides access to the underlying input action "CameraControls/ChangeCamera".

Declaration

```
public InputAction ChangeCamera { get; }
```

Property Value

Type	Description
InputAction	

enabled

Declaration

```
public bool enabled { get; }
```

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

AddCallbacks(ICameraControlsActions)

Adds UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed and UnityEngine.InputSystem.InputAction.canceled callbacks provided via on all input actions contained in this map.

Declaration

```
public void AddCallbacks(SceneInputActions.ICameraControlsActions instance)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html).ICameraControlsActions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html)	*instance*	Callback instance.

Remarks

If `instance` is `null` or `instance` have already been added this method does nothing.

See Also

SceneInputActions (NWH.Common.Input.SceneInputActions.html).CameraControlsActions (NWH.Common.Input.SceneInputActions.CameraControlsActions.html)

Disable()

Declaration

```
public void Disable()
```

Enable()

Declaration

```
public void Enable()
```

Get()

Provides access to the underlying input action map instance.

Declaration

```
public InputActionMap Get()
```

Returns

Type	Description
InputActionMap	

RemoveCallbacks(ICameraControlsActions)

Unregisters and unregisters all input action callbacks via UnregisterCallbacks(ICameraControlsActions) (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html).

Declaration

```
public void RemoveCallbacks(SceneInputActions.ICameraControlsActions instance)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). ICameraControlsActions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html)	*instance*	

See Also

UnregisterCallbacks (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html)(ICameraControls Actions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html))

SetCallbacks(ICameraControlsActions)

Replaces all existing callback instances and previously registered input action callbacks associated with them with callbacks provided via .

Declaration

```
public void SetCallbacks(SceneInputActions.ICameraControlsActions instance)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). ICameraControlsActions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html)	*instance*	

Remarks

If `instance` is `null`, calling this method will only unregister all existing callbacks but not register any new callbacks.

See Also

AddCallbacks(ICameraControlsActions) (NWH.Common.Input.SceneInputActions.CameraControlsActions.html#NWH_Common_Input_SceneInputActions_CameraControlsActions_AddCallbacks_NWH_Common_Input_SceneInputActions_ICameraControlsActions_)
RemoveCallbacks(ICameraControlsActions) (NWH.Common.Input.SceneInputActions.CameraControlsActions.html#NWH_Common_Input_SceneInputActions_CameraControlsActions_RemoveCallbacks_NWH_Common_Input_SceneInputActions_ICameraControlsActions_)
UnregisterCallbacks (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html)(ICameraControls Actions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html))

Operators

implicit operator InputActionMap(CameraControlsActions)

Implicitly converts an to an instance.

Declaration

```
public static implicit operator InputActionMap(SceneInputActions.CameraControlsActions set)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). CameraControlsActions (NWH.Common.Input.SceneInputActions.CameraControlsActions.html)	*set*	

Returns

Type	Description
InputActionMap	

Interface SceneInputActions.ICameraControlsActions

Interface to implement callback methods for all input action callbacks associated with input actions defined by "CameraControls" which allows adding and removing callbacks.

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public interface SceneInputActions.ICameraControlsActions
```

Methods

OnCameraPanning(CallbackContext)

Method invoked when associated input action "CameraPanning" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnCameraPanning(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnCameraPanningModifier(CallbackContext)

Method invoked when associated input action "CameraPanningModifier" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnCameraPanningModifier(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnCameraRotation(CallbackContext)

Method invoked when associated input action "CameraRotation" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnCameraRotation(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnCameraRotationModifier(CallbackContext)

Method invoked when associated input action "CameraRotationModifier" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnCameraRotationModifier(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnCameraZoom(CallbackContext)

Method invoked when associated input action "CameraZoom" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnCameraZoom(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnChangeCamera(CallbackContext)

Method invoked when associated input action "ChangeCamera" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnChangeCamera(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

See Also

AddCallbacks (NWH.Common.Input.SceneInputActions.CameraControlsActions.html#NWH_Common_Input_SceneInputActions_CameraControlsActions_AddCallbacks_NWH_Common_Input_SceneInputActions_ICameraControlsActions_)(ICameraControlsActions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html))
RemoveCallbacks (NWH.Common.Input.SceneInputActions.CameraControlsActions.html#NWH_Common_Input_SceneInputActions_CameraControlsActions_RemoveCallbacks_NWH_Common_Input_SceneInputActions_ICameraControlsActions_)(ICameraControlsActions (NWH.Common.Input.SceneInputActions.ICameraControlsActions.html))

Interface SceneInputActions.ISceneControlsActions

Interface to implement callback methods for all input action callbacks associated with input actions defined by "SceneControls" which allows adding and removing callbacks.

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public interface SceneInputActions.ISceneControlsActions
```

Methods

OnChangeVehicle(CallbackContext)

Method invoked when associated input action "ChangeVehicle" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnChangeVehicle(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnDragObjectModifier(CallbackContext)

Method invoked when associated input action "DragObjectModifier" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnDragObjectModifier(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnFPSMovement(CallbackContext)

Method invoked when associated input action "FPSMovement" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnFPSMovement(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnShowCursor(CallbackContext)

Method invoked when associated input action "ShowCursor" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnShowCursor(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnToggleGUI(CallbackContext)

Method invoked when associated input action "ToggleGUI" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnToggleGUI(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

See Also

AddCallbacks (NWH.Common.Input.SceneInputActions.SceneControlsActions.html#NWH_Common_Input_SceneInputActions_SceneControlsActions_AddCallbacks_NWH_Common_Input_SceneInputActions_ISceneControlsActions_)(IScene ControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html))
RemoveCallbacks (NWH.Common.Input.SceneInputActions.SceneControlsActions.html#NWH_Common_Input_SceneInputActions_SceneControlsActions_RemoveCallbacks_NWH_Common_Input_SceneInputActions_ISceneControlsActions_)(IScene ControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html))

Struct SceneInputActions.SceneControlsActions

Provides access to input actions defined in input action map "SceneControls".

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public struct SceneInputActions.SceneControlsActions
```

Constructors

SceneControlsActions(SceneInputActions)

Construct a new instance of the input action map wrapper class.

Declaration

```
public SceneControlsActions(SceneInputActions wrapper)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html)	*wrapper*	

Properties

ChangeVehicle

Provides access to the underlying input action "SceneControls/ChangeVehicle".

Declaration

```
public InputAction ChangeVehicle { get; }
```

Property Value

Type	Description
InputAction	

DragObjectModifier

Provides access to the underlying input action "SceneControls/DragObjectModifier".

Declaration

```
public InputAction DragObjectModifier { get; }
```

Property Value

Type	Description
InputAction	

FPSMovement

Provides access to the underlying input action "SceneControls/FPSMovement".

Declaration

```
public InputAction FPSMovement { get; }
```

Property Value

Type	Description
InputAction	

ShowCursor

Provides access to the underlying input action "SceneControls/ShowCursor".

Declaration

```
public InputAction ShowCursor { get; }
```

Property Value

Type	Description
InputAction	

ToggleGUI

Provides access to the underlying input action "SceneControls/ToggleGUI".

Declaration

```
public InputAction ToggleGUI { get; }
```

Property Value

Type	Description
InputAction	

enabled

Declaration

```
public bool enabled { get; }
```

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

AddCallbacks(ISceneControlsActions)

Adds UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed and UnityEngine.InputSystem.InputAction.canceled callbacks provided via on all input actions contained in this map.

Declaration

```
public void AddCallbacks(SceneInputActions.ISceneControlsActions instance)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html).ISceneControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html)	*instance*	Callback instance.

Remarks

If `instance` is `null` or `instance` have already been added this method does nothing.

See Also

SceneInputActions (NWH.Common.Input.SceneInputActions.html).SceneControlsActions (NWH.Common.Input.SceneInputActions.SceneControlsActions.html)

Disable()

Declaration

```
public void Disable()
```

Enable()

Declaration

```
public void Enable()
```

Get()

Provides access to the underlying input action map instance.

Declaration

```
public InputActionMap Get()
```

Returns

Type	Description
InputActionMap	

RemoveCallbacks(ISceneControlsActions)

Unregisters and unregisters all input action callbacks via UnregisterCallbacks(ISceneControlsActions) (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html).

Declaration

```
public void RemoveCallbacks(SceneInputActions.ISceneControlsActions instance)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). ISceneControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html)	*instance*	

See Also

UnregisterCallbacks (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html)(ISceneControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html))

SetCallbacks(ISceneControlsActions)

Replaces all existing callback instances and previously registered input action callbacks associated with them with callbacks provided via .

Declaration

```
public void SetCallbacks(SceneInputActions.ISceneControlsActions instance)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). ISceneControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html)	*instance*	

Remarks

If `instance` is `null`, calling this method will only unregister all existing callbacks but not register any new callbacks.

See Also

AddCallbacks(ISceneControlsActions) (NWH.Common.Input.SceneInputActions.SceneControlsActions.html#NWH_Common_Input_SceneInputActions_SceneControlsActions_AddCallbacks_NWH_Common_Input_SceneInputActions_ISceneControlsActions_) RemoveCallbacks(ISceneControlsActions) (NWH.Common.Input.SceneInputActions.SceneControlsActions.html#NWH_Common_Input_SceneInputActions_SceneControlsActions_RemoveCallbacks_NWH_Common_Input_SceneInputActions_ISceneControlsActions_) UnregisterCallbacks (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html)(ISceneControlsActions (NWH.Common.Input.SceneInputActions.ISceneControlsActions.html))

Operators

implicit operator InputActionMap(SceneControlsActions)

Implicitly converts an to an instance.

Declaration

```
public static implicit operator InputActionMap(SceneInputActions.SceneControlsActions set)
```

Parameters

Type	Name	Description
SceneInputActions (NWH.Common.Input.SceneInputActions.html). SceneControlsActions (NWH.Common.Input.SceneInputActions.SceneControlsActions.html)	*set*	

Returns

Type	Description
InputActionMap	

Class SceneInputProviderBase

InputProvider for scene and camera related behavior.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ SceneInputProviderBase
              ↳ InputManagerSceneInputProvider (NWH.Common.Input.InputManagerSceneInputProvider.html)
              ↳ InputSystemSceneInputProvider (NWH.Common.Input.InputSystemSceneInputProvider.html)
              ↳ MobileSceneInputProvider (NWH.Common.Input.MobileSceneInputProvider.html)

Inherited Members

InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.Awake() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Awake)
InputProvider.OnDestroy()
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)
InputProvider.CombinedInput<T>(Func<T, int>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Input (NWH.Common.Input.html)

Assembly: NWH.DWP2.dll

Syntax

```
public abstract class SceneInputProviderBase : InputProvider
```

Fields

requireCameraPanningModifier

If true a button press will be required to unlock camera panning.

Declaration

```
[Tooltip("    If true a button press will be required to unlock camera panning.")]
public bool requireCameraPanningModifier
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

requireCameraRotationModifier

If true a button press will be required to unlock camera rotation.

Declaration

```
[Tooltip("    If true a button press will be required to unlock camera rotation.")]
public bool requireCameraRotationModifier
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

CameraPanning()

Returns camera panning input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public virtual Vector2 CameraPanning()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

CameraPanningModifier()

Returns true when the camera panning modifier button is held. If requireCameraPanningModifier is false, always returns true.

Declaration

```
public virtual bool CameraPanningModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

CameraRotation()

Returns camera rotation input as a Vector2 (x = horizontal, y = vertical).

Declaration

```
public virtual Vector2 CameraRotation()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

CameraRotationModifier()

Returns true when the camera rotation modifier button is held. If requireCameraRotationModifier is false, always returns true.

Declaration

```
public virtual bool CameraRotationModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

CameraZoom()

Returns camera zoom input value. Positive = zoom in, negative = zoom out.

Declaration

```
public virtual float CameraZoom()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

ChangeCamera()

Returns true when the change camera button is pressed.

Declaration

```
public virtual bool ChangeCamera()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

ChangeVehicle()

Returns true when the change vehicle button is pressed.

Declaration

```
public virtual bool ChangeVehicle()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

CharacterMovement()

Returns character movement input as a Vector2 (x = horizontal, y = forward/back).

Declaration

```
public virtual Vector2 CharacterMovement()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

ToggleGUI()

Returns true when the toggle GUI button is pressed.

Declaration

```
public virtual bool ToggleGUI()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Namespace NWH.Common.SceneManagement

Classes

VehicleChanger (NWH.Common.SceneManagement.VehicleChanger.html)

Manages switching between multiple vehicles in a scene with support for both instant and character-based (enter/exit) modes.

Enums

VehicleChanger.CharacterLocation (NWH.Common.SceneManagement.VehicleChanger.CharacterLocation.html)

Represents the player's spatial relationship to vehicles.

Class VehicleChanger

Manages switching between multiple vehicles in a scene with support for both instant and character-based (enter/exit) modes.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ VehicleChanger

Namespace: NWH (NWH.html).Common (NWH.Common.html).SceneManagement (NWH.Common.SceneManagement.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DefaultExecutionOrder(500)]
public class VehicleChanger : MonoBehaviour
```

Remarks

VehicleChanger supports two modes: - Instant switching: Press a button to cycle through vehicles immediately - Character-based: Player must walk to a vehicle and enter/exit at designated points

In character-based mode, the player can only enter vehicles when near an EnterExitPoint and the vehicle is moving slowly enough. This creates more realistic vehicle switching similar to GTA-style games.

Inactive vehicles can optionally be put to sleep to improve performance when managing many vehicles in a scene.

Fields

activeVehicleIndex

Index of the current vehicle in vehicles list.

Declaration

```
[Tooltip("    Index of the current vehicle in vehicles list.")]
public int activeVehicleIndex
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

characterBased

Is vehicle changing character based? When true changing vehicles will require getting close to them to be able to enter, opposed to pressing a button to switch between vehicles.

Declaration

```
[Tooltip("Is vehicle changing character based? When true changing vehicles will require gett
ing close to them\r\nto be able to enter, opposed to pressing a button to switch between veh
icles.")]
public bool characterBased
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

characterObject

Game object representing a character. Can also be another vehicle.

Declaration

```
[Tooltip("    Game object representing a character. Can also be another vehicle.")]
public GameObject characterObject
```

Field Value

Type	Description
GameObject (https://docs.unity3d.com/ScriptReference/GameObject.html)	

enterDistance

Maximum distance at which the character will be able to enter the vehicle.

Declaration

```
[Range(0.2, 3)]
[Tooltip("    Maximum distance at which the character will be able to enter the vehicle.")]
public float enterDistance
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

enterExitTag

Tag of the object representing the point from which the enter distance will be measured. Useful if you want to enable you character to enter only when near the door.

Declaration

```
[Tooltip("Tag of the object representing the point from which the enter distance will be mea
sured. Useful if you want to enable you character to enter only when near the door.")]
public string enterExitTag
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

location

When the location is Near, the player can enter the vehicle.

Declaration

```
[Tooltip("When the location is Near, the player can enter the vehicle.")]
public VehicleChanger.CharacterLocation location
```

Field Value

Type	Description
VehicleChanger (NWH.Common.SceneManagement.VehicleChanger.html). CharacterLocation (NWH.Common.SceneManagement.VehicleChanger.CharacterLocation.html)	

maxEnterExitVehicleSpeed

Maximum speed at which the character will be able to enter / exit the vehicle.

Declaration

```
[Tooltip("    Maximum speed at which the character will be able to enter / exit the vehicl
e.")]
public float maxEnterExitVehicleSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

onDeactivateAll

Event invoked when all vehicles are deactivated (e.g., when exiting in character-based mode).

Declaration

```
public UnityEvent onDeactivateAll
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

onVehicleChanged

Event invoked whenever the active vehicle changes.

Declaration

```
public UnityEvent onVehicleChanged
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

putOtherVehiclesToSleep

Should the vehicles that the player is currently not using be put to sleep to improve performance?

Declaration

```
[Tooltip("    Should the vehicles that the player is currently not using be put to sleep to
improve performance?")]
public bool putOtherVehiclesToSleep
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

startInVehicle

Should the player start inside the vehicle?

Declaration

```
[Tooltip("Should the player start inside the vehicle?")]
public bool startInVehicle
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

vehicles

List of all of the vehicles that can be selected and driven in the scene.

Declaration

```
[Tooltip("List of all of the vehicles that can be selected and driven in the scene.")]
public List<Vehicle> vehicles
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Vehicle (NWH.Common.Vehicles.Vehicle.html)>	

Properties

Instance

Declaration

```
public static VehicleChanger Instance { get; }
```

Property Value

Type	Description
VehicleChanger (NWH.Common.SceneManagement.VehicleChanger.html)	

Methods

ChangeVehicle(Vehicle)

Switches to the specified vehicle if it exists in the vehicles list.

Declaration

```
public void ChangeVehicle(Vehicle ac)
```

Parameters

Type	Name	Description
Vehicle (NWH.Common.Vehicles.Vehicle.html)	*ac*	Vehicle reference to switch to.

ChangeVehicle(int)

Changes vehicle to requested vehicle.

Declaration

```
public void ChangeVehicle(int index)
```

Parameters

Type	Name	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	*index*	Index of a vehicle in Vehicles list.

DeactivateAllExceptActive()

Enables the current active vehicle and optionally disables all others based on putOtherVehiclesToSleep setting.

Declaration

```
public void DeactivateAllExceptActive()
```

DeactivateAllIncludingActive()

Disables all managed vehicles including the currently active one. Used when exiting vehicles in character-based mode.

Declaration

```
public void DeactivateAllIncludingActive()
```

DeregisterVehicle(Vehicle)

Removes a vehicle from the managed vehicles list. If the vehicle was active, automatically switches to the next vehicle.

Declaration

```
public void DeregisterVehicle(Vehicle v)
```

Parameters

Type	Name	Description
Vehicle (NWH.Common.Vehicles.Vehicle.html)	*v*	Vehicle to deregister.

EnterVehicle(Vehicle)

Puts the player inside the specified vehicle and activates it. In character-based mode, stores the entry position for later exit.

Declaration

```
public void EnterVehicle(Vehicle v)
```

Parameters

Type	Name	Description
Vehicle (NWH.Common.Vehicles.Vehicle.html)	*v*	Vehicle to enter.

ExitVehicle(Vehicle)

Removes the player from the vehicle and spawns the character object nearby. Character is positioned at the stored entry location.

Declaration

```
public void ExitVehicle(Vehicle v)
```

Parameters

Type	Name	Description
Vehicle (NWH.Common.Vehicles.Vehicle.html)	*v*	Vehicle to exit.

NextVehicle()

Switches to the next vehicle in the list, wrapping to the first vehicle when reaching the end.

Declaration

```
public void NextVehicle()
```

PreviousVehicle()

Switches to the previous vehicle in the list, wrapping to the last vehicle when at the beginning.

Declaration

```
public void PreviousVehicle()
```

RegisterVehicle(Vehicle)

Adds a vehicle to the managed vehicles list if not already present. Newly registered vehicles are automatically disabled unless they are the active vehicle.

Declaration

```
public void RegisterVehicle(Vehicle v)
```

Parameters

Type	Name	Description
Vehicle (NWH.Common.Vehicles.Vehicle.html)	*v*	Vehicle to register.

Enum VehicleChanger.CharacterLocation

Represents the player's spatial relationship to vehicles.

Namespace: NWH (NWH.html).Common (NWH.Common.html).SceneManagement (NWH.Common.SceneManagement.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum VehicleChanger.CharacterLocation
```

Fields

Name	Description
Inside	Player is currently inside a vehicle.
Near	Player is close enough to enter a vehicle.
OutOfRange	Player is too far from any vehicle to interact.

Namespace NWH.Common.ShiftingOrigin

Classes

ShiftingOrigin (NWH.Common.ShiftingOrigin.ShiftingOrigin.html)

Prevents floating point precision errors by shifting all scene objects back toward world origin when the main camera exceeds the distance threshold.

Class ShiftingOrigin

Prevents floating point precision errors by shifting all scene objects back toward world origin when the main camera exceeds the distance threshold.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ ShiftingOrigin

Namespace: NWH (NWH.html).Common (NWH.Common.html).ShiftingOrigin (NWH.Common.ShiftingOrigin.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class ShiftingOrigin : MonoBehaviour
```

Remarks

As objects move far from world origin [0,0,0], floating point precision degrades causing physics jitter and rendering artifacts. ShiftingOrigin solves this by periodically moving all scene content back toward origin, keeping the player near [0,0,0] at all times.

The shift is transparent to gameplay - relative positions remain identical. Useful for open world games, flight simulators, or any scenario with large travel distances.

Only affects the current scene. For multi-scene setups, ensure one ShiftingOrigin instance per loaded scene set.

Fields

Instance

Declaration

```
public static ShiftingOrigin Instance
```

Field Value

Type	Description
ShiftingOrigin (NWH.Common.ShiftingOrigin.ShiftingOrigin.html)	

distanceThreshold

Distance from world origin in meters that triggers an origin shift. Default 500m works well for most scenarios.

Declaration

```
public float distanceThreshold
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

onAfterJump

Event invoked after the origin shift completes and physics is re-synced.

Declaration

```
public UnityEvent onAfterJump
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

onBeforeJump

Event invoked before the origin shift begins. Rigidbody sleep thresholds are temporarily disabled.

Declaration

```
public UnityEvent onBeforeJump
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

Properties

TotalOffset

Cumulative offset applied to all objects since scene start. Useful for tracking absolute world position despite origin shifts.

Declaration

```
public Vector3 TotalOffset { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Methods

RefreshCaches()

Refreshes cached references to Rigidbodies and ParticleSystems. Call this if new physics objects or particle systems are added to the scene at runtime.

Declaration

```
public void RefreshCaches()
```

Namespace NWH.Common.Utility

Classes

AnimationCurveExtensions (NWH.Common.Utility.AnimationCurveExtensions.html)

Extension methods for AnimationCurve manipulation and processing.

ArrayExtensions (NWH.Common.Utility.ArrayExtensions.html)

Extension methods for array manipulation.

GameObjectExtensions (NWH.Common.Utility.GameObjectExtensions.html)

Extension methods for GameObject and Transform operations.

GeomUtility (NWH.Common.Utility.GeomUtility.html)

Collection of geometric utility functions for 3D math operations. Includes vector math, mesh calculations, triangle operations, and spatial queries.

PIDController (NWH.Common.Utility.PIDController.html)

Proportional-Integral-Derivative controller for smooth value regulation. Used for automated control systems like cruise control, stability systems, and steering assistance.

QuaternionExtensions (NWH.Common.Utility.QuaternionExtensions.html)

Extension methods for advanced Quaternion operations. Provides interpolation methods with control over rotation direction.

UnitConverter (NWH.Common.Utility.UnitConverter.html)

Static utility class for converting between various units of measurement. Includes conversions for distance, speed, fuel efficiency, and angular velocity.

Class AnimationCurveExtensions

Extension methods for AnimationCurve manipulation and processing.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ AnimationCurveExtensions

Namespace: NWH (NWH.html).Common (NWH.Common.html).Utility (NWH.Common.Utility.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class AnimationCurveExtensions
```

Methods

GenerateCurveArray(AnimationCurve, int)

Samples an AnimationCurve at regular intervals and returns the values as an array. Useful for pre-calculating curve values for performance-critical code.

Declaration

```
public static float[] GenerateCurveArray(this AnimationCurve self, int resolution = 256)
```

Parameters

Type	Name	Description
AnimationCurve (https://docs.unity3d.com/ScriptReference/AnimationCurve.html)	*self*	The curve to sample.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*resolution*	Number of samples to take. Higher values provide more precision.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)[]	Array of sampled values from 0 to 1.

MakeSmooth(AnimationCurve)

Smooths out a scripting-generated AnimationCurve by calculating appropriate tangents. Creates smooth transitions between keyframes.

Declaration

```
public static AnimationCurve MakeSmooth(this AnimationCurve inCurve)
```

Parameters

Type	Name	Description
AnimationCurve (https://docs.unity3d.com/ScriptReference/AnimationCurve.html)	*inCurve*	The curve to smooth.

Returns

Type	Description
AnimationCurve (https://docs.unity3d.com/ScriptReference/AnimationCurve.html)	A new smoothed AnimationCurve.

Class ArrayExtensions

Extension methods for array manipulation.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ArrayExtensions

Namespace: NWH (NWH.html).Common (NWH.Common.html).Utility (NWH.Common.Utility.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class ArrayExtensions
```

Methods

Fill<T>(T[], params T[])

Efficiently fills an array by repeating a pattern of values. Uses doubling strategy for performance.

Declaration

```
public static void Fill<T>(this T[] destinationArray, params T[] value)
```

Parameters

Type	Name	Description
T[]	*destinationArray*	Array to fill.
T[]	*value*	Pattern of values to repeat throughout the array.

Type Parameters

Name	Description
T	Type of array elements.

Class GameObjectExtensions

Extension methods for GameObject and Transform operations.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ GameObjectExtensions

Namespace: NWH (NWH.html).Common (NWH.Common.html).Utility (NWH.Common.Utility.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class GameObjectExtensions
```

Methods

FindBoundsIncludeChildren(GameObject)

Calculates the combined bounds of all MeshRenderers in a GameObject and its children.

Declaration

```
public static Bounds FindBoundsIncludeChildren(this GameObject gameObject)
```

Parameters

Type	Name	Description
GameObject (https://docs.unity3d.com/ScriptReference/GameObject.html)	*gameObject*	GameObject to calculate bounds for.

Returns

Type	Description
Bounds (https://docs.unity3d.com/ScriptReference/Bounds.html)	Combined bounds encapsulating all child renderers.

GetComponentInParent<T>(Transform, bool)

Searches for a component in parent GameObjects, with option to include inactive objects. More flexible than Unity's built-in GetComponentInParent.

Declaration

```
public static T GetComponentInParent<T>(this Transform transform, bool includeInactive = tru
e) where T : Component
```

Parameters

Type	Name	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	*transform*	Starting transform.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*includeInactive*	Include inactive GameObjects in search.

Returns

Type	Description
T	First component of type T found in parents, or null if none found.

Type Parameters

Name	Description
T	Type of component to find.

GetComponentInParentsOrChildren<T>(Transform, bool)

Searches for a component in parents first, then children if not found. Combines functionality of GetComponentInParent and GetComponentInChildren.

Declaration

```
public static T GetComponentInParentsOrChildren<T>(this Transform transform, bool includeIna
ctive = true) where T : Component
```

Parameters

Type	Name	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	*transform*	Starting transform.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*includeInactive*	Include inactive GameObjects in search.

Returns

Type	Description
T	First component of type T found, or null if none found.

Type Parameters

Name	Description
T	Type of component to find.

Class GeomUtility

Collection of geometric utility functions for 3D math operations. Includes vector math, mesh calculations, triangle operations, and spatial queries.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ GeomUtility

Namespace: NWH (NWH.html).Common (NWH.Common.html).Utility (NWH.Common.Utility.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class GeomUtility
```

Methods

AreaFromFourPoints(Vector3, Vector3, Vector3, Vector3)

Calculates the area of a quadrilateral from four points.

Declaration

```
public static float AreaFromFourPoints(Vector3 p1, Vector3 p2, Vector3 p3, Vector3 p4)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	First point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	Second point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p3*	Third point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p4*	Fourth point.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Area of the quadrilateral.

AreaFromThreePoints(Vector3, Vector3, Vector3)

Calculates the area of a triangle from three points.

Declaration

```
public static float AreaFromThreePoints(Vector3 p1, Vector3 p2, Vector3 p3)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	First point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	Second point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p3*	Third point.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Area of the triangle.

ChangeLayersRecursively(Transform, string)

Changes the layer of a transform and all its children recursively.

Declaration

```
public static void ChangeLayersRecursively(this Transform trans, string name)
```

Parameters

Type	Name	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	*trans*	Root transform.
string (https://learn.microsoft.com/dotnet/api/system.string)	*name*	Layer name.

ChangeObjectAlpha(GameObject, float)

Changes the alpha value of a GameObject's material color.

Declaration

```
public static void ChangeObjectAlpha(GameObject gameObject, float alpha)
```

Parameters

Type	Name	Description
GameObject (https://docs.unity3d.com/ScriptReference/GameObject.html)	*gameObject*	GameObject to modify.
float (https://learn.microsoft.com/dotnet/api/system.single)	*alpha*	New alpha value (0-1).

ChangeObjectColor(GameObject, Color)

Changes the color of a GameObject's material.

Declaration

```
public static void ChangeObjectColor(GameObject gameObject, Color color)
```

Parameters

Type	Name	Description
GameObject (https://docs.unity3d.com/ScriptReference/GameObject.html)	*gameObject*	GameObject to modify.
Color (https://docs.unity3d.com/ScriptReference/Color.html)	*color*	New color.

ClampMagnitude(Vector3, float, float)

Clamps the magnitude of a vector between minimum and maximum values.

Declaration

```
public static Vector3 ClampMagnitude(this Vector3 v, float min, float max)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v*	Vector to clamp.
float (https://learn.microsoft.com/dotnet/api/system.single)	*min*	Minimum magnitude.
float (https://learn.microsoft.com/dotnet/api/system.single)	*max*	Maximum magnitude.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Vector with clamped magnitude.

CopySign(float, float)

Copies the sign from one float to the magnitude of another.

Declaration

```
public static float CopySign(float mag, float sgn)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*mag*	Magnitude value.
float (https://learn.microsoft.com/dotnet/api/system.single)	*sgn*	Sign donor value.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Magnitude with the sign of sgn.

DistanceAlongNormal(Vector3, Vector3, Vector3)

Calculates the distance between two points projected along a normal vector.

Declaration

```
public static float DistanceAlongNormal(Vector3 a, Vector3 b, Vector3 normal)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*a*	First point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*b*	Second point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*normal*	Normal vector to project along.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Distance along normal.

Equal(Quaternion, Quaternion)

Checks if two Quaternion values are approximately equal.

Declaration

```
public static bool Equal(this Quaternion a, Quaternion b)
```

Parameters

Type	Name	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*a*	First quaternion.
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*b*	Second quaternion.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if angle between quaternions is less than 0.1 degrees.

FindArea(Vector3, Vector3, Vector3, Vector3)

Calculates the area of a quadrilateral defined by four points.

Declaration

```
public static float FindArea(Vector3 A, Vector3 B, Vector3 C, Vector3 D)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*A*	First corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*B*	Second corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*C*	Third corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*D*	Fourth corner.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Area of the quad.

FindCenter(Vector3, Vector3, Vector3, Vector3)

Finds the center point of a quad or triangle defined by 3 or 4 points.

Declaration

```
public static Vector3 FindCenter(Vector3 a, Vector3 b, Vector3 c, Vector3 d)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*a*	First corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*b*	Second corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*c*	Third corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*d*	Fourth corner (can equal first corner for triangle).

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Center point.

FindChordLine(Vector3, Vector3, Vector3, Vector3, float)

Finds a point along the chord line of a quad at the specified percentage.

Declaration

```
public static Vector3 FindChordLine(Vector3 a, Vector3 b, Vector3 c, Vector3 d, float chordP
ercent)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*a*	First corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*b*	Second corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*c*	Third corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*d*	Fourth corner.
float (https://learn.microsoft.com/dotnet/api/system.single)	*chordPercent*	Position along chord (0-1).

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Point on chord line.

FindDistanceToSegment(Vector3, Vector3, Vector3)

Calculates the distance between a point and a line segment.

Declaration

```
public static float FindDistanceToSegment(Vector3 pt, Vector3 p1, Vector3 p2)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pt*	Point to measure from.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	First endpoint of segment.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	Second endpoint of segment.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Distance to the segment.

FindMeshCenter(Mesh)

Find mesh center by averaging. Returns local center.

Declaration

```
public static Vector3 FindMeshCenter(Mesh mesh)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*mesh*	

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

FindSpanLine(Vector3, Vector3, Vector3, Vector3, float)

Finds a point along the span line of a quad at the specified percentage.

Declaration

```
public static Vector3 FindSpanLine(Vector3 a, Vector3 b, Vector3 c, Vector3 d, float spanPer
cent)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*a*	First corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*b*	Second corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*c*	Third corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*d*	Fourth corner.
float (https://learn.microsoft.com/dotnet/api/system.single)	*spanPercent*	Position along span (0-1).

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Point on span line.

InverseTransformPointUnscaled(Transform, Vector3)

Transforms a point from world to local space without applying scale.

Declaration

```
public static Vector3 InverseTransformPointUnscaled(this Transform transform, Vector3 positi
on)
```

Parameters

Type	Name	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	*transform*	Transform to use.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*position*	World position.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Local position without scale.

LinePlaneIntersection(Vector3, Vector3, Vector3, Vector3)

Finds the intersection point between a line and a plane.

Declaration

```
public static Vector3 LinePlaneIntersection(Vector3 planePoint, Vector3 planeNormal, Vector3
linePoint, Vector3 lineDirection)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*planePoint*	Point on the plane.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*planeNormal*	Normal vector of the plane.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*linePoint*	Point on the line.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*lineDirection*	Direction of the line.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Intersection point, or Vector3.zero if parallel.

MeshArea(Mesh)

Calculates area of a complete mesh.

Declaration

```
public static float MeshArea(Mesh mesh)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*mesh*	

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

NearEqual(Vector3, Vector3, float)

Checks if two Vector3 values are approximately equal within a threshold. Uses squared magnitude for performance.

Declaration

```
public static bool NearEqual(this Vector3 a, Vector3 b, float threshold = 0.01)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*a*	First vector.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*b*	Second vector.
float (https://learn.microsoft.com/dotnet/api/system.single)	*threshold*	Maximum squared distance to consider equal.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if vectors are within threshold distance.

NearestPointOnLine(Vector3, Vector3, Vector3)

Finds the nearest point on an infinite line to a given point.

Declaration

```
public static Vector3 NearestPointOnLine(Vector3 linePnt, Vector3 lineDir, Vector3 pnt)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*linePnt*	Point on the line.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*lineDir*	Direction of the line.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pnt*	Point to find nearest point from.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Nearest point on the line.

NearlyEqual(float, float, double)

Checks if two float values are nearly equal within an epsilon threshold.

Declaration

```
public static bool NearlyEqual(this float a, float b, double epsilon)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*a*	First value.
float (https://learn.microsoft.com/dotnet/api/system.single)	*b*	Second value.
double (https://learn.microsoft.com/dotnet/api/system.double)	*epsilon*	Maximum difference to consider equal.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if values are within epsilon.

Perpendicular(Vector3)

Calculates a perpendicular vector to the given vector.

Declaration

```
public static Vector3 Perpendicular(this Vector3 v)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v*	Input vector.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Perpendicular vector.

PointInTriangle(Vector3, Vector3, Vector3, Vector3, float)

Checks if a point lies inside a triangle.

Declaration

```
public static bool PointInTriangle(Vector3 A, Vector3 B, Vector3 C, Vector3 P, float dotThre
shold = 0.001)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*A*	First triangle vertex.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*B*	Second triangle vertex.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*C*	Third triangle vertex.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*P*	Point to test.
float (https://learn.microsoft.com/dotnet/api/system.single)	*dotThreshold*	Tolerance for point-on-plane test.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if point is inside triangle.

PointIsInsideRect(Vector2)

Checks if a 2D point is inside the screen rectangle.

Declaration

```
public static bool PointIsInsideRect(Vector2 point)
```

Parameters

Type	Name	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*point*	Point to check.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if point is inside screen bounds.

ProjectedMeshArea(Mesh, Vector3)

Calculates area of a mesh as viewed from the direction vector.

Declaration

```
public static float ProjectedMeshArea(Mesh mesh, Vector3 direction)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*mesh*	
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*direction*	

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

QuadLerp(Vector3, Vector3, Vector3, Vector3, float, float)

Performs bilinear interpolation on a quad defined by four points.

Declaration

```
public static Vector3 QuadLerp(Vector3 a, Vector3 b, Vector3 c, Vector3 d, float u, float v)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*a*	First corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*b*	Second corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*c*	Third corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*d*	Fourth corner.
float (https://learn.microsoft.com/dotnet/api/system.single)	*u*	U parameter (0-1).
float (https://learn.microsoft.com/dotnet/api/system.single)	*v*	V parameter (0-1).

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Interpolated point.

QuaternionMagnitude(Quaternion)

Calculates the magnitude of a quaternion.

Declaration

```
public static float QuaternionMagnitude(Quaternion q)
```

Parameters

Type	Name	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*q*	Quaternion.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Magnitude.

RectArea(Vector3, Vector3, Vector3, Vector3)

Calculates the area of a rectangle from four corner points.

Declaration

```
public static float RectArea(Vector3 p1, Vector3 p2, Vector3 p3, Vector3 p4)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	First corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	Second corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p3*	Third corner.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p4*	Fourth corner.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Area of the rectangle.

RotatePointAroundPivot(Vector3, Vector3, Vector3)

Rotates a point around a pivot by the specified angles.

Declaration

```
public static Vector3 RotatePointAroundPivot(Vector3 point, Vector3 pivot, Vector3 angles)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*point*	Point to rotate.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pivot*	Pivot point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*angles*	Euler angles for rotation.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Rotated point.

RoundToStep(int, int)

Rounds a value to the nearest multiple of step.

Declaration

```
public static float RoundToStep(int value, int step)
```

Parameters

Type	Name	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	*value*	Value to round.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*step*	Step size.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Rounded value.

RoundToStep(float, float)

Rounds a value to the nearest multiple of step.

Declaration

```
public static float RoundToStep(float value, float step)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*value*	Value to round.
float (https://learn.microsoft.com/dotnet/api/system.single)	*step*	Step size.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Rounded value.

RoundedMax(Vector3)

Returns a vector with only the largest component preserved (rounded to 1 or -1), others set to 0.

Declaration

```
public static Vector3 RoundedMax(this Vector3 v)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v*	Input vector.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Vector with dominant axis isolated.

SignedVolumeOfTriangle(Vector3, Vector3, Vector3)

Calculates the signed volume contribution of a triangle relative to the origin.

Declaration

```
public static float SignedVolumeOfTriangle(Vector3 p1, Vector3 p2, Vector3 p3)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	First vertex.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	Second vertex.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p3*	Third vertex.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Signed volume.

SquareDistance(Vector3, Vector3)

Calculates squared distance between two points. Faster than regular distance.

Declaration

```
public static float SquareDistance(Vector3 a, Vector3 b)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*a*	First point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*b*	Second point.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Squared distance.

TransformPointUnscaled(Transform, Vector3)

Transforms a point from local to world space without applying scale.

Declaration

```
public static Vector3 TransformPointUnscaled(this Transform transform, Vector3 position)
```

Parameters

Type	Name	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	*transform*	Transform to use.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*position*	Local position.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	World position without scale.

TriArea(Vector3, Vector3, Vector3)

Calculates area of a single triangle from it's three points.

Declaration

```
public static float TriArea(Vector3 p1, Vector3 p2, Vector3 p3)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p3*	

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

TriArea(Vector3, Vector3, Vector3, Vector3)

Declaration

```
public static float TriArea(Vector3 p1, Vector3 p2, Vector3 p3, Vector3 view)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p3*	
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*view*	

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Vector3Abs(Vector3)

Returns a vector with absolute values of all components.

Declaration

```
public static Vector3 Vector3Abs(Vector3 v)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v*	Input vector.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Vector with absolute values.

Vector3Lerp(Vector3, Vector3, float)

Linear interpolation between two vectors with value clamping.

Declaration

```
public static Vector3 Vector3Lerp(Vector3 v1, Vector3 v2, float value)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v1*	Start vector.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v2*	End vector.
float (https://learn.microsoft.com/dotnet/api/system.single)	*value*	Interpolation value (0-1).

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Interpolated vector.

Vector3OneOver(Vector3)

Returns a vector with reciprocal values (1/x, 1/y, 1/z).

Declaration

```
public static Vector3 Vector3OneOver(Vector3 v)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v*	Input vector.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Vector with reciprocal values.

Vector3RoundToInt(Vector3)

Rounds all components of a vector to nearest integer.

Declaration

```
public static Vector3 Vector3RoundToInt(Vector3 v)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*v*	Input vector.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Rounded vector.

VolumeOfMesh(Mesh)

Calculates the volume enclosed by a mesh.

Declaration

```
public static float VolumeOfMesh(Mesh mesh)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*mesh*	Mesh to calculate volume for.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Volume of the mesh.

Class PIDController

Proportional-Integral-Derivative controller for smooth value regulation. Used for automated control systems like cruise control, stability systems, and steering assistance.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ PIDController

Namespace: NWH (NWH.html).Common (NWH.Common.html).Utility (NWH.Common.Utility.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class PIDController
```

Remarks

PID controllers combine three control strategies:

- Proportional: Reacts to current error
- Integral: Eliminates accumulated error over time
- Derivative: Anticipates future error based on rate of change Tune the three gain values to achieve desired response characteristics.

Constructors

PIDController(float, float, float, float, float)

Creates a new PID controller with specified gains and output limits.

Declaration

```
public PIDController(float gainProportional, float gainIntegral, float gainDerivative, float
outputMin, float outputMax)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*gainProportional*	Proportional gain (Kp). Higher values increase response to current error.
float (https://learn.microsoft.com/dotnet/api/system.single)	*gainIntegral*	Integral gain (Ki). Higher values eliminate steady-state error faster.
float (https://learn.microsoft.com/dotnet/api/system.single)	*gainDerivative*	Derivative gain (Kd). Higher values dampen oscillations.
float (https://learn.microsoft.com/dotnet/api/system.single)	*outputMin*	Minimum output value.
float (https://learn.microsoft.com/dotnet/api/system.single)	*outputMax*	Maximum output value.

Fields

maxValue

Maximum output value. Output will be clamped to this value.

Declaration

```
public float maxValue
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

minValue

Minimum output value. Output will be clamped to this value.

Declaration

```
public float minValue
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Properties

GainDerivative

The derivative term is proportional to the rate of change of the error

Declaration

```
public float GainDerivative { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

GainIntegral

The integral term is proportional to both the magnitude of the error and the duration of the error

Declaration

```
public float GainIntegral { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

GainProportional

The proportional term produces an output value that is proportional to the current error value

Declaration

```
public float GainProportional { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Remarks

Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change.

IntegralTerm

Adjustment made by considering the accumulated error over time

Declaration

```
public float IntegralTerm { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Remarks

An alternative formulation of the integral action, is the proportional-summation-difference used in discrete-time systems

ProcessVariable

The current value

Declaration

```
public float ProcessVariable { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

ProcessVariableLast

The last reported value (used to calculate the rate of change)

Declaration

```
public float ProcessVariableLast { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

SetPoint

The desired value

Declaration

```
public float SetPoint { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

ControlVariable(float)

The controller output

Declaration

```
public float ControlVariable(float timeSinceLastUpdate)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*timeSinceLastUpdate*	timespan of the elapsed time since the previous time that ControlVariable was called

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Value of the variable that needs to be controlled

Class QuaternionExtensions

Extension methods for advanced Quaternion operations. Provides interpolation methods with control over rotation direction.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ QuaternionExtensions

Namespace: NWH (NWH.html).Common (NWH.Common.html).Utility (NWH.Common.Utility.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class QuaternionExtensions
```

Methods

Add(Quaternion, Quaternion)

Adds two quaternions component-wise.

Declaration

```
public static Quaternion Add(Quaternion p, Quaternion q)
```

Parameters

Type	Name	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*p*	First quaternion.
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*q*	Second quaternion.

Returns

Type	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	Component-wise sum.

Lerp(Quaternion, Quaternion, float, bool)

Linear interpolation between two quaternions with optional short/long path control. Unlike Unity's Quaternion.Lerp, this allows choosing rotation direction.

Declaration

```
public static Quaternion Lerp(Quaternion p, Quaternion q, float t, bool shortWay)
```

Parameters

Type	Name	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*p*	Starting rotation.
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*q*	Target rotation.
float (https://learn.microsoft.com/dotnet/api/system.single)	*t*	Interpolation factor (0 to 1).
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*shortWay*	True for shortest rotation path, false for longest.

Returns

Type	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	Interpolated quaternion.

ScalarMultiply(Quaternion, float)

Multiplies all components of a quaternion by a scalar value.

Declaration

```
public static Quaternion ScalarMultiply(Quaternion input, float scalar)
```

Parameters

Type	Name	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*input*	Input quaternion.
float (https://learn.microsoft.com/dotnet/api/system.single)	*scalar*	Scalar multiplier.

Returns

Type	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	Scaled quaternion.

Slerp(Quaternion, Quaternion, float, bool)

Spherical linear interpolation between two quaternions with optional short/long path control. Provides smooth rotation interpolation with control over rotation direction.

Declaration

```
public static Quaternion Slerp(Quaternion p, Quaternion q, float t, bool shortWay)
```

Parameters

Type	Name	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*p*	Starting rotation.
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	*q*	Target rotation.
float (https://learn.microsoft.com/dotnet/api/system.single)	*t*	Interpolation factor (0 to 1).
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*shortWay*	True for shortest rotation path, false for longest.

Returns

Type	Description
Quaternion (https://docs.unity3d.com/ScriptReference/Quaternion.html)	Interpolated quaternion.

Class UnitConverter

Static utility class for converting between various units of measurement. Includes conversions for distance, speed, fuel efficiency, and angular velocity.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ UnitConverter

Namespace: NWH (NWH.html).Common (NWH.Common.html).Utility (NWH.Common.Utility.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class UnitConverter
```

Methods

AngularVelocityToRPM(float)

Converts angular velocity (rad/s) to rotations per minute.

Declaration

```
public static float AngularVelocityToRPM(float angularVelocity)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*angularVelocity*	Angular velocity in rad/s.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Rotations per minute (RPM).

Inch_To_Meter(float)

Converts inches to meters.

Declaration

```
public static float Inch_To_Meter(float inch)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*inch*	Value in inches.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Value in meters.

KmlToL100km(float)

km/l to l/100km

Declaration

```
public static float KmlToL100km(float kml)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*kml*	Fuel efficiency in km/l.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Fuel efficiency in l/100km.

KmlToMpg(float)

km/l to mpg

Declaration

```
public static float KmlToMpg(float kml)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*kml*	Fuel efficiency in km/l.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Fuel efficiency in mpg.

L100kmToKml(float)

l/100km to km/l

Declaration

```
public static float L100kmToKml(float l100km)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*l100km*	Fuel efficiency in l/100km.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Fuel efficiency in km/l.

L100kmToMpg(float)

l/100km to mpg

Declaration

```
public static float L100kmToMpg(float l100km)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*l100km*	Fuel efficiency in l/100km.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Fuel efficiency in mpg.

Meter_To_Inch(float)

Converts meters to inches.

Declaration

```
public static float Meter_To_Inch(float meters)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*meters*	Value in meters.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Value in inches.

MpgToKml(float)

mpg to km/l

Declaration

```
public static float MpgToKml(float mpg)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*mpg*	Fuel efficiency in mpg.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Fuel efficiency in km/l.

MpgToL100km(float)

mpg to l/100km

Declaration

```
public static float MpgToL100km(float mpg)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*mpg*	Fuel efficiency in mpg.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Fuel efficiency in l/100km.

MphToKph(float)

miles/h to km/h

Declaration

```
public static float MphToKph(float value)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*value*	Speed in mph.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in km/h.

MpsToKph(float)

m/s to km/h

Declaration

```
public static float MpsToKph(float value)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*value*	Speed in m/s.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in km/h.

MpsToMph(float)

m/s to miles/h

Declaration

```
public static float MpsToMph(float value)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*value*	Speed in m/s.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in mph.

RPMToAngularVelocity(float)

Converts rotations per minute to angular velocity (rad/s).

Declaration

```
public static float RPMToAngularVelocity(float RPM)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*RPM*	Rotations per minute.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Angular velocity in rad/s.

Speed_kmhToMph(float)

Converts km/h to mph.

Declaration

```
public static float Speed_kmhToMph(float kmh)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*kmh*	Speed in km/h.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in mph.

Speed_kmhToMs(float)

Converts km/h to m/s.

Declaration

```
public static float Speed_kmhToMs(float kmh)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*kmh*	Speed in km/h.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in m/s.

Speed_mphToKmh(float)

Converts mph to km/h.

Declaration

```
public static float Speed_mphToKmh(float mph)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*mph*	Speed in mph.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in km/h.

Speed_mphToMs(float)

Converts mph to m/s.

Declaration

```
public static float Speed_mphToMs(float mph)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*mph*	Speed in mph.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in m/s.

Speed_msToKph(float)

Converts m/s to km/h.

Declaration

```
public static float Speed_msToKph(float ms)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*ms*	Speed in m/s.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in km/h.

Speed_msToMph(float)

Converts m/s to mph.

Declaration

```
public static float Speed_msToMph(float ms)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*ms*	Speed in m/s.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Speed in mph.

Namespace NWH.Common.Vehicles

Classes

FollowVehicleState (NWH.Common.Vehicles.FollowVehicleState.html)

Automatically enables/disables a GameObject based on parent vehicle's active state. Useful for optimizing performance by disabling effects, audio, or visuals when a vehicle is inactive.

ShowInSettings (NWH.Common.Vehicles.ShowInSettings.html)

Attribute that marks a field to be displayed in runtime settings UI. Allows players to adjust vehicle parameters during gameplay.

ShowInTelemetry (NWH.Common.Vehicles.ShowInTelemetry.html)

Attribute that marks a field or property to be displayed in the runtime telemetry UI. Used for monitoring vehicle parameters during gameplay and debugging.

Vehicle (NWH.Common.Vehicles.Vehicle.html)

Base class for all NWH vehicles including VehiclePhysics2.VehicleController and DWP2.AdvancedShipController.

VehicleReflectionProbe (NWH.Common.Vehicles.VehicleReflectionProbe.html)

Manages vehicle reflection probe settings, switching between baked and realtime modes based on vehicle activity to optimize performance.

Enums

VehicleReflectionProbe.ProbeType (NWH.Common.Vehicles.VehicleReflectionProbe.ProbeType.html)

Type of reflection probe to use.

Class FollowVehicleState

Automatically enables/disables a GameObject based on parent vehicle's active state. Useful for optimizing performance by disabling effects, audio, or visuals when a vehicle is inactive.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ FollowVehicleState

Namespace: NWH (NWH.html).Common (NWH.Common.html).Vehicles (NWH.Common.Vehicles.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DefaultExecutionOrder(21)]
public class FollowVehicleState : MonoBehaviour
```

Remarks

Attach this component to child objects that should only be active when the vehicle is being simulated. When the vehicle is put to sleep (disabled), attached objects are also disabled, saving processing time for effects that wouldn't be visible anyway.

Class ShowInSettings

Attribute that marks a field to be displayed in runtime settings UI. Allows players to adjust vehicle parameters during gameplay.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Attribute (https://learn.microsoft.com/dotnet/api/system.attribute)
    ↳ ShowInSettings

Namespace: NWH (NWH.html).Common (NWH.Common.html).Vehicles (NWH.Common.Vehicles.html)

Assembly: NWH.DWP2.dll

Syntax

```
[AttributeUsage(AttributeTargets.Field)]
public class ShowInSettings : Attribute
```

Constructors

ShowInSettings()

Declaration

```
public ShowInSettings()
```

ShowInSettings(float, float, float)

Creates a settings attribute with specified min, max, and step values.

Declaration

```
public ShowInSettings(float min, float max, float step = 0.1)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*min*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*max*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*step*	

ShowInSettings(string)

Creates a settings attribute with a custom display name.

Declaration

```
public ShowInSettings(string name)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*name*	

ShowInSettings(string, float, float, float)

Creates a settings attribute with custom name and value constraints.

Declaration

```
public ShowInSettings(string name, float min, float max, float step = 0.1)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*name*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*min*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*max*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*step*	

Fields

max

Maximum value for the setting slider.

Declaration

```
public float max
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

min

Minimum value for the setting slider.

Declaration

```
public float min
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

name

Display name for the setting in the UI.

Declaration

```
public string name
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

step

Increment step for the slider. Smaller values allow finer adjustment.

Declaration

```
public float step
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Class ShowInTelemetry

Attribute that marks a field or property to be displayed in the runtime telemetry UI. Used for monitoring vehicle parameters during gameplay and debugging.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Attribute (https://learn.microsoft.com/dotnet/api/system.attribute)
    ↳ ShowInTelemetry

Namespace: NWH (NWH.html).Common (NWH.Common.html).Vehicles (NWH.Common.Vehicles.html)

Assembly: NWH.DWP2.dll

Syntax

```
[AttributeUsage(AttributeTargets.All)]
public class ShowInTelemetry : Attribute
```

Constructors

ShowInTelemetry(float, float, string, string, int)

Creates a ShowInTelemetry attribute with optional parameters.

Declaration

```
public ShowInTelemetry(float min = NaN, float max = NaN, string format = null, string unit =
null, int priority = 1)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*min*	
float (https://learn.microsoft.com/dotnet/api/system.single)	*max*	
string (https://learn.microsoft.com/dotnet/api/system.string)	*format*	
string (https://learn.microsoft.com/dotnet/api/system.string)	*unit*	
int (https://learn.microsoft.com/dotnet/api/system.int32)	*priority*	

Properties

Format

Format string for displaying the value (e.g., "0.00", "0.0").

Declaration

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

Property Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

Max

Maximum value for the field (used for progress bar visualization).

Declaration

```
public float Max { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Min

Minimum value for the field (used for progress bar visualization).

Declaration

```
public float Min { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Priority

Display priority. 0 = highest (always visible), 3 = lowest (detailed info).

Declaration

```
public int Priority { get; set; }
```

Property Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

Unit

Unit of measurement (e.g., "km/h", "RPM", "N", "°").

Declaration

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

Property Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

Class Vehicle

Base class for all NWH vehicles including VehiclePhysics2.VehicleController and DWP2.AdvancedShipController.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ Vehicle
            ↳ AdvancedShipController (NWH.DWP2.ShipController.AdvancedShipController.html)

Namespace: NWH (NWH.html).Common (NWH.Common.html).Vehicles (NWH.Common.Vehicles.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DisallowMultipleComponent]
[RequireComponent(typeof(Rigidbody))]
public abstract class Vehicle : MonoBehaviour
```

Fields

ActiveVehicles

Declaration

```
public static List<Vehicle> ActiveVehicles
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Vehicle (NWH.Common.Vehicles.Vehicle.html)>	

INPUT_DEADZONE

Any input below this value will register as no input.

Declaration

```
public const float INPUT_DEADZONE = 0.02
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

KINDA_SMALL_NUMBER

Like SMALL_NUMBER but a bit bigger.

Declaration

```
public const float KINDA_SMALL_NUMBER = 0.01
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

SMALL_NUMBER

Anything below this can be considered 0.

Declaration

```
public const float SMALL_NUMBER = 1E-05
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

SPEED_DEADZONE

Any speed below this value will register as no speed.

Declaration

```
public const float SPEED_DEADZONE = 0.2
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

isPlayerControllable

True if the vehicle can be driven by the player. False if the vehicle is passive (such as a trailer). A vehicle that has isPlayerControllable can be the ActiveVehicle.

Declaration

```
public bool isPlayerControllable
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

onActiveVehicleChanged

Called when active vehicle is changed. First parameter is the previously active vehicle and the second parameter is the currently active vehicle (at the time of the callback). Params can be null.

Declaration

```
public static UnityEvent<Vehicle, Vehicle> onActiveVehicleChanged
```

Field Value

Type	Description
UnityEvent<Vehicle (NWH.Common.Vehicles.Vehicle.html), Vehicle (NWH.Common.Vehicles.Vehicle.html)>	

onCameraEnterVehicle

Called when the camera enters the vehicle.

Declaration

```
public UnityEvent onCameraEnterVehicle
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

onCameraExitVehicle

Called when the camera exits the vehicle.

Declaration

```
public UnityEvent onCameraExitVehicle
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

onDisable

Called when vehicle is put to sleep.

Declaration

```
[Tooltip("    Called when vehicle is put to sleep.")]
[NonSerialized]
public UnityEvent onDisable
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

onEnable

Called when vehicle is woken up.

Declaration

```
[Tooltip("    Called when vehicle is woken up.")]
[NonSerialized]
public UnityEvent onEnable
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

onMultiplayerStatusChanged

Invoked when MultiplayerIsRemote value gets changed. Is true if remote.

Declaration

```
public UnityEvent<bool> onMultiplayerStatusChanged
```

Field Value

Type	Description
UnityEvent<bool (https://learn.microsoft.com/dotnet/api/system.boolean)>	

vehicleRigidbody

Cached value of vehicle rigidbody.

Declaration

```
[Tooltip("    Cached value of vehicle rigidbody.")]
[NonSerialized]
public Rigidbody vehicleRigidbody
```

Field Value

Type	Description
Rigidbody (https://docs.unity3d.com/ScriptReference/Rigidbody.html)	

vehicleTransform

Cached value of vehicle transform.

Declaration

```
[Tooltip("    Cached value of vehicle transform.")]
[NonSerialized]
public Transform vehicleTransform
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

Properties

ActiveVehicle

Declaration

```
public static Vehicle ActiveVehicle { get; }
```

Property Value

Type	Description
Vehicle (NWH.Common.Vehicles.Vehicle.html)	

AngularVelocity

Cached angular velocity of the vehicle.

Declaration

```
public Vector3 AngularVelocity { get; protected set; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

AngularVelocityMagnitude

Cached angular velocity maginitude of the vehicle.

Declaration

```
public float AngularVelocityMagnitude { get; protected set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

CameraInsideVehicle

True when camera is inside vehicle (cockpit, cabin, etc.). Set by the 'CameraInsideVehicle' component. Used for audio effects.

Declaration

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

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

LocalAcceleration

Cached acceleration in local coordinates (z-forward)

Declaration

```
public Vector3 LocalAcceleration { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

LocalForwardAcceleration

Cached acceleration in forward direction in local coordinates (z-forward).

Declaration

```
public float LocalForwardAcceleration { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

LocalForwardVelocity

Cached velocity in forward direction in local coordinates (z-forward).

Declaration

```
public float LocalForwardVelocity { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

LocalVelocity

Cached velocity in m/s in local coordinates.

Declaration

```
[ShowInTelemetry(NaN, NaN, null, null, 1)]
public Vector3 LocalVelocity { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

MultiplayerIsRemote

True if the vehicle is a client (remote) and not simulated. If true the input is expected to be synced through the network.

Declaration

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

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Speed

Cached speed of the vehicle in the forward direction. ALWAYS POSITIVE. For positive/negative version use SpeedSigned.

Declaration

```
[ShowInTelemetry(NaN, NaN, "0.0", "m/s", 1)]
public float Speed { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

SpeedSigned

Cached speed of the vehicle in the forward direction. Can be positive (forward) or negative (reverse). Equal to LocalForwardVelocity.

Declaration

```
[ShowInTelemetry(NaN, NaN, "0.0", "m/s", 1)]
public float SpeedSigned { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Velocity

Cached velocity of the vehicle in world coordinates.

Declaration

```
public Vector3 Velocity { get; protected set; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

VelocityMagnitude

Cached velocity magnitude of the vehicle in world coordinates.

Declaration

```
public float VelocityMagnitude { get; protected set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

Awake()

Declaration

```
public virtual void Awake()
```

FixedUpdate()

Declaration

```
public virtual void FixedUpdate()
```

OnDisable()

Declaration

```
public virtual void OnDisable()
```

OnEnable()

Declaration

```
public virtual void OnEnable()
```

Class VehicleReflectionProbe

Manages vehicle reflection probe settings, switching between baked and realtime modes based on vehicle activity to optimize performance.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ VehicleReflectionProbe

Namespace: NWH (NWH.html).Common (NWH.Common.html).Vehicles (NWH.Common.Vehicles.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(ReflectionProbe))]
[DefaultExecutionOrder(19)]
public class VehicleReflectionProbe : MonoBehaviour
```

Remarks

Realtime reflection probes are expensive. This component switches to cheaper baked probes when the vehicle is inactive, maintaining visual quality while improving performance. The probe is automatically re-baked when the vehicle becomes inactive to capture the current environment state.

Fields

asleepProbeType

Probe type to use when vehicle is inactive/sleeping. Default is Baked for performance.

Declaration

```
public VehicleReflectionProbe.ProbeType asleepProbeType
```

Field Value

Type	Description
VehicleReflectionProbe (NWH.Common.Vehicles.VehicleReflectionProbe.html).ProbeType (NWH.Common.Vehicles.VehicleReflectionProbe.ProbeType.html)	

awakeProbeType

Probe type to use when vehicle is active. Default is Realtime for accurate reflections.

Declaration

```
public VehicleReflectionProbe.ProbeType awakeProbeType
```

Field Value

Type	Description
VehicleReflectionProbe (NWH.Common.Vehicles.VehicleReflectionProbe.html). ProbeType (NWH.Common.Vehicles.VehicleReflectionProbe.ProbeType.html)	

bakeOnSleep

Automatically bake the probe when vehicle becomes inactive to capture environment state.

Declaration

```
public bool bakeOnSleep
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

bakeOnStart

Bake the probe once on start for initial environment capture.

Declaration

```
public bool bakeOnStart
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Enum VehicleReflectionProbe.ProbeType

Type of reflection probe to use.

Namespace: NWH (NWH.html).Common (NWH.Common.html).Vehicles (NWH.Common.Vehicles.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum VehicleReflectionProbe.ProbeType
```

Fields

Name	Description
Baked	Pre-rendered cubemap, low cost but static.
Realtime	Updates every frame, high cost but accurate.

Namespace NWH.DWP2

Classes

GeometryUtils (NWH.DWP2.GeometryUtils.html)

Utility class for geometric calculations used in buoyancy simulation. Provides methods for calculating centroids, areas, and surface-weighted centers of triangles and quadrilaterals.

UnderwaterFog (NWH.DWP2.UnderwaterFog.html)

Simple utility script that enables Unity fog when the main camera goes underwater. Attach this script to any GameObject in the scene and assign the water surface transform.

Class GeometryUtils

Utility class for geometric calculations used in buoyancy simulation. Provides methods for calculating centroids, areas, and surface-weighted centers of triangles and quadrilaterals.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ GeometryUtils

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class GeometryUtils
```

Methods

CalculateCentroid(Vector3, Vector3, Vector3, Vector3)

Calculates the centroid of a quadrilateral defined by four points.

Declaration

```
public static Vector3 CalculateCentroid(Vector3 pointA, Vector3 pointB, Vector3 pointC, Vect
or3 pointD)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointA*	First corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointB*	Second corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointC*	Third corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointD*	Fourth corner point.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	The centroid position of the quadrilateral.

CalculateQuadrilateralArea(Vector3, Vector3, Vector3, Vector3)

Calculates the surface area of a quadrilateral by dividing it into two triangles.

Declaration

```
public static float CalculateQuadrilateralArea(Vector3 pointA, Vector3 pointB, Vector3 point
C, Vector3 pointD)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointA*	First corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointB*	Second corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointC*	Third corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointD*	Fourth corner point.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Total surface area of the quadrilateral.

CalculateSurfaceWeightedCenter(Vector3, Vector3, Vector3, Vector3)

Calculates the surface-weighted center of a quadrilateral. The center is weighted by the area of the two triangles that make up the quadrilateral.

Declaration

```
public static Vector3 CalculateSurfaceWeightedCenter(Vector3 pointA, Vector3 pointB, Vector3
pointC, Vector3 pointD)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointA*	First corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointB*	Second corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointC*	Third corner point.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointD*	Fourth corner point.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	The surface-weighted center position.

CalculateTriangleArea(Vector3, Vector3, Vector3)

Calculates the surface area of a triangle using the cross product method.

Declaration

```
public static float CalculateTriangleArea(Vector3 pointA, Vector3 pointB, Vector3 pointC)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointA*	First vertex of the triangle.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointB*	Second vertex of the triangle.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*pointC*	Third vertex of the triangle.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Surface area of the triangle.

Class UnderwaterFog

Simple utility script that enables Unity fog when the main camera goes underwater. Attach this script to any GameObject in the scene and assign the water surface transform.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ UnderwaterFog

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class UnderwaterFog : MonoBehaviour
```

Fields

waterTransform

Transform of the water surface. Used to determine if camera is underwater.

Declaration

```
public Transform waterTransform
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

Namespace NWH.DWP2.DemoContent

Classes

CubeGridSpawner (NWH.DWP2.DemoContent.CubeGridSpawner.html)

Demo script that spawns a 3D grid of cubes with WaterObjects attached. Used for stress-testing and demonstrating the buoyancy system with multiple objects.

WaterObjectFromScript (NWH.DWP2.DemoContent.WaterObjectFromScript.html)

An example on how to add WaterObject to an existing object at runtime.

Class CubeGridSpawner

Demo script that spawns a 3D grid of cubes with WaterObjects attached. Used for stress-testing and demonstrating the buoyancy system with multiple objects.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ CubeGridSpawner

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).DemoContent (NWH.DWP2.DemoContent.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class CubeGridSpawner : MonoBehaviour
```

Fields

depth

Spacing between cubes on the Z axis.

Declaration

```
public float depth
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

height

Spacing between cubes on the Y axis.

Declaration

```
public float height
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

width

Spacing between cubes on the X axis.

Declaration

```
public float width
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

xResolution

Number of cubes to spawn along the X axis.

Declaration

```
public int xResolution
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

yResolution

Number of cubes to spawn along the Y axis.

Declaration

```
public int yResolution
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

zResolution

Number of cubes to spawn along the Z axis.

Declaration

```
public int zResolution
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

Class WaterObjectFromScript

An example on how to add WaterObject to an existing object at runtime.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WaterObjectFromScript

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).DemoContent (NWH.DWP2.DemoContent.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class WaterObjectFromScript : MonoBehaviour
```

Namespace NWH.DWP2.MeshDecimation

Classes

Comparer (NWH.DWP2.MeshDecimation.Comparer.html)

Comparer for sorting vertices by their edge collapse cost during mesh decimation.

History (NWH.DWP2.MeshDecimation.History.html)

Tracks mesh decimation history for undo/redo operations during progressive mesh simplification.

MeshDecimate (NWH.DWP2.MeshDecimation.MeshDecimate.html)

Progressive mesh decimation implementation using edge collapse algorithm. Reduces mesh complexity while preserving overall shape and texture coordinates.

Tri (NWH.DWP2.MeshDecimation.Tri.html)

Represents a triangle in the mesh decimation algorithm. Stores triangle vertices, normals, UVs, and adjacency information.

Vert (NWH.DWP2.MeshDecimation.Vert.html)

Represents a vertex in the mesh decimation algorithm. Stores vertex data and relationships for edge collapse operations.

Class Comparer

Comparer for sorting vertices by their edge collapse cost during mesh decimation.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Comparer

Implements

IComparer (https://learn.microsoft.com/dotnet/api/system.collections.icomparer)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MeshDecimation (NWH.DWP2.MeshDecimation.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class Comparer : IComparer
```

Methods

Compare(object, object)

Compares two vertices based on their edge collapse cost.

Declaration

```
public int Compare(object x, object y)
```

Parameters

Type	Name	Description
object (https://learn.microsoft.com/dotnet/api/system.object)	*x*	First vertex to compare.
object (https://learn.microsoft.com/dotnet/api/system.object)	*y*	Second vertex to compare.

Returns

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	-1 if x has lower cost, 0 if equal, 1 if x has higher cost.

Implements

IComparer (https://learn.microsoft.com/dotnet/api/system.collections.icomparer)

Class History

Tracks mesh decimation history for undo/redo operations during progressive mesh simplification.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ History

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MeshDecimation (NWH.DWP2.MeshDecimation.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class History
```

Fields

id

Unique identifier for this history entry.

Declaration

```
public int id
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

removedTriangles

List of triangle indices that were removed during this step.

Declaration

```
public List<int> removedTriangles
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< int (https://learn.microsoft.com/dotnet/api/system.int32)>	

replacedVertex

List of vertex replacement operations performed during this step.

Declaration

```
public List<ArrayList> replacedVertex
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< ArrayList (https://learn.microsoft.com/dotnet/api/system.collections.arraylist)>	

Methods

RemovedTriangle(int)

Records a triangle removal operation.

Declaration

```
public void RemovedTriangle(int f)
```

Parameters

Type	Name	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	*f*	Index of the triangle that was removed.

ReplaceVertex(int, int, int, Vector3, Vector2, int, Vector3, Vector2)

Records a vertex replacement operation including original and new vertex data.

Declaration

```
public void ReplaceVertex(int f, int u, int v, Vector3 normal, Vector2 uv, int newV, Vector3
newNormal, Vector2 newUv)
```

Parameters

Type	Name	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	*f*	Face index.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*u*	Original vertex index in the triangle.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*v*	Vertex ID being replaced.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*normal*	Original vertex normal.
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*uv*	Original texture coordinates.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*newV*	New vertex ID.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*newNormal*	New vertex normal.
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*newUv*	New texture coordinates.

Class MeshDecimate

Progressive mesh decimation implementation using edge collapse algorithm. Reduces mesh complexity while preserving overall shape and texture coordinates.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ MeshDecimate

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MeshDecimation (NWH.DWP2.MeshDecimation.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class MeshDecimate
```

Fields

bRecalculateNormals

Whether to recalculate smooth normals after decimation.

Declaration

```
public bool bRecalculateNormals
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

finalNormals

Final vertex normals after decimation.

Declaration

```
public Vector3[] finalNormals
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

finalTriangles

Final triangle indices after decimation.

Declaration

```
public int[] finalTriangles
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)[]	

finalUVs

Final texture coordinates after decimation.

Declaration

```
public Vector2[] finalUVs
```

Field Value

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)[]	

finalVertices

Final vertex positions after decimation.

Declaration

```
public Vector3[] finalVertices
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

lastTarget

Last target vertex count from progressive mesh operation.

Declaration

```
public int lastTarget
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

lockSelPoint

Whether to lock selected vertices from being collapsed.

Declaration

```
public bool lockSelPoint
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

lodDataSize

Size in bytes of LOD history data for progressive mesh.

Declaration

```
public float lodDataSize
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

preCalculateDone

Whether pre-calculation of decimation data has been completed.

Declaration

```
public bool preCalculateDone
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

ratio

Target ratio of vertices to keep (0.0 to 1.0).

Declaration

```
public float ratio
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

selectedVertices

List of vertices marked as selected (protected from collapse if locked).

Declaration

```
public List<Vector3> selectedVertices
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)>	

smoothAngle

Smoothing angle threshold in degrees for normal calculation.

Declaration

```
public float smoothAngle
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

Calculate(Mesh)

Calculates the decimated mesh at the current ratio. Applies progressive mesh simplification and outputs final geometry arrays. PreCalculate must be called first.

Declaration

```
public void Calculate(Mesh tmpMesh)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*tmpMesh*	Mesh reference (used for validation).

PreCalculate(Mesh)

Pre-calculates all edge collapse operations and builds the progressive mesh history. Must be called once before Calculate can be used.

Declaration

```
public void PreCalculate(Mesh tmpMesh)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*tmpMesh*	Input mesh to decimate.

Class Tri

Represents a triangle in the mesh decimation algorithm. Stores triangle vertices, normals, UVs, and adjacency information.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Tri

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MeshDecimation (NWH.DWP2.MeshDecimation.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class Tri
```

Constructors

Tri(int, Vert, Vert, Vert, Vector2, Vector2, Vector2)

Initializes a new triangle with vertices, texture coordinates, and establishes vertex relationships.

Declaration

```
public Tri(int id, Vert v0, Vert v1, Vert v2, Vector2 uv0, Vector2 uv1, Vector2 uv2)
```

Parameters

Type	Name	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	*id*	Unique identifier for this triangle.
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v0*	First vertex.
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v1*	Second vertex.
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v2*	Third vertex.
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*uv0*	Texture coordinate for first vertex.
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*uv1*	Texture coordinate for second vertex.
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*uv2*	Texture coordinate for third vertex.

Fields

defaultIndex0

Original mesh vertex index for vertex 0.

Declaration

```
public int defaultIndex0
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

defaultIndex1

Original mesh vertex index for vertex 1.

Declaration

```
public int defaultIndex1
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

defaultIndex2

Original mesh vertex index for vertex 2.

Declaration

```
public int defaultIndex2
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

deleted

Whether this triangle has been deleted.

Declaration

```
public bool deleted
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

id

Unique identifier for this triangle.

Declaration

```
public int id
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

normal

Face normal vector.

Declaration

```
public Vector3 normal
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

uv0

Texture coordinate for vertex 0.

Declaration

```
public Vector2 uv0
```

Field Value

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

uv1

Texture coordinate for vertex 1.

Declaration

```
public Vector2 uv1
```

Field Value

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

uv2

Texture coordinate for vertex 2.

Declaration

```
public Vector2 uv2
```

Field Value

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

v0

First vertex of the triangle.

Declaration

```
public Vert v0
```

Field Value

Type	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	

v1

Second vertex of the triangle.

Declaration

```
public Vert v1
```

Field Value

Type	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	

v2

Third vertex of the triangle.

Declaration

```
public Vert v2
```

Field Value

Type	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	

vn0

Vertex normal for vertex 0.

Declaration

```
public Vector3 vn0
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

vn1

Vertex normal for vertex 1.

Declaration

```
public Vector3 vn1
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

vn2

Vertex normal for vertex 2.

Declaration

```
public Vector3 vn2
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Methods

HasVertex(Vert)

Checks if this triangle contains a given vertex.

Declaration

```
public bool HasVertex(Vert v)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v*	Vertex to check.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if the triangle contains the vertex, false otherwise.

RecalculateAvgNormals(float)

Recalculates averaged vertex normals based on adjacent face normals. Only called when normal recalculation is enabled. Smooths normals even at UV seams.

Declaration

```
public void RecalculateAvgNormals(float smoothAngleDot)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*smoothAngleDot*	Dot product threshold for normal smoothing.

RecalculateNormal()

Recalculates the face normal using cross product of edge vectors.

Declaration

```
public void RecalculateNormal()
```

RemoveTriangle(History)

Removes this triangle and updates vertex relationships. Records the removal in history for potential undo operations.

Declaration

```
public void RemoveTriangle(History his)
```

Parameters

Type	Name	Description
History (NWH.DWP2.MeshDecimation.History.html)	*his*	History object to record the removal.

ReplaceVertex(Vert, Vert, Vector2, Vector3, History)

Replaces a vertex in this triangle with a new vertex and updates all relationships. Records the replacement in history for potential undo operations.

Declaration

```
public void ReplaceVertex(Vert vo, Vert vnew, Vector2 newUV, Vector3 newVN, History his)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*vo*	Old vertex to be replaced.
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*vnew*	New vertex to replace with.
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*newUV*	New texture coordinate.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*newVN*	New vertex normal.
History (NWH.DWP2.MeshDecimation.History.html)	*his*	History object to record the replacement.

SetDefaultIndices(int, int, int)

Sets the original mesh vertex indices for this triangle.

Declaration

```
public void SetDefaultIndices(int n0, int n1, int n2)
```

Parameters

Type	Name	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	*n0*	Original index for vertex 0.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*n1*	Original index for vertex 1.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*n2*	Original index for vertex 2.

normalAt(Vert)

Gets the vertex normal for a given vertex.

Declaration

```
public Vector3 normalAt(Vert v)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v*	Vertex to query.

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Vertex normal for the vertex, or zero vector if not found.

setUV(Vert, Vector2)

Sets the texture coordinate for a given vertex.

Declaration

```
public void setUV(Vert v, Vector2 newuv)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v*	Vertex to update.
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	*newuv*	New texture coordinate.

setVN(Vert, Vector3)

Sets the vertex normal for a given vertex.

Declaration

```
public void setVN(Vert v, Vector3 newNormal)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v*	Vertex to update.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*newNormal*	New vertex normal.

uvAt(Vert)

Gets the texture coordinate for a given vertex.

Declaration

```
public Vector2 uvAt(Vert v)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v*	Vertex to query.

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	Texture coordinate for the vertex, or zero vector if not found.

Class Vert

Represents a vertex in the mesh decimation algorithm. Stores vertex data and relationships for edge collapse operations.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Vert

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MeshDecimation (NWH.DWP2.MeshDecimation.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class Vert
```

Constructors

Vert(Vector3, int, bool)

Initializes a new vertex with position, ID, and selection status.

Declaration

```
public Vert(Vector3 position, int id, bool selected)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*position*	3D position of the vertex.
int (https://learn.microsoft.com/dotnet/api/system.int32)	*id*	Unique identifier.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*selected*	Whether the vertex is selected.

Fields

collapse

Target vertex for edge collapse operation.

Declaration

```
public Vert collapse
```

Field Value

Type	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	

cost

Edge collapse cost for this vertex.

Declaration

```
public float cost
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

deleted

Whether this vertex has been deleted.

Declaration

```
public bool deleted
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

face

List of triangles that use this vertex.

Declaration

```
public List<Tri> face
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Tri (NWH.DWP2.MeshDecimation.Tri.html)>	

id

Unique identifier for this vertex.

Declaration

```
public int id
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

neighbor

List of neighboring vertices connected by edges.

Declaration

```
public List<Vert> neighbor
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Vert (NWH.DWP2.MeshDecimation.Vert.html)>	

position

3D position of the vertex.

Declaration

```
public Vector3 position
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

selected

Whether this vertex is marked as selected (prevents collapse if locked).

Declaration

```
public bool selected
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

AddFace(Tri)

Adds a triangle to this vertex's face list.

Declaration

```
public void AddFace(Tri f)
```

Parameters

Type	Name	Description
Tri (NWH.DWP2.MeshDecimation.Tri.html)	*f*	Triangle to add.

AddNeighbor(Vert)

Adds a vertex to the neighbor list if not already present.

Declaration

```
public void AddNeighbor(Vert v)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v*	Vertex to add as neighbor.

IsBorder()

Checks if this vertex is on the mesh border. A vertex is on the border if any of its edges is shared by only one triangle.

Declaration

```
public bool IsBorder()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if the vertex is on the border, false otherwise.

RemoveFace(Tri)

Removes a triangle from this vertex's face list.

Declaration

```
public void RemoveFace(Tri f)
```

Parameters

Type	Name	Description
Tri (NWH.DWP2.MeshDecimation.Tri.html)	*f*	Triangle to remove.

RemoveIfNonNeighbor(Vert)

Removes a vertex from the neighbor list if they no longer share any faces.

Declaration

```
public void RemoveIfNonNeighbor(Vert v)
```

Parameters

Type	Name	Description
Vert (NWH.DWP2.MeshDecimation.Vert.html)	*v*	Vertex to potentially remove from neighbors.

RemoveVert()

Removes this vertex and breaks all neighbor connections.

Declaration

```
public void RemoveVert()
```

Namespace NWH.DWP2.MiConvexHull

Classes

ConvexFace<TVertex, TFace> (NWH.DWP2.MiConvexHull.ConvexFace-2.html)

A convex face representation containing adjacency information.

ConvexHull (NWH.DWP2.MiConvexHull.ConvexHull.html)

Factory class for computing convex hulls.

ConvexHull<TVertex, TFace> (NWH.DWP2.MiConvexHull.ConvexHull-2.html)

Representation of a convex hull.

DefaultConvexFace<TVertex> (NWH.DWP2.MiConvexHull.DefaultConvexFace-1.html)

A default convex face representation that inherits from ConvexFace with self-referencing type parameter.

DefaultTriangulationCell<TVertex> (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)

Default triangulation cell that inherits from TriangulationCell with self-referencing type parameter.

DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html)

"Default" vertex.

DelaunayTriangulation<TVertex, TCell> (NWH.DWP2.MiConvexHull.DelaunayTriangulation-2.html)

Calculation and representation of Delaunay triangulation.

Triangulation (NWH.DWP2.MiConvexHull.Triangulation.html)

Factory class for creating triangulations.

TriangulationCell<TVertex, TCell> (NWH.DWP2.MiConvexHull.TriangulationCell-2.html)

Representation of the triangulation cell. Pretty much the same as ConvexFace, just wanted to distinguish the two. To declare your own face type, use class Face : DelaunayFace(of Vertex, of Face)

Vertex (NWH.DWP2.MiConvexHull.Vertex.html)

Unity-specific vertex implementation for convex hull calculations.

VoronoiEdge<TVertex, TCell> (NWH.DWP2.MiConvexHull.VoronoiEdge-2.html)

A class representing an (undirected) edge of the Voronoi graph.

VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh.html)

A factory class for creating a Voronoi mesh.

VoronoiMesh<TVertex, TCell, TEdge> (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)

A representation of a voronoi mesh.

Interfaces

ITriangulation<TVertex, TCell> (NWH.DWP2.MiConvexHull.ITriangulation-2.html)

Simple interface to unify different types of triangulations in the future.

IVertex (NWH.DWP2.MiConvexHull.IVertex.html)

An interface for a structure with nD position.

Class ConvexFace<TVertex, TFace>

A convex face representation containing adjacency information.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ConvexFace<TVertex, TFace>
    ↳ DefaultConvexFace<TVertex> (NWH.DWP2.MiConvexHull.DefaultConvexFace-1.html)
    ↳ TriangulationCell<TVertex, TCell> (NWH.DWP2.MiConvexHull.TriangulationCell-2.html)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public abstract class ConvexFace<TVertex, TFace> where TVertex : IVertex where TFace : Conve
xFace<TVertex, TFace>
```

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TFace	The type of the t face.

Properties

Adjacency

Adjacency. Array of length "dimension". If F = Adjacency[i] then the vertices shared with F are Vertices[j] where j != i. In the context of triangulation, can be null (indicates the cell is at boundary).

Declaration

```
public TFace[] Adjacency { get; set; }
```

Property Value

Type	Description
TFace[]	The adjacency.

Normal

The normal vector of the face. Null if used in triangulation.

Declaration

```
public double[] Normal { get; set; }
```

Property Value

Type	Description
double (https://learn.microsoft.com/dotnet/api/system.double)[]	The normal.

Vertices

The vertices stored in clockwise order for dimensions 2 - 4, in higher dimensions the order is arbitrary. Unless I accidentally switch some index somewhere in which case the order is CCW. Either way, it is consistent. 3D Normal = (V[1] - V[0]) x (V[2] - V[1]).

Declaration

```
public TVertex[] Vertices { get; set; }
```

Property Value

Type	Description
TVertex[]	The vertices.

Class ConvexHull

Factory class for computing convex hulls.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ConvexHull

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class ConvexHull
```

Methods

Create(IList<double[]>, double)

Creates a convex hull of the input data.

Declaration

```
public static ConvexHull<DefaultVertex, DefaultConvexFace<DefaultVertex>> Create(IList<doubl
e[]> data, double PlaneDistanceTolerance = 1E-10)
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1) <double (https://learn.microsoft.com/dotnet/api/system.double) []>	*data*	The data.
double (https://learn.microsoft.com/dotnet/api/system.double)	*PlaneDistanceTolerance*	The plane distance tolerance.

Returns

Type	Description
ConvexHull (NWH.DWP2.MiConvexHull.ConvexHull-2.html)< DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html), DefaultConvexFace (NWH.DWP2.MiConvexHull.DefaultConvexFace-1.html) <DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html)>>	ConvexHull<DefaultVertex, DefaultConvexFace<DefaultVertex>>.

Create<TVertex>(IList<TVertex>, double)

Creates a convex hull of the input data.

Declaration

```
public static ConvexHull<TVertex, DefaultConvexFace<TVertex>> Create<TVertex>(IList<TVertex>
data, double PlaneDistanceTolerance = 1E-10) where TVertex : IVertex
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1) <TVertex>	*data*	The data.
double (https://learn.microsoft.com/dotnet/api/system.double)	*PlaneDistanceTolerance*	The plane distance tolerance.

Returns

Type	Description
ConvexHull (NWH.DWP2.MiConvexHull.ConvexHull-2.html)<TVertex, DefaultConvexFace (NWH.DWP2.MiConvexHull.DefaultConvexFace-1.html) <TVertex>>	ConvexHull<TVertex, DefaultConvexFace<TVertex>>.

Type Parameters

Name	Description
TVertex	The type of the t vertex.

Create<TVertex, TFace>(IList<TVertex>, double)

Creates a convex hull of the input data.

Declaration

```
public static ConvexHull<TVertex, TFace> Create<TVertex, TFace>(IList<TVertex> data, double
PlaneDistanceTolerance = 1E-10) where TVertex : IVertex where TFace : ConvexFace<TVertex, TF
ace>, new()
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1) <TVertex>	*data*	The data.
double (https://learn.microsoft.com/dotnet/api/system.double)	*PlaneDistanceTolerance*	The plane distance tolerance.

Returns

Type	Description
ConvexHull (NWH.DWP2.MiConvexHull.ConvexHull-2.html)<TVertex, TFace>	ConvexHull<TVertex, TFace>.

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TFace	The type of the t face.

Class ConvexHull<TVertex, TFace>

Representation of a convex hull.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ConvexHull<TVertex, TFace>

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class ConvexHull<TVertex, TFace> where TVertex : IVertex where TFace : ConvexFace<TVe
rtex, TFace>, new()
```

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TFace	The type of the t face.

Properties

Faces

Faces of the convex hull.

Declaration

```
public IEnumerable<TFace> Faces { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<TFace>	The faces.

Points

Vertices of the convex hull.

Declaration

```
public IEnumerable<TVertex> Points { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1) <TVertex>	The points.

Methods

Create(IList<TVertex>, double)

Creates the convex hull.

Declaration

```
public static ConvexHull<TVertex, TFace> Create(IList<TVertex> data, double PlaneDistanceTol
erance)
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1) <TVertex>	*data*	The data.
double (https://learn.microsoft.com/dotnet/api/system.double)	*PlaneDistanceTolerance*	The plane distance tolerance.

Returns

Type	Description
ConvexHull (NWH.DWP2.MiConvexHull.ConvexHull-2.html)<TVertex, TFace>	ConvexHull<TVertex, TFace>.

Exceptions

Type	Condition
ArgumentNullException (https://learn.microsoft.com/dotnet/api/system.argumentnullexception)	The supplied data is null.
ArgumentNullException (https://learn.microsoft.com/dotnet/api/system.argumentnullexception)	data

Class DefaultConvexFace<TVertex>

A default convex face representation that inherits from ConvexFace with self-referencing type parameter.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ConvexFace (NWH.DWP2.MiConvexHull.ConvexFace-2.html)<TVertex, DefaultConvexFace (NWH.DWP2.MiConvexHull.DefaultConvexFace-1.html)<TVertex>>
    ↳ DefaultConvexFace<TVertex>

Inherited Members

ConvexFace<TVertex, DefaultConvexFace<TVertex>>.Adjacency (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Adjacency)
ConvexFace<TVertex, DefaultConvexFace<TVertex>>.Vertices (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Vertices)
ConvexFace<TVertex, DefaultConvexFace<TVertex>>.Normal (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Normal)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DefaultConvexFace<TVertex> : ConvexFace<TVertex, DefaultConvexFace<TVertex>> wh
ere TVertex : IVertex
```

Type Parameters

Name	Description
TVertex	The type of the vertex.

Class DefaultTriangulationCell<TVertex>

Default triangulation cell that inherits from TriangulationCell with self-referencing type parameter.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ConvexFace (NWH.DWP2.MiConvexHull.ConvexFace-2.html)<TVertex, DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)<TVertex>>
    ↳ TriangulationCell (NWH.DWP2.MiConvexHull.TriangulationCell-2.html)<TVertex, DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)<TVertex>>
      ↳ DefaultTriangulationCell<TVertex>

Inherited Members

ConvexFace<TVertex, DefaultTriangulationCell<TVertex>>.Adjacency (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Adjacency)
ConvexFace<TVertex, DefaultTriangulationCell<TVertex>>.Vertices (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Vertices)
ConvexFace<TVertex, DefaultTriangulationCell<TVertex>>.Normal (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Normal)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DefaultTriangulationCell<TVertex> : TriangulationCell<TVertex, DefaultTriangula
tionCell<TVertex>> where TVertex : IVertex
```

Type Parameters

Name	Description
TVertex	The type of the vertex.

Class DefaultVertex

"Default" vertex.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ DefaultVertex

Implements

IVertex (NWH.DWP2.MiConvexHull.IVertex.html)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DefaultVertex : IVertex
```

Properties

Position

Position of the vertex.

Declaration

```
public double[] Position { get; set; }
```

Property Value

Type	Description
double (https://learn.microsoft.com/dotnet/api/system.double)[]	The position.

Implements

IVertex (NWH.DWP2.MiConvexHull.IVertex.html)

See Also

IVertex (NWH.DWP2.MiConvexHull.IVertex.html)

Class DelaunayTriangulation<TVertex, TCell>

Calculation and representation of Delaunay triangulation.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ DelaunayTriangulation<TVertex, TCell>

Implements

ITriangulation (NWH.DWP2.MiConvexHull.ITriangulation-2.html)<TVertex, TCell>

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class DelaunayTriangulation<TVertex, TCell> : ITriangulation<TVertex, TCell> where TV
ertex : IVertex where TCell : TriangulationCell<TVertex, TCell>, new()
```

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TCell	The type of the t cell.

Properties

Cells

Cells of the triangulation.

Declaration

```
public IEnumerable<TCell> Cells { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<TCell>	The cells.

Methods

Create(IList<TVertex>)

Creates the Delaunay triangulation of the input data.

Declaration

```
public static DelaunayTriangulation<TVertex, TCell> Create(IList<TVertex> data)
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	The data.

Returns

Type	Description
DelaunayTriangulation (NWH.DWP2.MiConvexHull.DelaunayTriangulation-2.html) <TVertex, TCell>	DelaunayTriangulation<TVertex, TCell>.

Exceptions

Type	Condition
ArgumentNullException (https://learn.microsoft.com/dotnet/api/system.argumentnullexception)	data

Implements

ITriangulation<TVertex, TCell> (NWH.DWP2.MiConvexHull.ITriangulation-2.html)

See Also

ITriangulation (NWH.DWP2.MiConvexHull.ITriangulation-2.html)<TVertex, TCell>

Interface ITriangulation<TVertex, TCell>

Simple interface to unify different types of triangulations in the future.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public interface ITriangulation<TVertex, TCell> where TVertex : IVertex where TCell : Triang
ulationCell<TVertex, TCell>, new()
```

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TCell	The type of the t cell.

Properties

Cells

Triangulation simplexes. For 2D - triangles, 3D - tetrahedrons, etc ...

Declaration

```
IEnumerable<TCell> Cells { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<TCell>	The cells.

Interface IVertex

An interface for a structure with nD position.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public interface IVertex
```

Properties

Position

Position of the vertex.

Declaration

```
double[] Position { get; }
```

Property Value

Type	Description
double (https://learn.microsoft.com/dotnet/api/system.double)[]	The position.

Class Triangulation

Factory class for creating triangulations.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Triangulation

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class Triangulation
```

Methods

CreateDelaunay(IList<double[]>)

Creates the Delaunay triangulation of the input data.

Declaration

```
public static ITriangulation<DefaultVertex, DefaultTriangulationCell<DefaultVertex>> CreateD
elaunay(IList<double[]> data)
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<double (https://learn.microsoft.com/dotnet/api/system.double)[]>	*data*	The data.

Returns

Type	Description
ITriangulation (NWH.DWP2.MiConvexHull.ITriangulation-2.html)<DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html), DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)<DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html)>>	ITriangulation<DefaultVertex, DefaultTriangulationCell<DefaultVertex>>.

CreateDelaunay<TVertex>(IList<TVertex>)

Creates the Delaunay triangulation of the input data.

Declaration

```
public static ITriangulation<TVertex, DefaultTriangulationCell<TVertex>> CreateDelaunay<TVer
tex>(IList<TVertex> data) where TVertex : IVertex
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	The data.

Returns

Type	Description
ITriangulation (NWH.DWP2.MiConvexHull.ITriangulation-2.html) <TVertex, DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html) <TVertex>>	ITriangulation<TVertex, DefaultTriangulationCell<TVertex>>.

Type Parameters

Name	Description
TVertex	The type of the t vertex.

CreateDelaunay<TVertex, TFace>(IList<TVertex>)

Creates the Delaunay triangulation of the input data.

Declaration

```
public static ITriangulation<TVertex, TFace> CreateDelaunay<TVertex, TFace>(IList<TVertex> d
ata) where TVertex : IVertex where TFace : TriangulationCell<TVertex, TFace>, new()
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	

Returns

Type	Description
ITriangulation (NWH.DWP2.MiConvexHull.ITriangulation-2.html)<TVertex, TFace>	

Type Parameters

Name	Description
TVertex	
TFace	

CreateVoronoi(IList<double[]>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<DefaultVertex, DefaultTriangulationCell<DefaultVertex>, VoronoiEdg
e<DefaultVertex, DefaultTriangulationCell<DefaultVertex>>> CreateVoronoi(IList<double[]> dat
a)
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)< double (https://learn.microsoft.com/dotnet/api/system.double)[]>	*data*	

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)< DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html), DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)< DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html)>, VoronoiEdge (NWH.DWP2.MiConvexHull.VoronoiEdge-2.html)< DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html), DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)< DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html)>>>	

CreateVoronoi<TVertex>(IList<TVertex>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<TVertex, DefaultTriangulationCell<TVertex>, VoronoiEdge<TVertex, D
efaultTriangulationCell<TVertex>>> CreateVoronoi<TVertex>(IList<TVertex> data) where TVertex
: IVertex
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)<TVertex, DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)<TVertex>, VoronoiEdge (NWH.DWP2.MiConvexHull.VoronoiEdge-2.html)<TVertex, DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)<TVertex>>>	

Type Parameters

Name	Description
TVertex	

CreateVoronoi<TVertex, TCell>(IList<TVertex>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<TVertex, TCell, VoronoiEdge<TVertex, TCell>> CreateVoronoi<TVerte
x, TCell>(IList<TVertex> data) where TVertex : IVertex where TCell : TriangulationCell<TVert
ex, TCell>, new()
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)<TVertex, TCell, VoronoiEdge (NWH.DWP2.MiConvexHull.VoronoiEdge-2.html)<TVertex, TCell>>	

Type Parameters

Name	Description
TVertex	
TCell	

CreateVoronoi<TVertex, TCell, TEdge>(IList<TVertex>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<TVertex, TCell, TEdge> CreateVoronoi<TVertex, TCell, TEdge>(IList<
TVertex> data) where TVertex : IVertex where TCell : TriangulationCell<TVertex, TCell>, new
() where TEdge : VoronoiEdge<TVertex, TCell>, new()
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)<TVertex, TCell, TEdge>	

Type Parameters

Name	Description
TVertex	
TCell	
TEdge	

Class TriangulationCell<TVertex, TCell>

Representation of the triangulation cell. Pretty much the same as ConvexFace, just wanted to distinguish the two. To declare your own face type, use class Face : DelaunayFace(of Vertex, of Face)

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ConvexFace (NWH.DWP2.MiConvexHull.ConvexFace-2.html)<TVertex, TCell>
    ↳ TriangulationCell<TVertex, TCell>
      ↳ DefaultTriangulationCell<TVertex> (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html)

Inherited Members

ConvexFace<TVertex, TCell>.Adjacency (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Adjacency)
ConvexFace<TVertex, TCell>.Vertices (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Vertices)
ConvexFace<TVertex, TCell>.Normal (NWH.DWP2.MiConvexHull.ConvexFace-2.html#NWH_DWP2_MiConvexHull_ConvexFace_2_Normal)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public abstract class TriangulationCell<TVertex, TCell> : ConvexFace<TVertex, TCell> where T
Vertex : IVertex where TCell : ConvexFace<TVertex, TCell>
```

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TCell	The type of the t cell.

See Also

ConvexFace (NWH.DWP2.MiConvexHull.ConvexFace-2.html)<TVertex, TFace>

Class Vertex

Unity-specific vertex implementation for convex hull calculations.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Vertex

Implements

IVertex (NWH.DWP2.MiConvexHull.IVertex.html)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class Vertex : IVertex
```

Constructors

Vertex(double, double, double)

Initializes a new vertex with specified coordinates.

Declaration

```
public Vertex(double x, double y, double z)
```

Parameters

Type	Name	Description
double (https://learn.microsoft.com/dotnet/api/system.double)	*x*	X coordinate.
double (https://learn.microsoft.com/dotnet/api/system.double)	*y*	Y coordinate.
double (https://learn.microsoft.com/dotnet/api/system.double)	*z*	Z coordinate.

Vertex(Vector3)

Initializes a new vertex from a Unity Vector3.

Declaration

```
public Vertex(Vector3 ver)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*ver*	Unity Vector3 position.

Properties

Position

Position of the vertex in 3D space.

Declaration

```
public double[] Position { get; set; }
```

Property Value

Type	Description
double (https://learn.microsoft.com/dotnet/api/system.double)[]	

Methods

ToVec()

Converts the vertex position to a Unity Vector3.

Declaration

```
public Vector3 ToVec()
```

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	Unity Vector3 representation of the vertex position.

Implements

IVertex (NWH.DWP2.MiConvexHull.IVertex.html)

Class VoronoiEdge<TVertex, TCell>

A class representing an (undirected) edge of the Voronoi graph.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ VoronoiEdge<TVertex, TCell>

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class VoronoiEdge<TVertex, TCell> where TVertex : IVertex where TCell : Triangulation
Cell<TVertex, TCell>
```

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TCell	The type of the t cell.

Constructors

VoronoiEdge()

Create an instance of the edge.

Declaration

```
public VoronoiEdge()
```

VoronoiEdge(TCell, TCell)

Create an instance of the edge.

Declaration

```
public VoronoiEdge(TCell source, TCell target)
```

Parameters

Type	Name	Description
TCell	*source*	The source.
TCell	*target*	The target.

Properties

Source

Source of the edge.

Declaration

```
public TCell Source { get; }
```

Property Value

Type	Description
TCell	The source.

Target

Target of the edge.

Declaration

```
public TCell Target { get; }
```

Property Value

Type	Description
TCell	The target.

Methods

Equals(object)

...

Declaration

```
public override bool Equals(object obj)
```

Parameters

Type	Name	Description
object (https://learn.microsoft.com/dotnet/api/system.object)	*obj*	The object to compare with the current object.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	`true` if the specified object (https://learn.microsoft.com/dotnet/api/system.object) is equal to this instance; otherwise, `false`.

Overrides

object.Equals(object) (https://learn.microsoft.com/dotnet/api/system.object)

GetHashCode()

...

Declaration

```
public override int GetHashCode()
```

Returns

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	A hash code for this instance, suitable for use in hashing algorithms and data structures like a hash table.

Overrides

object.GetHashCode()

Class VoronoiMesh

A factory class for creating a Voronoi mesh.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ VoronoiMesh

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class VoronoiMesh
```

Methods

Create(IList<double[]>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<DefaultVertex, DefaultTriangulationCell<DefaultVertex>, VoronoiEdg
e<DefaultVertex, DefaultTriangulationCell<DefaultVertex>>> Create(IList<double[]> data)
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1).< double (https://learn.microsoft.com/dotnet/api/system.double)[]>	*data*	The data.

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html) < DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html) , DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html) < DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html) >, VoronoiEdge (NWH.DWP2.MiConvexHull.VoronoiEdge-2.html) < DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html) , DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html) < DefaultVertex (NWH.DWP2.MiConvexHull.DefaultVertex.html) >>>	VoronoiMesh<DefaultVertex, DefaultTriangulationCell<DefaultVertex>, VoronoiEdge<DefaultVertex, DefaultTriangulationCell<DefaultVertex>>>.

Create<TVertex>(IList<TVertex>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<TVertex, DefaultTriangulationCell<TVertex>, VoronoiEdge<TVertex, D
efaultTriangulationCell<TVertex>>> Create<TVertex>(IList<TVertex> data) where TVertex : IVer
tex
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	The data.

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html) <TVertex, DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html) <TVertex>, VoronoiEdge (NWH.DWP2.MiConvexHull.VoronoiEdge-2.html) <TVertex, DefaultTriangulationCell (NWH.DWP2.MiConvexHull.DefaultTriangulationCell-1.html) <TVertex>>>	VoronoiMesh<TVertex, DefaultTriangulationCell<TVertex>, VoronoiEdge<TVertex, DefaultTriangulationCell<TVertex>>>.

Type Parameters

Name	Description
TVertex	The type of the t vertex.

Create<TVertex, TCell>(IList<TVertex>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<TVertex, TCell, VoronoiEdge<TVertex, TCell>> Create<TVertex, TCell
>(IList<TVertex> data) where TVertex : IVertex where TCell : TriangulationCell<TVertex, TCel
l>, new()
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	The data.

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)<TVertex, TCell, VoronoiEdge (NWH.DWP2.MiConvexHull.VoronoiEdge-2.html)<TVertex, TCell>>	VoronoiMesh<TVertex, TCell, VoronoiEdge<TVertex, TCell>>.

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TCell	The type of the t cell.

Create<TVertex, TCell, TEdge>(IList<TVertex>)

Create the voronoi mesh.

Declaration

```
public static VoronoiMesh<TVertex, TCell, TEdge> Create<TVertex, TCell, TEdge>(IList<TVertex
> data) where TVertex : IVertex where TCell : TriangulationCell<TVertex, TCell>, new() where
TEdge : VoronoiEdge<TVertex, TCell>, new()
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	The data.

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)<TVertex, TCell, TEdge>	VoronoiMesh<TVertex, TCell, TEdge>.

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TCell	The type of the t cell.
TEdge	The type of the t edge.

Class VoronoiMesh<TVertex, TCell, TEdge>

A representation of a voronoi mesh.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ VoronoiMesh<TVertex, TCell, TEdge>

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).MiConvexHull (NWH.DWP2.MiConvexHull.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class VoronoiMesh<TVertex, TCell, TEdge> where TVertex : IVertex where TCell : Triang
ulationCell<TVertex, TCell>, new() where TEdge : VoronoiEdge<TVertex, TCell>, new()
```

Type Parameters

Name	Description
TVertex	The type of the t vertex.
TCell	The type of the t cell.
TEdge	The type of the t edge.

Properties

Edges

Edges connecting the cells. The same information can be retrieved Cells' Adjacency.

Declaration

```
public IEnumerable<TEdge> Edges { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<TEdge>	The edges.

Vertices

Vertices of the diagram.

Declaration

```
public IEnumerable<TCell> Vertices { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<TCell>	The vertices.

Methods

Create(IList<TVertex>)

Create a Voronoi diagram of the input data.

Declaration

```
public static VoronoiMesh<TVertex, TCell, TEdge> Create(IList<TVertex> data)
```

Parameters

Type	Name	Description
IList (https://learn.microsoft.com/dotnet/api/system.collections.generic.ilist-1)<TVertex>	*data*	The data.

Returns

Type	Description
VoronoiMesh (NWH.DWP2.MiConvexHull.VoronoiMesh-3.html)<TVertex, TCell, TEdge>	VoronoiMesh<TVertex, TCell, TEdge>.

Exceptions

Type	Condition
ArgumentNullException (https://learn.microsoft.com/dotnet/api/system.argumentnullexception)	data

Namespace NWH.DWP2.SailController

Classes

HullWindApplicator (NWH.DWP2.SailController.HullWindApplicator.html)

Applies wind drag forces to the hull of a sailing vessel based on the global wind conditions. Calculates the projected area of the hull facing the wind and applies an appropriate drag force. Does not account for the waterline - dimensions should represent only the hull portion above water. Requires a WindGenerator to be present in the scene.

SailController (NWH.DWP2.SailController.SailController.html)

Manages sail physics by calculating and applying lift and drag forces based on sail geometry, orientation, and wind conditions. Uses four corner transforms (a, b, c, d) to define sail shape, enabling dynamic sail configurations including furling and multi-part sails. Calculates apparent wind from true wind and vessel velocity, then applies aerodynamic forces at the sail's surface-weighted center. The corner transforms follow a clockwise pattern: a (bottom left/front), b (top left/front), c (top right/rear), d (bottom right/rear). Corner transforms can be attached to different parents to enable sail furling or complex rigging systems. Use SailRotator for user-controlled sail rotation. Requires a WindGenerator in the scene and a SailPreset for aerodynamic coefficients.

SailPreset (NWH.DWP2.SailController.SailPreset.html)

Defines the aerodynamic characteristics of a sail through lift and drag coefficient curves. Contains the relationship between angle of attack and force generation. Different sail types (square, lateen, bermuda, etc.) can be represented by different presets. Can be created via Assets > Create > NWH > DWP2 > SailPreset.

SailRotator (NWH.DWP2.SailController.SailRotator.html)

Rotates a transform based on player input from the AdvancedShipController. Used to control sail orientation in response to the RotateSail input axis. Typically attached to a sail boom or mast that rotates horizontally. Requires an AdvancedShipController component on a parent GameObject.

WindGenerator (NWH.DWP2.SailController.WindGenerator.html)

Generates procedural wind with randomized gusts for sail simulation. Creates realistic wind conditions by varying speed and direction over time. Provides a singleton instance accessible to all sail and wind-related components. Wind direction uses world coordinates where 0 degrees points along the positive Z-axis. Only one WindGenerator should exist per scene.

WindIndicator (NWH.DWP2.SailController.WindIndicator.html)

Visual indicator that rotates to show wind direction. Displays apparent wind when attached as a child of a SailController. Displays true wind from the WindGenerator when not under a SailController. Useful for debugging sail setup and helping players understand wind conditions. The transform's forward direction will point into the wind.

Class HullWindApplicator

Applies wind drag forces to the hull of a sailing vessel based on the global wind conditions. Calculates the projected area of the hull facing the wind and applies an appropriate drag force. Does not account for the waterline - dimensions should represent only the hull portion above water. Requires a WindGenerator to be present in the scene.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ HullWindApplicator

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).SailController (NWH.DWP2.SailController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class HullWindApplicator : MonoBehaviour
```

Fields

dimensions

Physical dimensions of the hull above the waterline. X represents width, Y represents height, Z represents length. These dimensions are used to calculate the projected area when wind hits the hull from different angles.

Declaration

```
[Tooltip("The dimensions of the object (x = width, y = height, z = length).")]
[SerializeField]
public Vector3 dimensions
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

dragCoefficient

Drag coefficient applied to the wind force calculation. Higher values result in stronger wind resistance. Typical values range from 0.5 to 2.0 depending on hull shape and surface characteristics.

Declaration

```
[Tooltip("The drag coefficient of the object.")]
[SerializeField]
public float dragCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Class SailController

Manages sail physics by calculating and applying lift and drag forces based on sail geometry, orientation, and wind conditions. Uses four corner transforms (a, b, c, d) to define sail shape, enabling dynamic sail configurations including furling and multi-part sails. Calculates apparent wind from true wind and vessel velocity, then applies aerodynamic forces at the sail's surface-weighted center. The corner transforms follow a clockwise pattern: a (bottom left/front), b (top left/front), c (top right/rear), d (bottom right/rear). Corner transforms can be attached to different parents to enable sail furling or complex rigging systems. Use SailRotator for user-controlled sail rotation. Requires a WindGenerator in the scene and a SailPreset for aerodynamic coefficients.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ SailController

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).SailController (NWH.DWP2.SailController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class SailController : MonoBehaviour
```

Fields

a

Bottom left corner of the sail for square sails, or bottom front corner for triangular sails. First point in the clockwise corner definition pattern. Can be attached to any parent transform to enable dynamic sail configurations.

Declaration

```
[Tooltip("Bottom left sail corner if square sail.\r\nOtherwise bottom front.")]
public Transform a
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

airDensity

Air density in kg/m³ used in aerodynamic force calculations. Standard sea level value is 1.225 kg/m³. Can be adjusted as a multiplier to scale all sail forces uniformly without modifying the preset.

Declaration

```
[Tooltip("The air density. Can also be used\r\nas a force coefficient as this affects both l
ift and drag forces equally.")]
public float airDensity
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

b

Top left corner of the sail for square sails, or top front corner for triangular sails. Second point in the clockwise corner definition pattern. Can be attached to any parent transform to enable dynamic sail configurations.

Declaration

```
[Tooltip("Top left sail corner if square sail.\r\nOtherwise top front.")]
public Transform b
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

c

Top right corner of the sail for square sails, or top rear corner for triangular sails. Third point in the clockwise corner definition pattern. Can be attached to any parent transform to enable dynamic sail configurations.

Declaration

```
[Tooltip("Top right sail corner if square sail.\r\nOtherwise top rear.")]
public Transform c
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

d

Bottom right corner of the sail for square sails, or bottom rear corner for triangular sails. Fourth point in the clockwise corner definition pattern. Can be attached to any parent transform to enable dynamic sail configurations.

Declaration

```
[Tooltip("Bottom right sail corner if square sail.\r\nOtherwise bottom rear.")]
public Transform d
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

sailPreset

Defines the aerodynamic characteristics of the sail through lift and drag coefficient curves. Contains the relationship between angle of attack and force coefficients.

Declaration

```
public SailPreset sailPreset
```

Field Value

Type	Description
SailPreset (NWH.DWP2.SailController.SailPreset.html)	

Properties

AngleOfAttack

Angle in degrees between the sail's forward direction and the apparent wind direction. Measured on the horizontal plane using Vector3.up as the reference axis. Used to determine lift and drag coefficients from the SailPreset curves. Positive values indicate wind from the starboard side, negative values from port side.

Declaration

```
public float AngleOfAttack { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

ApparentWind

Wind experienced by the sail relative to the moving vessel. Calculated as the vector difference between true wind and vessel velocity. This is the effective wind that generates aerodynamic forces on the sail. Measured in meters per second.

Declaration

```
public Vector3 ApparentWind { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

SailArea

Total surface area of the sail in square meters. Calculated from the quadrilateral formed by corner points a, b, c, and d. Used in aerodynamic force calculations along with dynamic pressure.

Declaration

```
public float SailArea { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

SailCenter

Surface-weighted center point of the sail in world space. All aerodynamic forces are applied at this position. Calculated from the quadrilateral formed by corner points a, b, c, and d.

Declaration

```
public Vector3 SailCenter { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

SailForce

Total aerodynamic force vector applied to the vessel at the sail center point. Combines lift and drag forces based on sail geometry, apparent wind, and aerodynamic coefficients. Measured in Newtons.

Declaration

```
public Vector3 SailForce { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

SailForward

Forward direction vector of the sail in world space. Determined by the vector from corner point d to corner point a. Used to calculate the angle of attack relative to the apparent wind.

Declaration

```
public Vector3 SailForward { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

SailRight

Right direction vector of the sail in world space. Derived from the cross product of SailUp and SailForward. Defines the direction of lift force generation perpendicular to the sail plane.

Declaration

```
public Vector3 SailRight { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

SailUp

Up direction vector of the sail in world space. Calculated from the average of the top edge to the average of the bottom edge. Used to determine sail orientation and compensate for heel angle.

Declaration

```
public Vector3 SailUp { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

ShipVelocity

Current velocity of the vessel's Rigidbody in world space. Used to calculate apparent wind by combining with true wind. Measured in meters per second.

Declaration

```
public Vector3 ShipVelocity { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

TrueWind

True wind vector from the WindGenerator, representing the actual environmental wind. Does not account for vessel motion. Measured in meters per second.

Declaration

```
public Vector3 TrueWind { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Class SailPreset

Defines the aerodynamic characteristics of a sail through lift and drag coefficient curves. Contains the relationship between angle of attack and force generation. Different sail types (square, lateen, bermuda, etc.) can be represented by different presets. Can be created via Assets > Create > NWH > DWP2 > SailPreset.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ ScriptableObject (https://docs.unity3d.com/ScriptReference/ScriptableObject.html)
      ↳ SailPreset

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).SailController (NWH.DWP2.SailController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[CreateAssetMenu(fileName = "SailPreset", menuName = "NWH/DWP2/SailPreset", order = 1)]
public class SailPreset : ScriptableObject
```

Fields

description

Optional description of the sail type and its characteristics.

Declaration

```
public string description
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

dragCoefficientVsAoACurve

Defines how drag coefficient varies with angle of attack. X-axis represents angle of attack in degrees (-180 to 180). Y-axis represents the drag coefficient multiplier. Drag acts in the direction of the apparent wind.

Declaration

```
public AnimationCurve dragCoefficientVsAoACurve
```

Field Value

Type	Description
AnimationCurve (https://docs.unity3d.com/ScriptReference/AnimationCurve.html)	

dragScale

Global multiplier for all drag forces. Values greater than 1 increase drag, values less than 1 reduce it. Does not affect the shape of the drag curve.

Declaration

```
public float dragScale
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

liftCoefficientVsAoACurve

Defines how lift coefficient varies with angle of attack. X-axis represents angle of attack in degrees (-180 to 180). Y-axis represents the lift coefficient multiplier. Lift acts perpendicular to the sail plane.

Declaration

```
public AnimationCurve liftCoefficientVsAoACurve
```

Field Value

Type	Description
AnimationCurve (https://docs.unity3d.com/ScriptReference/AnimationCurve.html)	

liftScale

Global multiplier for all lift forces. Values greater than 1 increase lift, values less than 1 reduce it. Does not affect the shape of the lift curve.

Declaration

```
public float liftScale
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Class SailRotator

Rotates a transform based on player input from the AdvancedShipController. Used to control sail orientation in response to the RotateSail input axis. Typically attached to a sail boom or mast that rotates horizontally. Requires an AdvancedShipController component on a parent GameObject.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ SailRotator

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).SailController (NWH.DWP2.SailController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class SailRotator : MonoBehaviour
```

Fields

rotationAxis

Local axis around which the transform rotates. Default (0, 1, 0) rotates around the Y-axis. Use negative values to reverse rotation direction. Magnitude is ignored - only direction matters.

Declaration

```
[Tooltip("Rotation axis of this transform.\r\nUse -1 to reverse the rotation.")]
public Vector3 rotationAxis
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

rotationSpeed

Maximum rotation speed in degrees per second when input is at full deflection. Higher values allow faster sail adjustment. Combined with rotationAxis and player input to determine final rotation.

Declaration

```
[Tooltip("Rotation speed of this transform in deg/s.\r\nMultiplied by the rotationAxis to ge
t the final rotation.")]
public float rotationSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Class WindGenerator

Generates procedural wind with randomized gusts for sail simulation. Creates realistic wind conditions by varying speed and direction over time. Provides a singleton instance accessible to all sail and wind-related components. Wind direction uses world coordinates where 0 degrees points along the positive Z-axis. Only one WindGenerator should exist per scene.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WindGenerator

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).SailController (NWH.DWP2.SailController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class WindGenerator : MonoBehaviour
```

Fields

Instance

Singleton instance accessible globally. Used by SailController and HullWindApplicator to get current wind conditions.

Declaration

```
public static WindGenerator Instance
```

Field Value

Type	Description
WindGenerator (NWH.DWP2.SailController.WindGenerator.html)	

baseDirection

Primary wind direction in degrees on the horizontal plane. 0 degrees corresponds to positive Z-axis (north in Unity coordinates). Rotation follows the right-hand rule around the Y-axis. Wind will vary around this base direction within the limits of maxDirectionVariation.

Declaration

```
[Tooltip("Base wind direction in degrees with 0 degrees indicating Z-forward ('north').")]
public float baseDirection
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

baseSpeed

Average wind speed in meters per second. Wind will vary around this base speed within the limits of maxSpeedVariation. Typical sailing winds range from 5 m/s (light breeze) to 20 m/s (strong wind).

Declaration

```
[Tooltip("Base wind speed in m/s.")]
public float baseSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxDirectionVariation

Maximum angular deviation from baseDirection during wind gusts. Measured in degrees. Higher values create more unpredictable wind shifts. Realistic values range from 15 to 45 degrees.

Declaration

```
[Tooltip("Maximum possible variation of the direction in degrees from the\r\nbaseDirectio
n.")]
public float maxDirectionVariation
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxSpeedVariation

Maximum speed deviation from baseSpeed during wind gusts. Measured in meters per second. Higher values create stronger variations between lulls and gusts. Can be positive or negative relative to base speed.

Declaration

```
[Tooltip("Maximum possible variation/deviation of the wind from the baseSpeed.")]
public float maxSpeedVariation
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxVariationInterval

Longest possible duration between wind condition changes. Measured in seconds. New wind conditions are selected randomly between minVariationInterval and this value.

Declaration

```
[Tooltip("Maximum interval between the wind variations / changes.")]
public float maxVariationInterval
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

minVariationInterval

Shortest possible duration between wind condition changes. Measured in seconds. New wind conditions are selected randomly between this value and maxVariationInterval.

Declaration

```
[Tooltip("Minimum interval between the wind variations / changes.")]
public float minVariationInterval
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Properties

CurrentDirection

Current wind direction in degrees on the horizontal plane. Smoothly interpolates between randomized target directions. 0 degrees corresponds to positive Z-axis.

Declaration

```
public float CurrentDirection { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

CurrentSpeed

Current wind speed magnitude in meters per second. Smoothly interpolates between randomized target speeds.

Declaration

```
public float CurrentSpeed { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

CurrentWind

Current wind vector in world space. Combines CurrentDirection and CurrentSpeed into a directional velocity vector. Updated every FixedUpdate with smooth damping between gust transitions. Measured in meters per second.

Declaration

```
public Vector3 CurrentWind { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Class WindIndicator

Visual indicator that rotates to show wind direction. Displays apparent wind when attached as a child of a SailController. Displays true wind from the WindGenerator when not under a SailController. Useful for debugging sail setup and helping players understand wind conditions. The transform's forward direction will point into the wind.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WindIndicator

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).SailController (NWH.DWP2.SailController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class WindIndicator : MonoBehaviour
```

Namespace NWH.DWP2.ShipController

Classes

AdvancedShipController (NWH.DWP2.ShipController.AdvancedShipController.html)

Main controller for ships, boats and other water vessels. Manages propulsion through engines and propellers, steering via rudders, and additional control through bow/stern thrusters. Includes stabilization systems, anchor functionality, and multiplayer support. Can be used for everything from small boats to large ships.

Anchor (NWH.DWP2.ShipController.Anchor.html)

Approximates the behavior of a real anchor by keeping the object near the anchored position, but allowing for some movement.

Engine (NWH.DWP2.ShipController.Engine.html)

Represents a ship engine and propeller system that generates thrust. Simulates RPM-based thrust generation with start/stop sequences and sound effects. Includes propeller efficiency modeling through thrust curves and support for reverse thrust. Multiple engines can be used per ship for differential thrust control.

InputManagerShipInputProvider (NWH.DWP2.ShipController.InputManagerShipInputProvider.html)

Ship input provider for Unity's legacy Input Manager. Handles keyboard and gamepad input through Input.GetAxis and Input.GetButton. Requires proper axis and button setup in Project Settings > Input Manager. Falls back to hardcoded keys if input bindings are not configured.

InputSystemShipInputProvider (NWH.DWP2.ShipController.InputSystemShipInputProvider.html)

Ship input provider for Unity's new Input System. Uses ShipInputActions asset for input binding configuration. Supports keyboard, gamepad, and custom input devices through the Input System. Input bindings can be modified by editing the ShipInputActions asset.

MobileShipInputProvider (NWH.DWP2.ShipController.MobileShipInputProvider.html)

Ship input provider for mobile devices using on-screen UI controls. Reads input from Unity UI Sliders and MobileInputButtons placed in the scene. All control assignments are optional - assign only the controls you need for your ship.

Rudder (NWH.DWP2.ShipController.Rudder.html)

Represents a ship rudder that rotates based on steering input. Provides visual rotation only - actual steering forces are applied by WaterObject components on the rudder itself. Multiple rudders can be used on a single ship.

ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html)

Provides programmatic access to UnityEngine.InputSystem.InputActionAsset, UnityEngine.InputSystem.InputActionMap, UnityEngine.InputSystem.InputAction and UnityEngine.InputSystem.InputControlScheme instances defined in asset "Packages/com.nwh.dynamicwaterphysics/Runtime/ShipController/Input/InputProviders/InputSystemProvider/ShipInputActions.inputactions".

ShipInputHandler (NWH.DWP2.ShipController.ShipInputHandler.html)

Manages ship input by retrieving values from active ShipInputProviders and storing them in ShipInputStates. Automatically polls all registered input providers and combines their inputs. Can be disabled for manual input control via scripting or AI.

ShipInputProvider (NWH.DWP2.ShipController.ShipInputProvider.html)

Base class for all ship input providers. Inherit from this to create custom input systems for ship controls. Provides virtual methods for all ship control inputs including throttle, steering, thrusters, and submarine depth control.

Sink (NWH.DWP2.ShipController.Sink.html)

Simulates a ship taking on water and sinking by gradually increasing mass over time. Works with VariableCenterOfMass component to affect buoyancy and cause the ship to sink. Can be triggered to simulate hull breaches or flooding.

Submarine (NWH.DWP2.ShipController.Submarine.html)

Enables submarine functionality by controlling ballast mass for diving and surfacing. Manages depth control through variable mass and can automatically maintain horizontal orientation. Works with VariableCenterOfMass to adjust buoyancy for depth changes.

Thruster (NWH.DWP2.ShipController.Thruster.html)

Lateral thruster for sideways ship movement. Bow thrusters are mounted at the front for tight turning and docking, stern thrusters at the rear. Multiple thrusters of each type can be added for increased lateral control.

Structs

ShipInputActions.ShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html)

Provides access to input actions defined in input action map "ShipControls".

ShipInputStates (NWH.DWP2.ShipController.ShipInputStates.html)

Container for all ship input states. Stores current values for all ship controls including throttle, steering, thrusters, and special inputs.

Interfaces

ShipInputActions.IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html)

Interface to implement callback methods for all input action callbacks associated with input actions defined by "ShipControls" which allows adding and removing callbacks.

Enums

Engine.RotationDirection (NWH.DWP2.ShipController.Engine.RotationDirection.html)

Visual rotation direction of the propeller.

Engine.State (NWH.DWP2.ShipController.Engine.State.html)

Current operational state of the engine.

Engine.ThrottleBinding (NWH.DWP2.ShipController.Engine.ThrottleBinding.html)

Which throttle input channel this engine responds to. Allows independent control of multiple engines on the same ship.

Thruster.RotationDirection (NWH.DWP2.ShipController.Thruster.RotationDirection.html)

Visual rotation direction of the thruster propeller.

Thruster.ThrusterPosition (NWH.DWP2.ShipController.Thruster.ThrusterPosition.html)

Position of the thruster on the ship. Determines which input controls it and the direction of thrust application.

Class AdvancedShipController

Main controller for ships, boats and other water vessels. Manages propulsion through engines and propellers, steering via rudders, and additional control through bow/stern thrusters. Includes stabilization systems, anchor functionality, and multiplayer support. Can be used for everything from small boats to large ships.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ Vehicle (NWH.Common.Vehicles.Vehicle.html)
            ↳ AdvancedShipController

Inherited Members

Vehicle.INPUT_DEADZONE (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_INPUT_DEADZONE)
Vehicle.SPEED_DEADZONE (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_SPEED_DEADZONE)
Vehicle.SMALL_NUMBER (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_SMALL_NUMBER)
Vehicle.KINDA_SMALL_NUMBER (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_KINDA_SMALL_NUMBER)
Vehicle.ActiveVehicles (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_ActiveVehicles)
Vehicle.onActiveVehicleChanged (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_onActiveVehicleChanged)
Vehicle.isPlayerControllable (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_isPlayerControllable)
Vehicle.vehicleRigidbody (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_vehicleRigidbody)
Vehicle.vehicleTransform (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_vehicleTransform)
Vehicle.ActiveVehicle (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_ActiveVehicle)
Vehicle.Awake() (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_Awake)
Vehicle.CameraInsideVehicle (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_CameraInsideVehicle)
Vehicle.onCameraEnterVehicle (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_onCameraEnterVehicle)
Vehicle.onCameraExitVehicle (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_onCameraExitVehicle)
Vehicle.onDisable (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_onDisable)
Vehicle.onEnable (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_onEnable)
Vehicle.MultiplayerIsRemote (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_MultiplayerIsRemote)
Vehicle.onMultiplayerStatusChanged (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_onMultiplayerStatusChanged)
Vehicle.LocalAcceleration (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_LocalAcceleration)
Vehicle.LocalForwardAcceleration (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_LocalForwardAcceleration)
Vehicle.LocalForwardVelocity (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_LocalForwardVelocity)
Vehicle.LocalVelocity (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_LocalVelocity)

Vehicle.Speed (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_Speed)
Vehicle.SpeedSigned (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_SpeedSigned)
Vehicle.Velocity (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_Velocity)
Vehicle.VelocityMagnitude
(NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_VelocityMagnitude)
Vehicle.AngularVelocity (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_AngularVelocity)
Vehicle.AngularVelocityMagnitude
(NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_AngularVelocityMagnitude)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(Anchor))]
[Serializable]
[DefaultExecutionOrder(90)]
public class AdvancedShipController : Vehicle
```

Fields

dropAnchorWhenInactive

Should the anchor be dropped when the ship is deactivated?

Declaration

```
[Tooltip("Should the anchor be dropped when the ship is deactivated?")]
public bool dropAnchorWhenInactive
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

engines

Ship's engines.

Declaration

```
[Tooltip("List of engines. Each engine is a propulsion system in itself consisting of the en
gine and the propeller.")]
[SerializeField]
public List<Engine> engines
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Engine (NWH.DWP2.ShipController.Engine.html)>	

input

Class that handles all of the user input.

Declaration

```
[SerializeField]
[Tooltip("Class that handles all of the user input.")]
public ShipInputHandler input
```

Field Value

Type	Description
ShipInputHandler (NWH.DWP2.ShipController.ShipInputHandler.html)	

maxStabilizationTorqueAngle

Angle at which roll stabilization torque reaches maximum.

Declaration

```
[Tooltip("Angle at which roll stabilization torque reaches maximum.")]
public float maxStabilizationTorqueAngle
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

onShipInitialized

Called after ship initialization.

Declaration

```
[Tooltip("Called after ship initialization.")]
public UnityEvent onShipInitialized
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

pitchStabilizationMaxTorque

Torque that will be applied to stabilize pitch when the ship pitch angle reaches maxStabilizationTorqueAngle.

Declaration

```
[Tooltip("Torque that will be applied to stabilize pitch when the ship pitch angle reaches m
axStabilizationTorqueAngle.")]
public float pitchStabilizationMaxTorque
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

referenceWaterObject

Declaration

```
[FormerlySerializedAs("ReferenceWaterObject")]
public WaterObject referenceWaterObject
```

Field Value

Type	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	

rollStabilizationMaxTorque

Torque that will be applied to stabilize roll when the ship roll angle reaches maxStabilizationTorqueAngle.

Declaration

```
[Tooltip("Torque that will be applied to stabilize roll when the ship roll angle reaches max
StabilizationTorqueAngle.")]
public float rollStabilizationMaxTorque
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

rudders

Ship's rudders.

Declaration

```
[SerializeField]
[Tooltip("Ship's rudders.")]
public List<Rudder> rudders
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Rudder (NWH.DWP2.ShipController.Rudder.html)>	

stabilizePitch

Should the ship pitch be stabilized?

Declaration

```
[Tooltip("Should the ship pitch be stabilized?")]
public bool stabilizePitch
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

stabilizeRoll

Should the ship roll be stabilized?

Declaration

```
[Tooltip("Should the ship roll be stabilized?")]
public bool stabilizeRoll
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

thrusters

Bow or stern thrusters that a ship has.

Declaration

```
[SerializeField]
[Tooltip("Bow or stern thrusters that a ship has.")]
public List<Thruster> thrusters
```

Field Value

Type	Description
List (https://learn.microsoft.com/dotnet/api/system.collections.generic.list-1)< Thruster (NWH.DWP2.ShipController.Thruster.html)>	

weighAnchorWhenActive

Should the anchor be weighed/lifted when the ship is activated?

Declaration

```
[Tooltip("Should the anchor be weighed/lifted when the ship is activated?")]
public bool weighAnchorWhenActive
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Properties

Anchor

Anchor script.

Declaration

```
public Anchor Anchor { get; }
```

Property Value

Type	Description
Anchor (NWH.DWP2.ShipController.Anchor.html)	

SpeedKnots

Speed in knots.

Declaration

```
public float SpeedKnots { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

FixedUpdate()

Declaration

```
public override void FixedUpdate()
```

Overrides

Vehicle.FixedUpdate() (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_FixedUpdate)

GetPitchAngle()

Returns pitch angle of the ship in degrees. Positive values indicate bow up, negative values indicate bow down.

Declaration

```
public float GetPitchAngle()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Pitch angle in degrees.

GetRollAngle()

Returns roll angle of the ship in degrees. Positive values indicate starboard side down, negative values indicate port side down.

Declaration

```
public float GetRollAngle()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Roll angle in degrees.

OnDisable()

Declaration

```
public override void OnDisable()
```

Overrides

Vehicle.OnDisable() (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_OnDisable)

OnEnable()

Declaration

```
public override void OnEnable()
```

Overrides

Vehicle.OnEnable() (NWH.Common.Vehicles.Vehicle.html#NWH_Common_Vehicles_Vehicle_OnEnable)

Start()

Declaration

```
public void Start()
```

Update()

Declaration

```
public void Update()
```

See Also

Engine (NWH.DWP2.ShipController.Engine.html)
Rudder (NWH.DWP2.ShipController.Rudder.html)
Thruster (NWH.DWP2.ShipController.Thruster.html)
Anchor
(NWH.DWP2.ShipController.AdvancedShipController.html#NWH_DWP2_ShipController_AdvancedShipController_Anchor)
ShipInputHandler (NWH.DWP2.ShipController.ShipInputHandler.html)
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)
Vehicle (NWH.Common.Vehicles.Vehicle.html)

Class Anchor

Approximates the behavior of a real anchor by keeping the object near the anchored position, but allowing for some movement.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ Anchor

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class Anchor : MonoBehaviour
```

Fields

dragForce

Maximum force that can be applied to anchor before it starts to drag.

Declaration

```
[Tooltip("Maximum force that can be applied to anchor before it starts to drag.")]
public float dragForce
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

dropOnStart

Should the anchor be dropped at start?

Declaration

```
[Tooltip("Should the anchor be dropped at start?")]
public bool dropOnStart
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

forceCoefficient

Coefficient by which the force will be multiplied when the object starts pulling on the anchor.

Declaration

```
[Tooltip("Coefficient by which the force will be multiplied when the object starts pulling o
n the anchor.")]
public float forceCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

localAnchorPoint

Point in coordinates local to the object this script is attached to.

Declaration

```
[Tooltip("Point in coordinates local to the object this script is attached to.")]
public Vector3 localAnchorPoint
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

zeroForceRadius

Radius around anchor in which the chain/rope is slack and in which no force will be applied.

Declaration

```
[Tooltip("Radius around anchor in which the chain/rope is slack and in which no force will b
e applied.")]
public float zeroForceRadius
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Properties

AnchorPoint

World space position of the anchor attachment point on the ship.

Declaration

```
public Vector3 AnchorPoint { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

AnchorPosition

Position of the anchor.

Declaration

```
public Vector3 AnchorPosition { get; set; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Dropped

Has the anchor been dropped?

Declaration

```
public bool Dropped { get; }
```

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

IsDragging

Is the anchor dragging on the floor?

Declaration

```
public bool IsDragging { get; }
```

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

ParentRigidbody

Rigidbody to which the force will be applied.

Declaration

```
public Rigidbody ParentRigidbody { get; }
```

Property Value

Type	Description
Rigidbody (https://docs.unity3d.com/ScriptReference/Rigidbody.html)	

Methods

Drop()

Drops the anchor. Opposite of Weigh()

Declaration

```
public void Drop()
```

Weigh()

Weighs (retracts) the anchor.

Declaration

```
public void Weigh()
```

Class Engine

Represents a ship engine and propeller system that generates thrust. Simulates RPM-based thrust generation with start/stop sequences and sound effects. Includes propeller efficiency modeling through thrust curves and support for reverse thrust. Multiple engines can be used per ship for differential thrust control.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Engine

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(AdvancedShipController))]
[Serializable]
public class Engine
```

Fields

applyThrustWhenAboveWater

Should thrust be applied when above water?

Declaration

```
[Tooltip("Should thrust be applied when above water?")]
public bool applyThrustWhenAboveWater
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

isOn

Is the engine currently on?

Declaration

```
[FormerlySerializedAs("_isOn")]
[Tooltip("Is the engine currently on?")]
public bool isOn
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

maxRPM

Max RPM of the engine.

Declaration

```
[FormerlySerializedAs("_maxRPM")]
[Tooltip("Max RPM of the engine.")]
public float maxRPM
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxSpeed

Ship speed in m/s at which the propeller thrust drops to zero due to water flow matching propeller speed. Used with thrustCurve to determine thrust efficiency at different ship speeds.

Declaration

```
[Tooltip("Ship speed at which propeller will reach its maximum rotational speed.")]
public float maxSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxThrust

Thrust at max RPM.

Declaration

```
[FormerlySerializedAs("_maxThrust")]
[Tooltip("Thrust at max RPM.")]
public float maxThrust
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

minRPM

Min RPM of the engine.

Declaration

```
[FormerlySerializedAs("_minRPM")]
[Tooltip("Min RPM of the engine.")]
public float minRPM
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

name

Display name for this engine.

Declaration

```
public string name
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

pitch

Idle pitch of the engine

Declaration

```
[Range(0, 2)]
[Tooltip("Idle pitch of the engine")]
public float pitch
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

pitchRange

Pitch range of the engine.

Declaration

```
[Tooltip("Pitch range of the engine.")]
[Range(0, 2)]
public float pitchRange
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

propellerRpmRatio

Engine RPM will be multiplied by this value to get rotation speed of the propeller. Animation only.

Declaration

```
[Tooltip("Engine RPM will be multiplied by this value to get rotation speed of the propelle
r. Animation only.")]
public float propellerRpmRatio
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

propellerTransform

Optional. Propeller transform. Visual rotation only, does not affect physics.

Declaration

```
[Tooltip("Optional. Propeller transform. Visual rotation only, does not affect physics.")]
public Transform propellerTransform
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

reverseThrustCoefficient

Amount of thrust that will be applied if ship is reversing

Declaration

```
[Tooltip("Amount of thrust that will be applied if ship is reversing")]
public float reverseThrustCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

rotationDirection

Direction of propeller rotation. Affects animation only.

Declaration

```
[Tooltip("Direction of propeller rotation. Affects animation only.")]
public Engine.RotationDirection rotationDirection
```

Field Value

Type	Description
Engine (NWH.DWP2.ShipController.Engine.html). RotationDirection (NWH.DWP2.ShipController.Engine.RotationDirection.html)	

rudderTransform

Optional. Only use if you vessel has propeller mounted to the rudder (as in outboard engines). Propuslion force direction will be rotated with rudder if assigned.

Declaration

```
[Tooltip("Optional. Only use if you vessel has propeller mounted to the rudder (as in outboa
rd engines). Propuslion force direction will be rotated with rudder if assigned.")]
public Transform rudderTransform
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

runningSource

Engine running audio source.

Declaration

```
[Tooltip("Engine running audio source.")]
public AudioSource runningSource
```

Field Value

Type	Description
AudioSource (https://docs.unity3d.com/ScriptReference/AudioSource.html)	

spinUpTime

Time needed to spin up the engines up to max RPM

Declaration

```
[FormerlySerializedAs("_spinUpTime")]
[Tooltip("Time needed to spin up the engines up to max RPM")]
public float spinUpTime
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

startDuration

How long the engine starting phase take?

Declaration

```
[Tooltip("How long the engine starting phase take?")]
public float startDuration
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

startOnThrottle

Should the engine start when the throttle is applied?

Declaration

```
[Tooltip("Should the engine start when the throttle is applied?")]
public bool startOnThrottle
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

startingRPM

Engine RPM when turning over. Used to determine starting sound pitch.

Declaration

```
[FormerlySerializedAs("startingRpm")]
[Tooltip("Engine RPM when turning over. Used to determine starting sound pitch.")]
public float startingRPM
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

startingSource

Engine starting source.

Declaration

```
[Tooltip("Engine starting source.")]
public AudioSource startingSource
```

Field Value

Type	Description
AudioSource (https://docs.unity3d.com/ScriptReference/AudioSource.html)	

stopDuration

How long will the engine stopping phase take?

Declaration

```
[Tooltip("How long will the engine stopping phase take?")]
public float stopDuration
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

stoppingSource

Engine stopping source.

Declaration

```
[Tooltip("Engine stopping source.")]
public AudioSource stoppingSource
```

Field Value

Type	Description
AudioSource (https://docs.unity3d.com/ScriptReference/AudioSource.html)	

throttleBinding

Which throttle input this engine responds to.

Declaration

```
public Engine.ThrottleBinding throttleBinding
```

Field Value

Type	Description
Engine (NWH.DWP2.ShipController.Engine.html). ThrottleBinding (NWH.DWP2.ShipController.Engine.ThrottleBinding.html)	

thrustCurve

Thrust curve of the propeller. X axis is speed in m/s and y axis is efficiency.

Declaration

```
[Tooltip("Thrust curve of the propeller. X axis is speed in m/s and y axis is efficiency.")]
public AnimationCurve thrustCurve
```

Field Value

Type	Description
AnimationCurve (https://docs.unity3d.com/ScriptReference/AnimationCurve.html)	

thrustDirection

Local direction in which the force will be applied. Does not affect the rotation of the propeller, which always happens around the local Z-axis of the propeller transform.

Declaration

```
[Tooltip("Local direction in which the force will be applied. Does not affect the rotation o
f the propeller, which always happens around the local Z-axis of the propeller transform.")]
public Vector3 thrustDirection
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

thrustPosition

Local position where the thust is applied, relative to ship.

Declaration

```
[Tooltip("Local position where the thust is applied, relative to ship.")]
public Vector3 thrustPosition
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

volume

Base volume of the engine

Declaration

```
[Range(0, 2)]
[Tooltip("Base volume of the engine")]
public float volume
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

volumeRange

Volume range of the engine.

Declaration

```
[Tooltip("Volume range of the engine.")]
[Range(0, 2)]
public float volumeRange
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Properties

EngineState

Current state of the engine (On, Off, Starting, or Stopping).

Declaration

```
public Engine.State EngineState { get; set; }
```

Property Value

Type	Description
Engine (NWH.DWP2.ShipController.Engine.html). State (NWH.DWP2.ShipController.Engine.State.html)	

RPM

Current RPM of the engine

Declaration

```
public float RPM { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

RpmPercent

Current RPM as a normalized value between 0 (minRPM) and 1 (maxRPM). Used for sound pitch and volume calculations.

Declaration

```
public float RpmPercent { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Submerged

True if the engine's thrust position is currently underwater. When false, the engine will not generate thrust unless applyThrustWhenAboveWater is enabled.

Declaration

```
public bool Submerged { get; }
```

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Thrust

Current thrust being generated by the engine and propeller in Newtons.

Declaration

```
public float Thrust { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

ThrustDirection

Direction of thrust force in world coordinates.

Declaration

```
public Vector3 ThrustDirection { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

ThrustPosition

Point at which thrust will be applied to the Rigidbody, in world coordinates.

Declaration

```
public Vector3 ThrustPosition { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Methods

Initialize(AdvancedShipController)

Initializes the engine with a reference to its parent ship controller.

Declaration

```
public void Initialize(AdvancedShipController sc)
```

Parameters

Type	Name	Description
AdvancedShipController (NWH.DWP2.ShipController.AdvancedShipController.html)	*sc*	The ship controller this engine belongs to.

SetDefaults()

Sets all engine parameters to default values suitable for most ships.

Declaration

```
public void SetDefaults()
```

SoundInit()

Initializes engine sound sources.

Declaration

```
public virtual void SoundInit()
```

SoundUpdate()

Updates engine sound pitch and volume based on RPM and state.

Declaration

```
public virtual void SoundUpdate()
```

StartEngine()

Starts the ship engine.

Declaration

```
public void StartEngine()
```

StopEngine()

Stops the ship engine.

Declaration

```
public void StopEngine()
```

Update()

Declaration

```
public virtual void Update()
```

Enum Engine.RotationDirection

Visual rotation direction of the propeller.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum Engine.RotationDirection
```

Fields

Name	Description
Left	
Right	

Enum Engine.State

Current operational state of the engine.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum Engine.State
```

Fields

Name	Description
Off	
On	
Starting	
Stopping	

Enum Engine.ThrottleBinding

Which throttle input channel this engine responds to. Allows independent control of multiple engines on the same ship.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum Engine.ThrottleBinding
```

Fields

Name	Description
Throttle	
Throttle2	
Throttle3	
Throttle4	

Class InputManagerShipInputProvider

Ship input provider for Unity's legacy Input Manager. Handles keyboard and gamepad input through Input.GetAxis and Input.GetButton. Requires proper axis and button setup in Project Settings > Input Manager. Falls back to hardcoded keys if input bindings are not configured.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ ShipInputProvider (NWH.DWP2.ShipController.ShipInputProvider.html)
              ↳ InputManagerShipInputProvider

Inherited Members

ShipInputProvider.Steering()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Steering)
ShipInputProvider.Throttle()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle)
ShipInputProvider.Throttle2()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle2)
ShipInputProvider.Throttle3()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle3)
ShipInputProvider.Throttle4()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle4)
ShipInputProvider.SternThruster()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_SternThruster)
ShipInputProvider.BowThruster()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_BowThruster)
ShipInputProvider.SubmarineDepth()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_SubmarineDepth)
ShipInputProvider.EngineStartStop()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_EngineStartStop)
ShipInputProvider.Anchor()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Anchor)
ShipInputProvider.RotateSail()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_RotateSail)
ShipInputProvider.DragObjectPosition()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_DragObjectPosition)
ShipInputProvider.DragObjectModifier()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_DragObjectModifier)
InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.Awake() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Awake)
InputProvider.OnDestroy()
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)

InputProvider.CombinedInput<T>(Func<T, int>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[DisallowMultipleComponent]
public class InputManagerShipInputProvider : ShipInputProvider
```

Class InputSystemShipInputProvider

Ship input provider for Unity's new Input System. Uses ShipInputActions asset for input binding configuration. Supports keyboard, gamepad, and custom input devices through the Input System. Input bindings can be modified by editing the ShipInputActions asset.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ ShipInputProvider (NWH.DWP2.ShipController.ShipInputProvider.html)
              ↳ InputSystemShipInputProvider

Inherited Members

InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.OnDestroy()
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)
InputProvider.CombinedInput<T>(Func<T, int>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class InputSystemShipInputProvider : ShipInputProvider
```

Fields

shipInputActions

Input action asset containing all ship control bindings.

Declaration

```
public ShipInputActions shipInputActions
```

Field Value

Type	Description
ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html)	

Methods

Anchor()

Toggle anchor drop/weigh state.

Declaration

```
public override bool Anchor()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if anchor toggle was triggered this frame.

Overrides

ShipInputProvider.Anchor()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Anchor)

Awake()

Declaration

```
public void Awake()
```

BowThruster()

Bow thruster input from -1 to 1 for lateral movement at the front.

Declaration

```
public override float BowThruster()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Bow thruster input value.

Overrides

ShipInputProvider.BowThruster() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_BowThruster)

DragObjectModifier()

Modifier key for object dragging.

Declaration

```
public override bool DragObjectModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if drag modifier is held.

Overrides

ShipInputProvider.DragObjectModifier() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_DragObjectModifier)

DragObjectPosition()

Mouse or touch position for object dragging in demo scenes.

Declaration

```
public override Vector2 DragObjectPosition()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	Drag position delta.

Overrides

ShipInputProvider.DragObjectPosition() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_DragObjectPosition)

EngineStartStop()

Toggle engine start/stop state.

Declaration

```
public override bool EngineStartStop()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if engine start/stop was triggered this frame.

Overrides

ShipInputProvider.EngineStartStop() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_EngineStartStop)

RotateSail()

Sail rotation input for sailing vessels.

Declaration

```
public override float RotateSail()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Sail rotation input value.

Overrides

ShipInputProvider.RotateSail() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_RotateSail)

Steering()

Horizontal steering input from -1 (port/left) to 1 (starboard/right).

Declaration

```
public override float Steering()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Steering input value.

Overrides

ShipInputProvider.Steering() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Steering)

SternThruster()

Stern thruster input from -1 to 1 for lateral movement at the rear.

Declaration

```
public override float SternThruster()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Stern thruster input value.

Overrides

ShipInputProvider.SternThruster() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_SternThruster)

SubmarineDepth()

Submarine depth control from -1 (surface) to 1 (dive).

Declaration

```
public override float SubmarineDepth()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Submarine depth input value.

Overrides

ShipInputProvider.SubmarineDepth() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_SubmarineDepth)

Throttle()

Primary throttle input from -1 (full reverse) to 1 (full forward).

Declaration

```
public override float Throttle()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle input value.

Overrides

ShipInputProvider.Throttle()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle)

Throttle2()

Secondary throttle input for independent engine control.

Declaration

```
public override float Throttle2()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle2 input value.

Overrides

ShipInputProvider.Throttle2()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle2)

Throttle3()

Tertiary throttle input for independent engine control.

Declaration

```
public override float Throttle3()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle3 input value.

Overrides

ShipInputProvider.Throttle3()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle3)

Throttle4()

Quaternary throttle input for independent engine control.

Declaration

```
public override float Throttle4()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle4 input value.

Overrides

ShipInputProvider.Throttle4()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle4)

Update()

Declaration

```
public void Update()
```

Class MobileShipInputProvider

Ship input provider for mobile devices using on-screen UI controls. Reads input from Unity UI Sliders and MobileInputButtons placed in the scene. All control assignments are optional - assign only the controls you need for your ship.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ ShipInputProvider (NWH.DWP2.ShipController.ShipInputProvider.html)
              ↳ MobileShipInputProvider

Inherited Members

ShipInputProvider.RotateSail()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_RotateSail)
ShipInputProvider.DragObjectPosition()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_DragObjectPosition)
ShipInputProvider.DragObjectModifier()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_DragObjectModifier)
InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.Awake() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Awake)
InputProvider.OnDestroy()
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)
InputProvider.CombinedInput<T>(Func<T, int>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>)
(NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class MobileShipInputProvider : ShipInputProvider
```

Fields

anchorButton

Button for toggling anchor.

Declaration

```
public MobileInputButton anchorButton
```

Field Value

Type	Description
MobileInputButton (NWH.Common.Input.MobileInputButton.html)	

bowThrusterSlider

Slider for bow thruster control.

Declaration

```
public Slider bowThrusterSlider
```

Field Value

Type	Description
Slider	

changeCameraButton

Button for changing camera in demo scenes.

Declaration

```
public MobileInputButton changeCameraButton
```

Field Value

Type	Description
MobileInputButton (NWH.Common.Input.MobileInputButton.html)	

changeShipButton

Button for changing ship in demo scenes.

Declaration

```
public MobileInputButton changeShipButton
```

Field Value

Type	Description
MobileInputButton (NWH.Common.Input.MobileInputButton.html)	

engineStartStopButton

Button for starting/stopping engines.

Declaration

```
public MobileInputButton engineStartStopButton
```

Field Value

Type	Description
MobileInputButton (NWH.Common.Input.MobileInputButton.html)	

steeringSlider

Slider for steering control.

Declaration

```
public Slider steeringSlider
```

Field Value

Type	Description
Slider	

sternThrusterSlider

Slider for stern thruster control.

Declaration

```
public Slider sternThrusterSlider
```

Field Value

Type	Description
Slider	

submarineDepthSlider

Slider for submarine depth control.

Declaration

```
public Slider submarineDepthSlider
```

Field Value

Type	Description
Slider	

throttleSlider

Slider for primary throttle control.

Declaration

```
public Slider throttleSlider
```

Field Value

Type	Description
Slider	

throttleSlider2

Slider for secondary throttle control.

Declaration

```
public Slider throttleSlider2
```

Field Value

Type	Description
Slider	

throttleSlider3

Slider for tertiary throttle control.

Declaration

```
public Slider throttleSlider3
```

Field Value

Type	Description
Slider	

throttleSlider4

Slider for quaternary throttle control.

Declaration

```
public Slider throttleSlider4
```

Field Value

Type	Description
Slider	

Methods

Anchor()

Toggle anchor drop/weigh state.

Declaration

```
public override bool Anchor()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if anchor toggle was triggered this frame.

Overrides

ShipInputProvider.Anchor()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Anchor)

BowThruster()

Bow thruster input from -1 to 1 for lateral movement at the front.

Declaration

```
public override float BowThruster()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Bow thruster input value.

Overrides

ShipInputProvider.BowThruster() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_BowThruster)

EngineStartStop()

Toggle engine start/stop state.

Declaration

```
public override bool EngineStartStop()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if engine start/stop was triggered this frame.

Overrides

ShipInputProvider.EngineStartStop() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_EngineStartStop)

Steering()

Horizontal steering input from -1 (port/left) to 1 (starboard/right).

Declaration

```
public override float Steering()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Steering input value.

Overrides

ShipInputProvider.Steering() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Steering)

SternThruster()

Stern thruster input from -1 to 1 for lateral movement at the rear.

Declaration

```
public override float SternThruster()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Stern thruster input value.

Overrides

ShipInputProvider.SternThruster() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_SternThruster)

SubmarineDepth()

Submarine depth control from -1 (surface) to 1 (dive).

Declaration

```
public override float SubmarineDepth()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Submarine depth input value.

Overrides

ShipInputProvider.SubmarineDepth() (NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_SubmarineDepth)

Throttle()

Primary throttle input from -1 (full reverse) to 1 (full forward).

Declaration

```
public override float Throttle()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle input value.

Overrides

ShipInputProvider.Throttle()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle)

Throttle2()

Secondary throttle input for independent engine control.

Declaration

```
public override float Throttle2()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle2 input value.

Overrides

ShipInputProvider.Throttle2()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle2)

Throttle3()

Tertiary throttle input for independent engine control.

Declaration

```
public override float Throttle3()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle3 input value.

Overrides

ShipInputProvider.Throttle3()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle3)

Throttle4()

Quaternary throttle input for independent engine control.

Declaration

```
public override float Throttle4()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle4 input value.

Overrides

ShipInputProvider.Throttle4()
(NWH.DWP2.ShipController.ShipInputProvider.html#NWH_DWP2_ShipController_ShipInputProvider_Throttle4)

Class Rudder

Represents a ship rudder that rotates based on steering input. Provides visual rotation only - actual steering forces are applied by WaterObject components on the rudder itself. Multiple rudders can be used on a single ship.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Rudder

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[Serializable]
public class Rudder
```

Fields

localRotationAxis

Axis around which the rudder will be rotated.

Declaration

```
[Tooltip("Axis around which the rudder will be rotated.")]
public Vector3 localRotationAxis
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

maxAngle

Max angle in degrees rudder will be able to reach.

Declaration

```
[Tooltip("Max angle in degrees rudder will be able to reach.")]
public float maxAngle
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

name

Name of the rudder. Can be any string.

Declaration

```
[Tooltip("Name of the rudder. Can be any string.")]
public string name
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

rotationSpeed

Rotation speed in degrees per second.

Declaration

```
[Tooltip("Rotation speed in degrees per second.")]
public float rotationSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

rudderTransform

Transform representing the rudder.

Declaration

```
[Tooltip("Transform representing the rudder.")]
public Transform rudderTransform
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

Properties

Angle

Current angle of the rudder in degrees. Positive angles indicate starboard turn, negative angles indicate port turn.

Declaration

```
public float Angle { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

AnglePercent

Current rudder angle as a normalized value between -1 and 1.

Declaration

```
public float AnglePercent { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

Initialize(AdvancedShipController)

Initializes the rudder with a reference to its parent ship controller.

Declaration

```
public void Initialize(AdvancedShipController sc)
```

Parameters

Type	Name	Description
AdvancedShipController (NWH.DWP2.ShipController.AdvancedShipController.html)	*sc*	The ship controller this rudder belongs to.

Update()

Declaration

```
public virtual void Update()
```

Class ShipInputActions

Provides programmatic access to UnityEngine.InputSystem.InputActionAsset, UnityEngine.InputSystem.InputActionMap, UnityEngine.InputSystem.InputAction and UnityEngine.InputSystem.InputControlScheme instances defined in asset "Packages/com.nwh.dynamicwaterphysics/Runtime/ShipController/Input/InputProviders/InputSystemProvider/ShipInputActions.inputactions".

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ShipInputActions

Implements

IInputActionCollection2
IInputActionCollection
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<InputAction>
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.ienumerable)
IDisposable (https://learn.microsoft.com/dotnet/api/system.idisposable)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class ShipInputActions : IInputActionCollection2, IInputActionCollection, IEnumerable<InputAction>, IEnume
rable, IDisposable
```

Remarks

This class is source generated and any manual edits will be discarded if the associated asset is reimported or modified.

Examples

```
using namespace UnityEngine;
using UnityEngine.InputSystem;

// Example of using an InputActionMap named "Player" from a UnityEngine.MonoBehaviour implementing callback inter
face.
public class Example : MonoBehaviour, MyActions.IPlayerActions
{
    private MyActions_Actions m_Actions;                  // Source code representation of asset.
    private MyActions_Actions.PlayerActions m_Player;     // Source code representation of action map.

    void Awake()
    {
        m_Actions = new MyActions_Actions();              // Create asset object.
        m_Player = m_Actions.Player;                      // Extract action map object.
        m_Player.AddCallbacks(this);                      // Register callback interface IPlayerActions.
    }

    void OnDestroy()
    {
        m_Actions.Dispose();                              // Destroy asset object.
    }

    void OnEnable()
    {
        m_Player.Enable();                                // Enable all actions within map.
    }

    void OnDisable()
    {
        m_Player.Disable();                               // Disable all actions within map.
    }

    #region Interface implementation of MyActions.IPlayerActions

    // Invoked when "Move" action is either started, performed or canceled.
    public void OnMove(InputAction.CallbackContext context)
    {
        Debug.Log($"OnMove: {context.ReadValue<Vector2>()}");
    }

    // Invoked when "Attack" action is either started, performed or canceled.
    public void OnAttack(InputAction.CallbackContext context)
    {
        Debug.Log($"OnAttack: {context.ReadValue<float>()}");
    }

    #endregion
}
```

Constructors

ShipInputActions()

Constructs a new instance.

Declaration

```
public ShipInputActions()
```

Properties

ShipControls

Provides a new ShipInputActions.ShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html) instance referencing this action map.

Declaration

```
public ShipInputActions.ShipControlsActions ShipControls { get; }
```

Property Value

Type	Description
ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html). ShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html)	

asset

Provides access to the underlying asset instance.

Declaration

```
public InputActionAsset asset { get; }
```

Property Value

Type	Description
InputActionAsset	

bindingMask

Declaration

```
public InputBinding? bindingMask { get; set; }
```

Property Value

Type	Description
InputBinding?	

bindings

Declaration

```
public IEnumerable<InputBinding> bindings { get; }
```

Property Value

Type	Description
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)<InputBinding>	

controlSchemes

Declaration

```
public ReadOnlyArray<InputControlScheme> controlSchemes { get; }
```

Property Value

Type	Description
ReadOnlyArray<InputControlScheme>	

devices

Declaration

```
public ReadOnlyArray<InputDevice>? devices { get; set; }
```

Property Value

Type	Description
ReadOnlyArray<InputDevice>?	

Methods

Contains(InputAction)

Declaration

```
public bool Contains(InputAction action)
```

Parameters

Type	Name	Description
InputAction	*action*	

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Disable()

Declaration

```
public void Disable()
```

Dispose()

Destroys this asset and all associated UnityEngine.InputSystem.InputAction instances.

Declaration

```
public void Dispose()
```

Enable()

Declaration

```
public void Enable()
```

~ShipInputActions()

Declaration

```
protected ~ShipInputActions()
```

FindAction(string, bool)

Declaration

```
public InputAction FindAction(string actionNameOrId, bool throwIfNotFound = false)
```

Parameters

Type	Name	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	*actionNameOrId*	
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*throwIfNotFound*	

Returns

Type	Description
InputAction	

FindBinding(InputBinding, out InputAction)

Declaration

```
public int FindBinding(InputBinding bindingMask, out InputAction action)
```

Parameters

Type	Name	Description
InputBinding	*bindingMask*	
InputAction	*action*	

Returns

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

GetEnumerator()

Returns an enumerator that iterates through the collection.

Declaration

```
public IEnumerator<InputAction> GetEnumerator()
```

Returns

Type	Description
IEnumerator (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerator-1) <InputAction>	An enumerator that can be used to iterate through the collection.

Implements

UnityEngine.InputSystem.IInputActionCollection2
UnityEngine.InputSystem.IInputActionCollection
IEnumerable<T> (https://learn.microsoft.com/dotnet/api/system.collections.generic.ienumerable-1)
IEnumerable (https://learn.microsoft.com/dotnet/api/system.collections.ienumerable)
IDisposable (https://learn.microsoft.com/dotnet/api/system.idisposable)

Interface ShipInputActions.IShipControls Actions

Interface to implement callback methods for all input action callbacks associated with input actions defined by "ShipControls" which allows adding and removing callbacks.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public interface ShipInputActions.IShipControlsActions
```

Methods

OnAnchor(CallbackContext)

Method invoked when associated input action "Anchor" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnAnchor(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnBowThruster(CallbackContext)

Method invoked when associated input action "BowThruster" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnBowThruster(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnEngineStartStop(CallbackContext)

Method invoked when associated input action "EngineStartStop" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnEngineStartStop(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnRotateSail(CallbackContext)

Method invoked when associated input action "RotateSail" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnRotateSail(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnSteering(CallbackContext)

Method invoked when associated input action "Steering" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnSteering(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnSternThruster(CallbackContext)

Method invoked when associated input action "SternThruster" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnSternThruster(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnSubmarineDepth(CallbackContext)

Method invoked when associated input action "SubmarineDepth" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnSubmarineDepth(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnThrottle(CallbackContext)

Method invoked when associated input action "Throttle" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnThrottle(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnThrottle2(CallbackContext)

Method invoked when associated input action "Throttle2" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnThrottle2(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnThrottle3(CallbackContext)

Method invoked when associated input action "Throttle3" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnThrottle3(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

OnThrottle4(CallbackContext)

Method invoked when associated input action "Throttle4" is either UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed or UnityEngine.InputSystem.InputAction.canceled.

Declaration

```
void OnThrottle4(InputAction.CallbackContext context)
```

Parameters

Type	Name	Description
InputAction.CallbackContext	*context*	

See Also

started
performed
canceled

See Also

AddCallbacks (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html#NWH_DWP2_ShipController_ShipInputActions_ShipControlsActions_AddCallbacks_NWH_DWP2_ShipController_ShipInputActions_IShipControlsActions_)(IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html))
RemoveCallbacks (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html#NWH_DWP2_ShipController_ShipInputActions_ShipControlsActions_RemoveCallbacks_NWH_DWP2_ShipController_ShipInputActions_IShipControlsActions_)(IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html))

Struct ShipInputActions.ShipControlsActions

Provides access to input actions defined in input action map "ShipControls".

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public struct ShipInputActions.ShipControlsActions
```

Constructors

ShipControlsActions(ShipInputActions)

Construct a new instance of the input action map wrapper class.

Declaration

```
public ShipControlsActions(ShipInputActions wrapper)
```

Parameters

Type	Name	Description
ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html)	*wrapper*	

Properties

Anchor

Provides access to the underlying input action "ShipControls/Anchor".

Declaration

```
public InputAction Anchor { get; }
```

Property Value

Type	Description
InputAction	

BowThruster

Provides access to the underlying input action "ShipControls/BowThruster".

Declaration

```
public InputAction BowThruster { get; }
```

Property Value

Type	Description
InputAction	

EngineStartStop

Provides access to the underlying input action "ShipControls/EngineStartStop".

Declaration

```
public InputAction EngineStartStop { get; }
```

Property Value

Type	Description
InputAction	

RotateSail

Provides access to the underlying input action "ShipControls/RotateSail".

Declaration

```
public InputAction RotateSail { get; }
```

Property Value

Type	Description
InputAction	

Steering

Provides access to the underlying input action "ShipControls/Steering".

Declaration

```
public InputAction Steering { get; }
```

Property Value

Type	Description
InputAction	

SternThruster

Provides access to the underlying input action "ShipControls/SternThruster".

Declaration

```
public InputAction SternThruster { get; }
```

Property Value

Type	Description
InputAction	

SubmarineDepth

Provides access to the underlying input action "ShipControls/SubmarineDepth".

Declaration

```
public InputAction SubmarineDepth { get; }
```

Property Value

Type	Description
InputAction	

Throttle

Provides access to the underlying input action "ShipControls/Throttle".

Declaration

```
public InputAction Throttle { get; }
```

Property Value

Type	Description
InputAction	

Throttle2

Provides access to the underlying input action "ShipControls/Throttle2".

Declaration

```
public InputAction Throttle2 { get; }
```

Property Value

Type	Description
InputAction	

Throttle3

Provides access to the underlying input action "ShipControls/Throttle3".

Declaration

```
public InputAction Throttle3 { get; }
```

Property Value

Type	Description
InputAction	

Throttle4

Provides access to the underlying input action "ShipControls/Throttle4".

Declaration

```
public InputAction Throttle4 { get; }
```

Property Value

Type	Description
InputAction	

enabled

Declaration

```
public bool enabled { get; }
```

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Methods

AddCallbacks(IShipControlsActions)

Adds UnityEngine.InputSystem.InputAction.started, UnityEngine.InputSystem.InputAction.performed and Unity Engine.InputSystem.InputAction.canceled callbacks provided via on all input actions contained in this map.

Declaration

```
public void AddCallbacks(ShipInputActions.IShipControlsActions instance)
```

Parameters

Type	Name	Description
ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html). IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html)	*instance*	Callback instance.

Remarks

If `instance` is `null` or `instance` have already been added this method does nothing.

See Also

ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html).ShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html)

Disable()

Declaration

```
public void Disable()
```

Enable()

Declaration

```
public void Enable()
```

Get()

Provides access to the underlying input action map instance.

Declaration

```
public InputActionMap Get()
```

Returns

Type	Description
InputActionMap	

RemoveCallbacks(IShipControlsActions)

Unregisters and unregisters all input action callbacks via UnregisterCallbacks(IShipControlsActions) (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html).

Declaration

```
public void RemoveCallbacks(ShipInputActions.IShipControlsActions instance)
```

Parameters

Type	Name	Description
ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html). IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html)	*instance*	

See Also

UnregisterCallbacks (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html)(IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html))

SetCallbacks(IShipControlsActions)

Replaces all existing callback instances and previously registered input action callbacks associated with them with callbacks provided via .

Declaration

```
public void SetCallbacks(ShipInputActions.IShipControlsActions instance)
```

Parameters

Type	Name	Description
ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html). IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html)	*instance*	

Remarks

If `instance` is `null`, calling this method will only unregister all existing callbacks but not register any new callbacks.

See Also

AddCallbacks(IShipControlsActions) (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html#NWH_DWP2_ShipController_ShipInputActions_ShipControlsActions_AddCallbacks_NWH_DWP2_ShipController_ShipInputActions_IShipControlsActions_) RemoveCallbacks(IShipControlsActions) (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html#NWH_DWP2_ShipController_ShipInputActions_ShipControlsActions_RemoveCallbacks_NWH_DWP2_ShipController_ShipInputActions_IShipControlsActions_) UnregisterCallbacks (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html)(IShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.IShipControlsActions.html))

Operators

implicit operator InputActionMap(ShipControlsActions)

Implicitly converts an to an instance.

Declaration

```
public static implicit operator InputActionMap(ShipInputActions.ShipControlsActions set)
```

Parameters

Type	Name	Description
ShipInputActions (NWH.DWP2.ShipController.ShipInputActions.html). ShipControlsActions (NWH.DWP2.ShipController.ShipInputActions.ShipControlsActions.html)	*set*	

Returns

Type	Description
InputActionMap	

Class ShipInputHandler

Manages ship input by retrieving values from active ShipInputProviders and storing them in ShipInputStates. Automatically polls all registered input providers and combines their inputs. Can be disabled for manual input control via scripting or AI.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ ShipInputHandler

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[Serializable]
public class ShipInputHandler
```

Fields

autoSetInput

When enabled input will be auto-retrieved from the InputProviders present in the scene. Disable to manualy set the input through external scripts, i.e. AI controller.

Declaration

```
[Tooltip("When enabled input will be auto-retrieved from the InputProviders present in the s
cene.\r\nDisable to manualy set the input through external scripts, i.e. AI controller.")]
public bool autoSetInput
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

modifyInputCallback

Callback invoked after input is processed each frame. Use this to modify input values programmatically before they are used.

Declaration

```
public UnityEvent modifyInputCallback
```

Field Value

Type	Description
UnityEvent (https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html)	

states

All the input states of the vehicle. Can be used to set input through scripting or copy the inputs over from other vehicle, such as truck to trailer.

Declaration

```
[Tooltip("All the input states of the vehicle. Can be used to set input through scripting or
copy the inputs\r\nover from other vehicle, such as truck to trailer.")]
public ShipInputStates states
```

Field Value

Type	Description
ShipInputStates (NWH.DWP2.ShipController.ShipInputStates.html)	

Properties

Anchor

Anchor toggle state. True for one frame when anchor input is pressed.

Declaration

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

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

BowThruster

Bow thruster input from -1 to 1.

Declaration

```
public float BowThruster { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

EngineStartStop

Engine start/stop toggle state. True for one frame when toggle input is pressed.

Declaration

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

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

RotateSail

Sail rotation input for sailing vessels.

Declaration

```
public float RotateSail { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Steering

Steering input from -1 (port/left) to 1 (starboard/right).

Declaration

```
public float Steering { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

SternThruster

Stern thruster input from -1 to 1.

Declaration

```
public float SternThruster { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

SubmarineDepth

Submarine depth control from -1 (surface) to 1 (dive).

Declaration

```
public float SubmarineDepth { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Throttle

Primary throttle input from -1 (full reverse) to 1 (full forward).

Declaration

```
public float Throttle { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Throttle2

Secondary throttle for independent engine control.

Declaration

```
public float Throttle2 { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Throttle3

Tertiary throttle for independent engine control.

Declaration

```
public float Throttle3 { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Throttle4

Quaternary throttle for independent engine control.

Declaration

```
public float Throttle4 { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

Update()

Updates all input values from registered input providers. Called automatically by AdvancedShipController.

Declaration

```
public void Update()
```

Class ShipInputProvider

Base class for all ship input providers. Inherit from this to create custom input systems for ship controls. Provides virtual methods for all ship control inputs including throttle, steering, thrusters, and submarine depth control.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ InputProvider (NWH.Common.Input.InputProvider.html)
            ↳ ShipInputProvider
              ↳ InputManagerShipInputProvider (NWH.DWP2.ShipController.InputManagerShipInputProvider.html)
              ↳ InputSystemShipInputProvider (NWH.DWP2.ShipController.InputSystemShipInputProvider.html)
              ↳ MobileShipInputProvider (NWH.DWP2.ShipController.MobileShipInputProvider.html)

Inherited Members

InputProvider.Instances (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Instances)
InputProvider.Awake() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_Awake)
InputProvider.OnDestroy() (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_OnDestroy)
InputProvider.CombinedInput<T>(Func<T, int>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Int32__)
InputProvider.CombinedInput<T>(Func<T, float>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Single__)
InputProvider.CombinedInput<T>(Func<T, bool>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_System_Boolean__)
InputProvider.CombinedInput<T>(Func<T, Vector2>) (NWH.Common.Input.InputProvider.html#NWH_Common_Input_InputProvider_CombinedInput__1_System_Func__0_UnityEngine_Vector2__)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public abstract class ShipInputProvider : InputProvider
```

Methods

Anchor()

Toggle anchor drop/weigh state.

Declaration

```
public virtual bool Anchor()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if anchor toggle was triggered this frame.

BowThruster()

Bow thruster input from -1 to 1 for lateral movement at the front.

Declaration

```
public virtual float BowThruster()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Bow thruster input value.

DragObjectModifier()

Modifier key for object dragging.

Declaration

```
public virtual bool DragObjectModifier()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if drag modifier is held.

DragObjectPosition()

Mouse or touch position for object dragging in demo scenes.

Declaration

```
public virtual Vector2 DragObjectPosition()
```

Returns

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	Drag position delta.

EngineStartStop()

Toggle engine start/stop state.

Declaration

```
public virtual bool EngineStartStop()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if engine start/stop was triggered this frame.

RotateSail()

Sail rotation input for sailing vessels.

Declaration

```
public virtual float RotateSail()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Sail rotation input value.

Steering()

Horizontal steering input from -1 (port/left) to 1 (starboard/right).

Declaration

```
public virtual float Steering()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Steering input value.

SternThruster()

Stern thruster input from -1 to 1 for lateral movement at the rear.

Declaration

```
public virtual float SternThruster()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Stern thruster input value.

SubmarineDepth()

Submarine depth control from -1 (surface) to 1 (dive).

Declaration

```
public virtual float SubmarineDepth()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Submarine depth input value.

Throttle()

Primary throttle input from -1 (full reverse) to 1 (full forward).

Declaration

```
public virtual float Throttle()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle input value.

Throttle2()

Secondary throttle input for independent engine control.

Declaration

```
public virtual float Throttle2()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle2 input value.

Throttle3()

Tertiary throttle input for independent engine control.

Declaration

```
public virtual float Throttle3()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle3 input value.

Throttle4()

Quaternary throttle input for independent engine control.

Declaration

```
public virtual float Throttle4()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Throttle4 input value.

Struct ShipInputStates

Container for all ship input states. Stores current values for all ship controls including throttle, steering, thrusters, and special inputs.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[Serializable]
public struct ShipInputStates
```

Fields

anchor

Anchor drop/weigh toggle state.

Declaration

```
public bool anchor
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

bowThruster

Bow thruster input from -1 to 1.

Declaration

```
[Range(-1, 1)]
public float bowThruster
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

changeCamera

Change camera input for demo scenes.

Declaration

```
public bool changeCamera
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

changeShip

Change ship input for demo scenes.

Declaration

```
public bool changeShip
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

engineStartStop

Engine start/stop toggle state.

Declaration

```
public bool engineStartStop
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

rotateSail

Sail rotation input for sailing vessels.

Declaration

```
[Range(-1, 1)]
public float rotateSail
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

steering

Steering input from -1 (port/left) to 1 (starboard/right).

Declaration

```
[Range(-1, 1)]
public float steering
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

sternThruster

Stern thruster input from -1 to 1.

Declaration

```
[Range(-1, 1)]
public float sternThruster
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

submarineDepth

Submarine depth control from 0 (surface) to 1 (dive).

Declaration

```
[Range(0, 1)]
public float submarineDepth
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

throttle

Primary throttle from -1 (full reverse) to 1 (full forward).

Declaration

```
[Range(-1, 1)]
public float throttle
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

throttle2

Secondary throttle for independent engine control.

Declaration

```
[Range(-1, 1)]
public float throttle2
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

throttle3

Tertiary throttle for independent engine control.

Declaration

```
[Range(-1, 1)]
public float throttle3
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

throttle4

Quaternary throttle for independent engine control.

Declaration

```
[Range(-1, 1)]
public float throttle4
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

Reset()

Resets all input states to default values.

Declaration

```
public void Reset()
```

Class Sink

Simulates a ship taking on water and sinking by gradually increasing mass over time. Works with VariableCenterOfMass component to affect buoyancy and cause the ship to sink. Can be triggered to simulate hull breaches or flooding.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ Sink

Implements

IMassAffector (NWH.Common.CoM.IMassAffector.html)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class Sink : MonoBehaviour, IMassAffector
```

Fields

addedMassPercentPerSecond

Rate at which mass is added per second as a percentage of maxAdditionalMass. Higher values cause faster sinking.

Declaration

```
[Tooltip("Percentage of initial mass that will be added each second to imitate water ingres
s")]
public float addedMassPercentPerSecond
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxAdditionalMass

Maximum mass in kg that can be added through flooding. This represents the total water mass that can enter the ship.

Declaration

```
[Tooltip("Maximum added mass after water ingress. 1f = 100% of orginal mass, 2f = 200% of or
iginal mass, etc.")]
public float maxAdditionalMass
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

GetMass()

Current mass of this affector in kilograms. Should return variable values for fuel tanks, cargo, etc.

Declaration

```
public float GetMass()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Mass in kg

GetTransform()

Returns transform of the mass affector.

Declaration

```
public Transform GetTransform()
```

Returns

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

GetWorldCenterOfMass()

World position of this affector's center of mass. Used for weighted center of mass calculations.

Declaration

```
public Vector3 GetWorldCenterOfMass()
```

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

ResetMass()

Resets the added mass to zero, returning the ship to its original mass.

Declaration

```
public void ResetMass()
```

SinkCoroutine()

Coroutine that gradually increases mass to simulate water ingress.

Declaration

```
public IEnumerator SinkCoroutine()
```

Returns

Type	Description
IEnumerator (https://learn.microsoft.com/dotnet/api/system.collections.ienumerator)	

StartSinking()

Begins the sinking process by starting mass increase over time.

Declaration

```
public void StartSinking()
```

StopSinking()

Stops the sinking process. Added mass remains until reset.

Declaration

```
public void StopSinking()
```

Implements

IMassAffector (NWH.Common.CoM.IMassAffector.html)

Class Submarine

Enables submarine functionality by controlling ballast mass for diving and surfacing. Manages depth control through variable mass and can automatically maintain horizontal orientation. Works with VariableCenterOfMass to adjust buoyancy for depth changes.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ Submarine

Implements

IMassAffector (NWH.Common.CoM.IMassAffector.html)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(AdvancedShipController))]
[RequireComponent(typeof(VariableCenterOfMass))]
[DefaultExecutionOrder(500)]
public class Submarine : MonoBehaviour, IMassAffector
```

Fields

ReferenceWaterObject

Reference to the WaterObject used for water level detection.

Declaration

```
public WaterObject ReferenceWaterObject
```

Field Value

Type	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	

ballastChangeSpeed

Speed of change of the ballast mass, as a percentage of maxBallastMass.

Declaration

```
[Tooltip("Speed of change of the ballast mass, as a percentage of maxBallastMass.")]
public float ballastChangeSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

keepHorizontal

If enabled submarine will try to keep horizontal by shifting the center of mass.

Declaration

```
[Tooltip("If enabled submarine will try to keep horizontal by shifting the center of mas
s.")]
public bool keepHorizontal
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

keepHorizontalSensitivity

Sensitivity of calculation trying to keep the submarine horizontal. Higher number will mean faster reaction.

Declaration

```
[Tooltip("Sensitivity of calculation trying to keep the submarine horizontal. Higher number
will mean faster reaction.")]
public float keepHorizontalSensitivity
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxBallastMass

Maximum ballast mass in kg that can be added to make the submarine sink. Higher values allow diving deeper and faster but require more time to surface.

Declaration

```
[FormerlySerializedAs("maxAdditionalMass")]
[FormerlySerializedAs("maxMassFactor")]
[Tooltip("Maximum additional mass that can be added (taking on water) to the base mass of th
e rigidbody to make submarine sink.")]
public float maxBallastMass
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxMassOffset

Maximum rigidbody center of mass offset that can be used to keep the submarine level.

Declaration

```
[Tooltip("Maximum rigidbody center of mass offset that can be used to keep the submarine lev
el.")]
public float maxMassOffset
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Properties

DepthInput

Input for depth control from -1 (surface) to 1 (dive). Positive values add ballast mass to sink, negative values reduce it to surface.

Declaration

```
public float DepthInput { get; set; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

GetMass()

Current mass of this affector in kilograms. Should return variable values for fuel tanks, cargo, etc.

Declaration

```
public float GetMass()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Mass in kg

GetTransform()

Returns transform of the mass affector.

Declaration

```
public Transform GetTransform()
```

Returns

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

GetWorldCenterOfMass()

World position of this affector's center of mass. Used for weighted center of mass calculations.

Declaration

```
public Vector3 GetWorldCenterOfMass()
```

Returns

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Implements

IMassAffector (NWH.Common.CoM.IMassAffector.html)

Class Thruster

Lateral thruster for sideways ship movement. Bow thrusters are mounted at the front for tight turning and docking, stern thrusters at the rear. Multiple thrusters of each type can be added for increased lateral control.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Thruster

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
[Serializable]
public class Thruster
```

Fields

maxThrust

Maximum thrust force in Newtons.

Declaration

```
[Tooltip("Max thrust in [N].")]
public float maxThrust
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

name

Display name for this thruster.

Declaration

```
[Tooltip("Name of the thruster - can be any string.")]
public string name
```

Field Value

Type	Description
string (https://learn.microsoft.com/dotnet/api/system.string)	

position

Position relative to ship transform where thrust force is applied.

Declaration

```
[Tooltip("Relative force application position.")]
public Vector3 position
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

propellerRotationDirection

Declaration

```
[FormerlySerializedAs("rotationDirection")]
[Tooltip("Rotation direction of the propeller. Visual only.")]
public Thruster.RotationDirection propellerRotationDirection
```

Field Value

Type	Description
Thruster (NWH.DWP2.ShipController.Thruster.html). RotationDirection (NWH.DWP2.ShipController.Thruster.RotationDirection.html)	

propellerRotationSpeed

Declaration

```
[Tooltip("Rotation speed of the propeller if assigned. Visual only.")]
public float propellerRotationSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

propellerTransform

Declaration

```
[Tooltip("Optional. Transform representing a propeller. Visual only.")]
public Transform propellerTransform
```

Field Value

Type	Description
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	

spinUpSpeed

Declaration

```
[Tooltip("Time needed to reach maxThrust.")]
public float spinUpSpeed
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

thrusterPosition

Whether this is a bow or stern thruster.

Declaration

```
public Thruster.ThrusterPosition thrusterPosition
```

Field Value

Type	Description
Thruster (NWH.DWP2.ShipController.Thruster.html). ThrusterPosition (NWH.DWP2.ShipController.Thruster.ThrusterPosition.html)	

Properties

Input

Current input value for this thruster from -1 to 1. Automatically retrieves from the appropriate input channel based on thrusterPosition.

Declaration

```
public float Input { get; }
```

Property Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

WorldPosition

World space position where thrust force is applied.

Declaration

```
public Vector3 WorldPosition { get; }
```

Property Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

Methods

Initialize(AdvancedShipController)

Initializes the thruster with a reference to its parent ship controller.

Declaration

```
public void Initialize(AdvancedShipController sc)
```

Parameters

Type	Name	Description
AdvancedShipController (NWH.DWP2.ShipController.AdvancedShipController.html)	*sc*	The ship controller this thruster belongs to.

Update()

Declaration

```
public virtual void Update()
```

Enum Thruster.RotationDirection

Visual rotation direction of the thruster propeller.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum Thruster.RotationDirection
```

Fields

Name	Description
Left	
Right	

Enum Thruster.ThrusterPosition

Position of the thruster on the ship. Determines which input controls it and the direction of thrust application.

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).ShipController (NWH.DWP2.ShipController.html)

Assembly: NWH.DWP2.dll

Syntax

```
public enum Thruster.ThrusterPosition
```

Fields

Name	Description
BowThruster	
SternThruster	

Namespace NWH.DWP2.WaterData

Classes

FlatWaterDataProvider (NWH.DWP2.WaterData.FlatWaterDataProvider.html)

Simple water data provider for flat, static water surfaces. Uses a constant water height based on the transform's Y position. Does not support water height queries, waves, normals, or flow.

RaycastWaterDataProvider (NWH.DWP2.WaterData.RaycastWaterDataProvider.html)

Water data provider that uses raycasting to detect water surface. Works with any water system that has a collider. Supports water height and normal queries through raycasting. Uses Unity's job system for parallel raycasts.

WaterDataProvider (NWH.DWP2.WaterData.WaterDataProvider.html)

Base class for providing water surface data to WaterObjects. Implementations provide water height, flow velocity, and surface normal information. Uses a trigger collider to automatically detect and register WaterObjects that enter the water. Inherit from this class to create adapters for different water systems.

Class FlatWaterDataProvider

Simple water data provider for flat, static water surfaces. Uses a constant water height based on the transform's Y position. Does not support water height queries, waves, normals, or flow.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WaterDataProvider (NWH.DWP2.WaterData.WaterDataProvider.html)
            ↳ FlatWaterDataProvider

Inherited Members

WaterDataProvider._singleHeightArray (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider__singleHeightArray)
WaterDataProvider._singlePointArray (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider__singlePointArray)
WaterDataProvider._triggerCollider (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider__triggerCollider)
WaterDataProvider.Awake() (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_Awake)
WaterDataProvider.GetWaterFlows(WaterObject, ref Vector3[], ref Vector3[]) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterFlows_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___UnityEngine_Vector3___)
WaterDataProvider.GetWaterNormals(WaterObject, ref Vector3[], ref Vector3[]) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterNormals_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___UnityEngine_Vector3___)
WaterDataProvider.GetWaterHeightsFlowsNormals(WaterObject, ref Vector3[], ref float[], ref Vector3[], ref Vector3[], bool, bool, bool) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeightsFlowsNormals_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___System_Single___UnityEngine_Vector3___UnityEngine_Vector3___System_Boolean_System_Boolean_System_Boolean_)
WaterDataProvider.GetWaterHeightSingle(WaterObject, Vector3) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeightSingle_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3_)
WaterDataProvider.PointInWater(WaterObject, Vector3) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_PointInWater_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3_)
WaterDataProvider.GetWaterHeight(WaterObject, Vector3) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeight_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3_)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterData (NWH.DWP2.WaterData.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class FlatWaterDataProvider : WaterDataProvider
```

Methods

GetWaterHeights(WaterObject, ref Vector3[], ref float[])

Returns water height at each given point. Override this method to provide water height data from your water system.

Declaration

```
public override void GetWaterHeights(WaterObject waterObject, ref Vector3[] points, ref floa
t[] waterHeights)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*points*	Position array in world coordinates.
float (https://learn.microsoft.com/dotnet/api/system.single) []	*waterHeights*	Water height array in world coordinates. Corresponds to positions.

Overrides

WaterDataProvider.GetWaterHeights(WaterObject, ref Vector3[], ref float[]) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeights_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___System_Single___)

SupportsWaterFlowQueries()

Does this water system support water velocity queries?

Declaration

```
public override bool SupportsWaterFlowQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

Overrides

WaterDataProvider.SupportsWaterFlowQueries() (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_SupportsWaterFlowQueries)

SupportsWaterHeightQueries()

Does this water system support water height queries?

Declaration

```
public override bool SupportsWaterHeightQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

Overrides

WaterDataProvider.SupportsWaterHeightQueries() (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_SupportsWaterHeightQueries)

SupportsWaterNormalQueries()

Does this water system support water normal queries?

Declaration

```
public override bool SupportsWaterNormalQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

Overrides

WaterDataProvider.SupportsWaterNormalQueries() (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_SupportsWaterNormalQueries)

Class RaycastWaterDataProvider

Water data provider that uses raycasting to detect water surface. Works with any water system that has a collider. Supports water height and normal queries through raycasting. Uses Unity's job system for parallel raycasts.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WaterDataProvider (NWH.DWP2.WaterData.WaterDataProvider.html)
            ↳ RaycastWaterDataProvider

Inherited Members

WaterDataProvider._singleHeightArray (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider__singleHeightArray)
WaterDataProvider._singlePointArray (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider__singlePointArray)
WaterDataProvider._triggerCollider (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider__triggerCollider)
WaterDataProvider.GetWaterFlows(WaterObject, ref Vector3[], ref Vector3[]) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterFlows_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___UnityEngine_Vector3___)
WaterDataProvider.GetWaterHeightsFlowsNormals(WaterObject, ref Vector3[], ref float[], ref Vector3[], ref Vector3[], bool, bool, bool) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeightsFlowsNormals_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___System_Single___UnityEngine_Vector3___UnityEngine_Vector3___System_Boolean_System_Boolean_System_Boolean_)
WaterDataProvider.GetWaterHeightSingle(WaterObject, Vector3) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeightSingle_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3_)
WaterDataProvider.PointInWater(WaterObject, Vector3) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_PointInWater_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3_)
WaterDataProvider.GetWaterHeight(WaterObject, Vector3) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeight_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3_)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterData (NWH.DWP2.WaterData.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class RaycastWaterDataProvider : WaterDataProvider
```

Fields

_flow

Declaration

```
protected Vector3 _flow
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

_hit

Declaration

```
protected RaycastHit _hit
```

Field Value

Type	Description
RaycastHit (https://docs.unity3d.com/ScriptReference/RaycastHit.html)	

_layerMask

Declaration

```
protected LayerMask _layerMask
```

Field Value

Type	Description
LayerMask (https://docs.unity3d.com/ScriptReference/LayerMask.html)	

_mesh

Declaration

```
protected Mesh _mesh
```

Field Value

Type	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	

_normals

Declaration

```
protected Vector3[] _normals
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

_prevDataSize

Declaration

```
protected int _prevDataSize
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

_queryParameters

Declaration

```
protected QueryParameters _queryParameters
```

Field Value

Type	Description
QueryParameters (https://docs.unity3d.com/ScriptReference/QueryParameters.html)	

_ray

Declaration

```
protected Ray _ray
```

Field Value

Type	Description
Ray (https://docs.unity3d.com/ScriptReference/Ray.html)	

_rayDirection

Declaration

```
protected Vector3 _rayDirection
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

_rayStartOffset

Declaration

```
protected Vector3 _rayStartOffset
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

_raycastCommands

Declaration

```
protected NativeArray<RaycastCommand> _raycastCommands
```

Field Value

Type	Description
NativeArray (https://docs.unity3d.com/ScriptReference/Unity.Collections.NativeArray_1.html)< RaycastCommand (https://docs.unity3d.com/ScriptReference/RaycastCommand.html)>	

_raycastHits

Declaration

```
protected NativeArray<RaycastHit> _raycastHits
```

Field Value

Type	Description
NativeArray (https://docs.unity3d.com/ScriptReference/Unity.Collections.NativeArray_1.html)< RaycastHit (https://docs.unity3d.com/ScriptReference/RaycastHit.html)>	

_raycastJobHandle

Declaration

```
protected JobHandle _raycastJobHandle
```

Field Value

Type	Description
JobHandle (https://docs.unity3d.com/ScriptReference/Unity.Jobs.JobHandle.html)	

_tmp

Declaration

```
protected Vector3 _tmp
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

_tmpCommand

Declaration

```
protected RaycastCommand _tmpCommand
```

Field Value

Type	Description
RaycastCommand (https://docs.unity3d.com/ScriptReference/RaycastCommand.html)	

_upVector

Declaration

```
protected Vector3 _upVector
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

_uv4

Declaration

```
protected Vector2 _uv4
```

Field Value

Type	Description
Vector2 (https://docs.unity3d.com/ScriptReference/Vector2.html)	

_vertDir

Declaration

```
protected Vector3 _vertDir
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

_vertIndex

Declaration

```
protected int _vertIndex
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

_zeroVector

Declaration

```
protected Vector3 _zeroVector
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

commandsPerJob

Number of raycast commands to process per job batch. Higher values may improve performance on systems with more CPU cores.

Declaration

```
[Tooltip("Minimum number of RaycastCommands per job.")]
public int commandsPerJob
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

objectLayer

Layer that floating objects are on. Used to prevent physics collisions between water and objects.

Declaration

```
[Tooltip("    Layer the water object(s) are on. Required to be able to disable physics colli
sions between water and object.")]
public int objectLayer
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

raycastDistance

Maximum distance raycasts can travel in each direction. Raycasts extend this distance above and below each point. Lower values improve performance but may miss water surfaces that are far from the object.

Declaration

```
[Tooltip("    Raycasts will start at this distance above the point and extend this distance
below the point. This means\r\n    that if the water surface is raycastDistance below or abo
ve the point, it will not be detected.\r\n    Using lower value will slightly improve perfor
mance of Raycasts.")]
public float raycastDistance
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

waterLayer

Layer that water surface colliders are on. Used to filter raycasts and prevent physics collisions.

Declaration

```
[Tooltip("    Layer the water is on. Required to be able to disable physics collisions betwe
en water and object.")]
public int waterLayer
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

Methods

Awake()

Declaration

```
public override void Awake()
```

Overrides

WaterDataProvider.Awake()
(NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_Awake)

Deallocate()

Deallocates native arrays used for raycasting. Called automatically on disable or destroy.

Declaration

```
public virtual void Deallocate()
```

GetWaterHeights(WaterObject, ref Vector3[], ref float[])

Returns water height at each given point. Override this method to provide water height data from your water system.

Declaration

```
public override void GetWaterHeights(WaterObject waterObject, ref Vector3[] points, ref floa
t[] waterHeights)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*points*	Position array in world coordinates.
float (https://learn.microsoft.com/dotnet/api/system.single) []	*waterHeights*	Water height array in world coordinates. Corresponds to positions.

Overrides

WaterDataProvider.GetWaterHeights(WaterObject, ref Vector3[], ref float[]) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterHeights_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___System_Single___)

GetWaterNormals(WaterObject, ref Vector3[], ref Vector3[])

Returns water surface normals at each given point. Override this method to provide water normal data from your water system.

Declaration

```
public override void GetWaterNormals(WaterObject waterObject, ref Vector3[] points, ref Vect
or3[] waterNormals)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*points*	Position array in world coordinates.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*waterNormals*	Water surface normal array in world coordinates. Corresponds to positions.

Overrides

WaterDataProvider.GetWaterNormals(WaterObject, ref Vector3[], ref Vector3[]) (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_GetWaterNormals_NWH_DWP2_WaterObjects_WaterObject_UnityEngine_Vector3___UnityEngine_Vector3___)

OnDestroy()

Declaration

```
public virtual void OnDestroy()
```

OnDisable()

Declaration

```
public virtual void OnDisable()
```

SupportsWaterFlowQueries()

Does this water system support water velocity queries?

Declaration

```
public override bool SupportsWaterFlowQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

Overrides

WaterDataProvider.SupportsWaterFlowQueries() (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_SupportsWaterFlowQueries)

SupportsWaterHeightQueries()

Does this water system support water height queries?

Declaration

```
public override bool SupportsWaterHeightQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

Overrides

WaterDataProvider.SupportsWaterHeightQueries() (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_SupportsWaterHeightQueries)

SupportsWaterNormalQueries()

Does this water system support water normal queries?

Declaration

```
public override bool SupportsWaterNormalQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

Overrides

WaterDataProvider.SupportsWaterNormalQueries() (NWH.DWP2.WaterData.WaterDataProvider.html#NWH_DWP2_WaterData_WaterDataProvider_SupportsWaterNormalQueries)

Class WaterDataProvider

Base class for providing water surface data to WaterObjects. Implementations provide water height, flow velocity, and surface normal information. Uses a trigger collider to automatically detect and register WaterObjects that enter the water. Inherit from this class to create adapters for different water systems.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WaterDataProvider
            ↳ FlatWaterDataProvider (NWH.DWP2.WaterData.FlatWaterDataProvider.html)
            ↳ RaycastWaterDataProvider (NWH.DWP2.WaterData.RaycastWaterDataProvider.html)

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterData (NWH.DWP2.WaterData.html)

Assembly: NWH.DWP2.dll

Syntax

```
public abstract class WaterDataProvider : MonoBehaviour
```

Fields

_singleHeightArray

Declaration

```
protected float[] _singleHeightArray
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)[]	

_singlePointArray

Declaration

```
protected Vector3[] _singlePointArray
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

_triggerCollider

Declaration

```
protected Collider _triggerCollider
```

Field Value

Type	Description
Collider (https://docs.unity3d.com/ScriptReference/Collider.html)	

Methods

Awake()

Declaration

```
public virtual void Awake()
```

GetWaterFlows(WaterObject, ref Vector3[], ref Vector3[])

Returns water flow velocity at each given point. Override this method to provide water flow data from your water system. Flow should be in world space and relative to the world, not the WaterObject.

Declaration

```
public virtual void GetWaterFlows(WaterObject waterObject, ref Vector3[] points, ref Vector3
[] waterFlows)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*points*	Position array in world coordinates.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*waterFlows*	Water flow velocity array in world coordinates. Corresponds to positions.

GetWaterHeight(WaterObject, Vector3)

Returns water height at the given world position.

Declaration

```
public float GetWaterHeight(WaterObject waterObject, Vector3 worldPoint)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject making the query.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*worldPoint*	Position in world coordinates.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Water surface height at the given point.

GetWaterHeightSingle(WaterObject, Vector3)

Returns water height at a single point. Less efficient than batch queries but useful for one-off checks.

Declaration

```
public virtual float GetWaterHeightSingle(WaterObject waterObject, Vector3 point)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*point*	Position in world coordinates.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Water height at the given point.

GetWaterHeights(WaterObject, ref Vector3[], ref float[])

Returns water height at each given point. Override this method to provide water height data from your water system.

Declaration

```
public virtual void GetWaterHeights(WaterObject waterObject, ref Vector3[] points, ref float
[] waterHeights)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*points*	Position array in world coordinates.
float (https://learn.microsoft.com/dotnet/api/system.single) []	*waterHeights*	Water height array in world coordinates. Corresponds to positions.

GetWaterHeightsFlowsNormals(WaterObject, ref Vector3[], ref float[], ref Vector3[], ref Vector3[], bool, bool, bool)

Queries water data based on enabled flags. Calls the appropriate getter methods based on which data types are requested.

Declaration

```
public void GetWaterHeightsFlowsNormals(WaterObject waterObject, ref Vector3[] points, ref f
loat[] waterHeights, ref Vector3[] waterFlows, ref Vector3[] waterNormals, bool useWaterHeig
ht, bool useWaterNormals, bool useWaterFlow)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*points*	Position array in world coordinates.
float (https://learn.microsoft.com/dotnet/api/system.single) []	*waterHeights*	Output water heights array.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*waterFlows*	Output water flows array.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*waterNormals*	Output water normals array.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*useWaterHeight*	Should water heights be queried.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*useWaterNormals*	Should water normals be queried.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*useWaterFlow*	Should water flows be queried.

GetWaterNormals(WaterObject, ref Vector3[], ref Vector3[])

Returns water surface normals at each given point. Override this method to provide water normal data from your water system.

Declaration

```
public virtual void GetWaterNormals(WaterObject waterObject, ref Vector3[] points, ref Vecto
r3[] waterNormals)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject requesting the data.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*points*	Position array in world coordinates.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html) []	*waterNormals*	Water surface normal array in world coordinates. Corresponds to positions.

PointInWater(WaterObject, Vector3)

Checks if a point is underwater. Accounts for wave height when water normals are supported.

Declaration

```
public bool PointInWater(WaterObject waterObject, Vector3 worldPoint)
```

Parameters

Type	Name	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	*waterObject*	WaterObject making the query.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*worldPoint*	Position to check in world coordinates.

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if the point is below the water surface.

SupportsWaterFlowQueries()

Does this water system support water velocity queries?

Declaration

```
public abstract bool SupportsWaterFlowQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

SupportsWaterHeightQueries()

Does this water system support water height queries?

Declaration

```
public abstract bool SupportsWaterHeightQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

SupportsWaterNormalQueries()

Does this water system support water normal queries?

Declaration

```
public abstract bool SupportsWaterNormalQueries()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if it does, false if it does not.

Namespace NWH.DWP2.WaterObjects

Classes

MassFromChildren (NWH.DWP2.WaterObjects.MassFromChildren.html)

Calculates mass of a Rigidbody from the children that have MassFromVolume script attached.

MassFromVolume (NWH.DWP2.WaterObjects.MassFromVolume.html)

Script to calculate mass from mesh volume and material density. Removes need for guessing mass and ensures that all objects of same density are submerged equally.

MeshUtility (NWH.DWP2.WaterObjects.MeshUtility.html)

Utility class for mesh processing operations used by WaterObject. Handles mesh simplification, convexification, vertex welding, and volume calculations.

SerializedMesh (NWH.DWP2.WaterObjects.SerializedMesh.html)

Serializable representation of a mesh. Used to store simulation mesh data without keeping mesh references in game files.

WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)

Core component for mesh-based buoyancy simulation. Calculates buoyancy and hydrodynamic forces by slicing mesh triangles at the water surface. Requires a MeshFilter with a valid mesh and a Rigidbody to apply forces to. Use WaterObjectWizard for one-click setup or manually configure the simulation mesh.

WaterObjectMaterial (NWH.DWP2.WaterObjects.WaterObjectMaterial.html)

ScriptableObject defining material properties for mass calculation. Contains density value used by MassFromVolume to calculate object mass. Create via Assets > Create > Dynamic Water Physics 2 > Water Object Material.

WaterObjectWizard (NWH.DWP2.WaterObjects.WaterObjectWizard.html)

Configuration for the WaterObject setup wizard. Used by the editor to automate WaterObject component setup.

WaterParticleSystem (NWH.DWP2.WaterObjects.WaterParticleSystem.html)

Generates water spray and splash particles based on WaterObject simulation data. Emits particles at the waterline where triangles intersect the water surface. Particle emission is controlled by object velocity and triangle forces. Requires a ParticleSystem component.

Class MassFromChildren

Calculates mass of a Rigidbody from the children that have MassFromVolume script attached.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ MassFromChildren

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(Rigidbody))]
public class MassFromChildren : MonoBehaviour
```

Methods

Calculate()

Calculates the total mass from all child MassFromVolume components and applies it to the Rigidbody. Also updates the VariableCenterOfMass component if present.

Declaration

```
public void Calculate()
```

Class MassFromVolume

Script to calculate mass from mesh volume and material density. Removes need for guessing mass and ensures that all objects of same density are submerged equally.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ MassFromVolume

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(WaterObject))]
public class MassFromVolume : MonoBehaviour
```

Fields

density

Calculated density of the material in kg/m³.

Declaration

```
public float density
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

mass

Calculated mass of the object in kg.

Declaration

```
public float mass
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

material

Material preset containing density value.

Declaration

```
public WaterObjectMaterial material
```

Field Value

Type	Description
WaterObjectMaterial (NWH.DWP2.WaterObjects.WaterObjectMaterial.html)	

volume

Calculated volume of the simulation mesh in m^3.

Declaration

```
public float volume
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Methods

CalculateAndApplyFromDensity(float)

Calculates mass from the given density and applies it to the Rigidbody.

Declaration

```
public float CalculateAndApplyFromDensity(float density)
```

Parameters

Type	Name	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	*density*	Material density in kg/m^3.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Calculated mass value.

CalculateAndApplyFromMaterial()

Calculates mass from the assigned material's density and applies it to the Rigidbody.

Declaration

```
public float CalculateAndApplyFromMaterial()
```

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Calculated mass value.

CalculateSimulationMeshVolume()

Gets volume of the simulation mesh. Scale-sensitive.

Declaration

```
public void CalculateSimulationMeshVolume()
```

SetDefaultAsMaterial()

Sets the default WaterObjectMaterial from Resources.

Declaration

```
public void SetDefaultAsMaterial()
```

Class MeshUtility

Utility class for mesh processing operations used by WaterObject. Handles mesh simplification, convexification, vertex welding, and volume calculations.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ MeshUtility

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
public static class MeshUtility
```

Methods

GenerateConvexMesh(Mesh)

Creates a convex hull from the input mesh. Useful for partially hollow meshes or open hulls.

Declaration

```
public static Mesh GenerateConvexMesh(Mesh mesh)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*mesh*	Mesh to convexify.

Returns

Type	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	Convex mesh wrapping the input mesh.

GenerateMesh(Vector3[], int[])

Generate mesh from vertices and triangles.

Declaration

```
public static Mesh GenerateMesh(Vector3[] vertices, int[] triangles)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	*vertices*	Array of vertices.
int (https://learn.microsoft.com/dotnet/api/system.int32)[]	*triangles*	Array of triangles (indices).

Returns

Type	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	

GenerateSimMesh(ref Mesh, ref Mesh, bool, bool, bool, float)

Generates a simulation mesh from the original mesh with optional processing steps. Processes the mesh according to the specified flags and simplification ratio.

Declaration

```
public static void GenerateSimMesh(ref Mesh originalMesh, ref Mesh simMesh, bool simplifyMes
h = false, bool convexifyMesh = false, bool weldColocatedVertices = true, float simplificati
onRatio = 1)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*originalMesh*	Source mesh to process.
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*simMesh*	Output simulation mesh.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*simplifyMesh*	Should the mesh be simplified to reduce triangle count.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*convexifyMesh*	Should the mesh be made convex.
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	*weldColocatedVertices*	Should vertices at the same position be merged.
float (https://learn.microsoft.com/dotnet/api/system.single)	*simplificationRatio*	Target triangle count as ratio of original (0-1).

SignedVolumeOfTriangle(Vector3, Vector3, Vector3)

Calculates the signed volume of a triangle with respect to the origin. Used for mesh volume calculations.

Declaration

```
public static float SignedVolumeOfTriangle(Vector3 p1, Vector3 p2, Vector3 p3)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p1*	First vertex.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p2*	Second vertex.
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*p3*	Third vertex.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Signed volume of the triangle.

VolumeOfMesh(Mesh, Transform)

Calculates the volume of a mesh in world space. Takes into account the transform's scale, position and rotation.

Declaration

```
public static float VolumeOfMesh(Mesh mesh, Transform transform)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*mesh*	Mesh to calculate volume for.
Transform (https://docs.unity3d.com/ScriptReference/Transform.html)	*transform*	Transform containing scale information.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Volume of the mesh in cubic meters.

WeldVertices(ref Mesh, float)

Merges vertices that are closer than the specified distance threshold. Improves performance by reducing vertex count and removing duplicates.

Declaration

```
public static void WeldVertices(ref Mesh aMesh, float aMaxDelta = 0.001)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*aMesh*	Mesh to process.
float (https://learn.microsoft.com/dotnet/api/system.single)	*aMaxDelta*	Maximum distance between vertices to be considered duplicates.

Class SerializedMesh

Serializable representation of a mesh. Used to store simulation mesh data without keeping mesh references in game files.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ SerializedMesh

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
[Serializable]
public class SerializedMesh
```

Fields

triangles

Triangle indices of the mesh.

Declaration

```
[SerializeField]
public int[] triangles
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)[]	

vertices

Vertex positions of the mesh.

Declaration

```
[SerializeField]
public Vector3[] vertices
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

Methods

Deserialize()

Deserializes the stored data back into a Mesh object.

Declaration

```
public Mesh Deserialize()
```

Returns

Type	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	Reconstructed mesh or null if data is invalid.

Serialize(Mesh)

Serializes a mesh into vertex and triangle arrays.

Declaration

```
public void Serialize(Mesh mesh)
```

Parameters

Type	Name	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	*mesh*	Mesh to serialize.

Class WaterObject

Core component for mesh-based buoyancy simulation. Calculates buoyancy and hydrodynamic forces by slicing mesh triangles at the water surface. Requires a MeshFilter with a valid mesh and a Rigidbody to apply forces to. Use WaterObjectWizard for one-click setup or manually configure the simulation mesh.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WaterObject

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(MeshFilter))]
[DisallowMultipleComponent]
public class WaterObject : MonoBehaviour
```

Fields

CurrentWaterDataProvider

Currently active WaterDataProvider.

Declaration

```
[NonSerialized]
[Tooltip("Currently active WaterDataProvider.")]
public WaterDataProvider CurrentWaterDataProvider
```

Field Value

Type	Description
WaterDataProvider (NWH.DWP2.WaterData.WaterDataProvider.html)	

LocalVertices

Vertices used for simulation, in local space.

Declaration

```
[NonSerialized]
[Tooltip("Vertices used for simulation, in local space.")]
public Vector3[] LocalVertices
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

ResultAreas

Result triangle areas.

Declaration

```
[NonSerialized]
[Tooltip("Result triangle areas.")]
public float[] ResultAreas
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)[]	

ResultCenters

Result triangle centers in world coordinates.

Declaration

```
[NonSerialized]
[Tooltip("Result triangle centers in world coordinates.")]
public Vector3[] ResultCenters
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

ResultDistances

Result distances to water surface.

Declaration

```
[NonSerialized]
[Tooltip("Result distances to water surface.")]
public float[] ResultDistances
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)[]	

ResultForce

Total result force.

Declaration

```
[NonSerialized]
[Tooltip("Total result force.")]
public Vector3 ResultForce
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

ResultForces

Per-triangle result forces.

Declaration

```
[NonSerialized]
[Tooltip("Per-triangle result forces.")]
public Vector3[] ResultForces
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

ResultNormals

Result triangle normals. Not equal to the mesh normals.

Declaration

```
[NonSerialized]
[Tooltip("Result triangle normals. Not equal to the mesh normals.")]
public Vector3[] ResultNormals
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

ResultP0s

Result sliced triangle vertices.

Declaration

```
[NonSerialized]
[Tooltip("Result sliced triangle vertices.")]
public Vector3[] ResultP0s
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

ResultStates

Result triangle states. 0 - Triangle is underwater 1 - Triangle is partially underwater 2 - Triangle is above water 3 - Triangle's object is disabled 4 - Triangle's object is deleted

Declaration

```
[NonSerialized]
[Tooltip("Result triangle states.\r\n0 - Triangle is underwater\r\n1 - Triangle is partially
underwater\r\n2 - Triangle is above water\r\n3 - Triangle's object is disabled\r\n4 - Triang
le's object is deleted")]
public int[] ResultStates
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)[]	

ResultTorque

Result total torque acting on the Rigidbody.

Declaration

```
[NonSerialized]
[Tooltip("Result total torque acting on the Rigidbody.")]
public Vector3 ResultTorque
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

ResultVelocities

Result velocities at triangle centers.

Declaration

```
[NonSerialized]
[Tooltip("Result velocities at triangle centers.")]
public Vector3[] ResultVelocities
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

RigidbodyAngVel

Angular velocity of the target Rigidbody.

Declaration

```
[NonSerialized]
[Tooltip("Angular velocity of the target Rigidbody.")]
public Vector3 RigidbodyAngVel
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

RigidbodyCoM

Center-of-mass of the target Rigidbody.

Declaration

```
[NonSerialized]
[Tooltip("Center of mass of the target Rigidbody.")]
public Vector3 RigidbodyCoM
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

RigidbodyLinearVel

(Linear) velocity of the target Rigidbody.

Declaration

```
[NonSerialized]
[Tooltip("(Linear) velocity of the target Rigidbody.")]
public Vector3 RigidbodyLinearVel
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

TriIndices

Triangles indices of the simulation mesh.

Declaration

```
[NonSerialized]
[Tooltip("Triangles indices of the simulation mesh.")]
public int[] TriIndices
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)[]	

WaterFlows

Water flows of the current water data provider. Default is used if no water data provider is present.

Declaration

```
[NonSerialized]
[Tooltip("Water flows of the current water data provider. Default is used if no water data p
rovider is present.")]
public Vector3[] WaterFlows
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

WaterHeights

Water surface heights of the current water data provider. Default is used if no water data provider is present.

Declaration

```
[NonSerialized]
[Tooltip("Water surface heights of the current water data provider. Default is used if no wa
ter data provider is present.")]
public float[] WaterHeights
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)[]	

WaterNormals

Water normals of the current water data provider. Default is used if no water data provider is present.

Declaration

```
[NonSerialized]
[Tooltip("Water normals of the current water data provider. Default is used if no water data
provider is present.")]
public Vector3[] WaterNormals
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

WorldVertices

Simulation vertices converted to world coordinates.

Declaration

```
[NonSerialized]
[Tooltip("Simulation vertices converted to world coordinates.")]
public Vector3[] WorldVertices
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)[]	

buoyantForceCoefficient

Coefficient by which the buoyancy forces are multiplied.

Declaration

```
[Tooltip("Coefficient by which the the buoyancy forces are multiplied.")]
public float buoyantForceCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

calculateWaterFlows

Should the water flows be queried and calculated?

Declaration

```
[Tooltip("Should the water flows be queried and calculated?")]
public bool calculateWaterFlows
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

calculateWaterHeights

Should the water heights be queried and calculated?

Declaration

```
[Tooltip("Should the water heights be queried and calculated?")]
public bool calculateWaterHeights
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

calculateWaterNormals

Should the water normals be queried and calculated?

Declaration

```
[Tooltip("Should the water normals be queried and calculated?")]
public bool calculateWaterNormals
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

convexifyMesh

Should the simulation mesh be made convex? Can be useful for half-open hulls and otherwise partially hollow meshes.

Declaration

```
[Tooltip("Should the simulation mesh be made convex?\r\nCan be useful for half-open hulls an
d otherwise partially hollow meshes.")]
public bool convexifyMesh
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

defaultWaterFlow

World water flow to be used when there is no WaterDataProvider present and calculateWaterFlows is enabled.

Declaration

```
[Tooltip("World water flow to be used when there is no WaterDataProvider present and calcula
teWaterFlows is enabled.")]
public Vector3 defaultWaterFlow
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

defaultWaterHeight

World water height to be used when there is no WaterDataProvider present.

Declaration

```
[Tooltip("World water height to be used when there is no WaterDataProvider present.")]
public float defaultWaterHeight
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

defaultWaterNormal

World water normal to be used when there is no WaterDataProvider present and calculateWaterNormals is enabled.

Declaration

```
[Tooltip("World water normal to be used when there is no WaterDataProvider present and calcu
lateWaterNormals is enabled.")]
public Vector3 defaultWaterNormal
```

Field Value

Type	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	

finalForceCoefficient

Coefficient by which the result force will get multiplied.

Declaration

```
[Tooltip("Coefficient by which the result force will get multiplied.")]
public float finalForceCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

finalTorqueCoefficient

The coefficient by which the torque force will get multiplied.

Declaration

```
[Tooltip("Coefficient by which the torque force will get multiplied.")]
public float finalTorqueCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

fluidDensity

Density of the fluid the object is in.

Declaration

```
[Tooltip("Density of the fluid the object is in.")]
public float fluidDensity
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

hydrodynamicForceCoefficient

Coefficient by which the hydrodynamic forces are multiplied.

Declaration

```
[Tooltip("Coefficient by which the hydrodynamic forces are multiplied..")]
public float hydrodynamicForceCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

instanceID

Cached instance ID of the GameObject.

Declaration

```
[NonSerialized]
[Tooltip("Cached instance ID of the GameObject.")]
public int instanceID
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

originalMesh

Original mesh of the object, non-simplified and non-convexified.

Declaration

```
[SerializeField]
[Tooltip("Original mesh of the object, non-simplified and non-convexified.")]
public Mesh originalMesh
```

Field Value

Type	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	

serializedSimulationMesh

Serialized version of the simulation mesh, used to prevent having to store the mesh in game files.

Declaration

```
[SerializeField]
[Tooltip("Serialized version of the simulation mesh, used to prevent having to store the mes
h in game files.")]
public SerializedMesh serializedSimulationMesh
```

Field Value

Type	Description
SerializedMesh (NWH.DWP2.WaterObjects.SerializedMesh.html)	

simplifyMesh

Should the simulation mesh be simplified / decimated?

Declaration

```
[Tooltip("Should the simulation mesh be simplified / decimated?")]
public bool simplifyMesh
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

skinDragCoefficient

The coefficient by which the skin drag force is multiplied. Skin drag is a force caused by water flowing over a surface.

Declaration

```
[Tooltip("Coefficient by which the skin drag force is multiplied.\r\nSkin drag is a force ca
used by water flowing over a surface.")]
public float skinDragCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

slamForceCoefficient

Coefficient applied to forces when a face of the object is entering the water.

Declaration

```
[Tooltip("Coefficient applied to forces when a face of the object is entering the water.")]
public float slamForceCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

submergedVolume

Current submerged volume in cubic meters. Updated each physics step based on how much of the mesh is underwater.

Declaration

```
[Tooltip("Resulting volume of buoyancy calculation.")]
public float submergedVolume
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

suctionForceCoefficient

Coefficient applied to forces when a face of the object is leaving the water.

Declaration

```
[Tooltip("Coefficient applied to forces when a face of the object is leaving the water.")]
public float suctionForceCoefficient
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

targetRigidbody

Rigidbody that the forces will be applied to.

Declaration

```
[Tooltip("Rigidbody that the forces will be applied to.")]
public Rigidbody targetRigidbody
```

Field Value

Type	Description
Rigidbody (https://docs.unity3d.com/ScriptReference/Rigidbody.html)	

targetTriangleCount

Target triangle count for the simulation mesh. The original mesh will be decimated to this number of triangles is "SimplifyMesh" is enabled.

Declaration

```
[FormerlySerializedAs("targetTris")]
[Range(8, 256)]
[Tooltip("Target triangle count for the simulation mesh.\r\nOriginal mesh will be decimated
to this number of triangles is 'SimplifyMesh' is enabled.\r\nOtherwise does nothing.")]
public int targetTriangleCount
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

triangleCount

Total number of triangles (not indices) on the simulation mesh.

Declaration

```
[Tooltip("Total number of triangles (not indices) on the simulation mesh.")]
public int triangleCount
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

velocityDotPower

Determines to which power the dot product between velocity and triangle normal will be raised. Higher values will result in lower hydrodynamic forces for situations in which the triangle is nearly parallel to the water velocity.

Declaration

```
[Range(0.01, 2)]
[Tooltip("Determines to which power the dot product between velocity and triangle normal wil
l be raised.\r\nHigher values will result in lower hydrodynamic forces for situations in whi
ch the triangle is nearly parallel to the\r\nwater velocity.")]
public float velocityDotPower
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

vertexCount

Total number of vertices on the simulation mesh.

Declaration

```
[Tooltip("Total number of vertices on the simulation mesh.")]
public int vertexCount
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

weldColocatedVertices

If true, the vertices with the same position will be welded. Improves performance and is highly recommended.

Declaration

```
[Tooltip("    If true vertices with same position will be welded.\r\n    Improves performanc
e and is highly recommended.")]
public bool weldColocatedVertices
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Properties

PreviewEnabled

Returns true if the simulation mesh is currently being previewed in the scene view.

Declaration

```
public bool PreviewEnabled { get; }
```

Property Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

SimulationMesh

Gets the simulation mesh, deserializing from serialized data if necessary.

Declaration

```
public Mesh SimulationMesh { get; }
```

Property Value

Type	Description
Mesh (https://docs.unity3d.com/ScriptReference/Mesh.html)	

Methods

GenerateSimMesh()

Generates or regenerates the simulation mesh based on current settings. Applies simplification, convexification, and vertex welding as configured. Must be called after changing mesh processing settings.

Declaration

```
public void GenerateSimMesh()
```

GetWaterHeightSingle(Vector3)

Returns water height at a single point using the current WaterDataProvider. Falls back to defaultWaterHeight if no provider is active.

Declaration

```
public float GetWaterHeightSingle(Vector3 point)
```

Parameters

Type	Name	Description
Vector3 (https://docs.unity3d.com/ScriptReference/Vector3.html)	*point*	Position in world coordinates.

Returns

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	Water height at the given point.

IsTouchingWater()

Checks if any part of the object is currently in contact with water. Returns true if at least one triangle is fully or partially submerged. Cache the result if calling frequently.

Declaration

```
public bool IsTouchingWater()
```

Returns

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	True if object is touching water.

OnEnterWaterDataProvider(WaterDataProvider)

Called when the WaterObject enters a WaterDataProvider's trigger volume. Registers the provider for water data queries.

Declaration

```
public void OnEnterWaterDataProvider(WaterDataProvider wdp)
```

Parameters

Type	Name	Description
WaterDataProvider (NWH.DWP2.WaterData.WaterDataProvider.html)	*wdp*	WaterDataProvider that was entered.

OnExitWaterDataProvider(WaterDataProvider)

Called when the WaterObject exits a WaterDataProvider's trigger volume. Unregisters the provider and resets to default water data if no providers remain.

Declaration

```
public void OnExitWaterDataProvider(WaterDataProvider wdp)
```

Parameters

Type	Name	Description
WaterDataProvider (NWH.DWP2.WaterData.WaterDataProvider.html)	*wdp*	WaterDataProvider that was exited.

StartSimMeshPreview()

Enables visualization of the simulation mesh in the scene view. Temporarily replaces the MeshFilter's mesh with the simulation mesh.

Declaration

```
public void StartSimMeshPreview()
```

StopSimMeshPreview()

Disables simulation mesh visualization in the scene view. Restores the original mesh to the MeshFilter.

Declaration

```
public void StopSimMeshPreview()
```

See Also

WaterObjectWizard (NWH.DWP2.WaterObjects.WaterObjectWizard.html)
WaterDataProvider (NWH.DWP2.WaterData.WaterDataProvider.html)
WaterObjectMaterial()
AdvancedShipController (NWH.DWP2.ShipController.AdvancedShipController.html)

Class WaterObjectMaterial

ScriptableObject defining material properties for mass calculation. Contains density value used by MassFromVolume to calculate object mass. Create via Assets > Create > Dynamic Water Physics 2 > Water Object Material.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ ScriptableObject (https://docs.unity3d.com/ScriptReference/ScriptableObject.html)
      ↳ WaterObjectMaterial

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
[CreateAssetMenu(fileName = "WaterObjectMaterial", menuName = "Dynamic Water Physics 2/Water
Object Material", order = 0)]
public class WaterObjectMaterial : ScriptableObject
```

Fields

density

Material density in kg/m³. Used with mesh volume to calculate realistic object mass. Examples: Wood ~600, Ice ~920, Aluminum ~2700, Steel ~7850.

Declaration

```
public float density
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

Class WaterObjectWizard

Configuration for the WaterObject setup wizard. Used by the editor to automate WaterObject component setup.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WaterObjectWizard

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
public class WaterObjectWizard : MonoBehaviour
```

Fields

addWaterParticleSystem

Should the wizard add a WaterParticleSystem component for splash effects.

Declaration

```
public bool addWaterParticleSystem
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

Class WaterParticleSystem

Generates water spray and splash particles based on WaterObject simulation data. Emits particles at the waterline where triangles intersect the water surface. Particle emission is controlled by object velocity and triangle forces. Requires a ParticleSystem component.

Inheritance

↳ object (https://learn.microsoft.com/dotnet/api/system.object)
  ↳ Object (https://docs.unity3d.com/ScriptReference/Object.html)
    ↳ Component (https://docs.unity3d.com/ScriptReference/Component.html)
      ↳ Behaviour (https://docs.unity3d.com/ScriptReference/Behaviour.html)
        ↳ MonoBehaviour (https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)
          ↳ WaterParticleSystem

Namespace: NWH (NWH.html).DWP2 (NWH.DWP2.html).WaterObjects (NWH.DWP2.WaterObjects.html)

Assembly: NWH.DWP2.dll

Syntax

```
[RequireComponent(typeof(ParticleSystem))]
public class WaterParticleSystem : MonoBehaviour
```

Fields

ReferenceWaterObject

WaterObject to read simulation data from. Will auto-detect from parent if not assigned.

Declaration

```
public WaterObject ReferenceWaterObject
```

Field Value

Type	Description
WaterObject (NWH.DWP2.WaterObjects.WaterObject.html)	

emit

Should the particle system emit?

Declaration

```
[Tooltip("Should the particle system emit?")]
public bool emit
```

Field Value

Type	Description
bool (https://learn.microsoft.com/dotnet/api/system.boolean)	

emitPerCycle

How many particles should be emitted each 'emitTimeInterval' seconds.

Declaration

```
[Tooltip("How many particles should be emitted each 'emitTimeInterval' seconds.")]
[Range(0, 20)]
public int emitPerCycle
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

emitTimeInterval

Determines how often the particles will be emitted.

Declaration

```
[Tooltip("Determines how often the particles will be emitted.")]
[Range(0, 0.1)]
public float emitTimeInterval
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

initialAlphaModifier

Multiplies initial alpha by this value. Alpha cannot be higher than maxInitialAlpha.

Declaration

```
[Tooltip("Multiplies initial alpha by this value. Alpha cannot be higher than maxInitialAlph
a.")]
[Range(0, 10)]
public float initialAlphaModifier
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

initialVelocityModifier

Determines how much velocity of the object will affect initial particle speed.

Declaration

```
[Tooltip("Determines how much velocity of the object will affect initial particle speed.")]
[Range(0, 5)]
public float initialVelocityModifier
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

maxInitialAlpha

Limit initial alpha to this value.

Declaration

```
[Tooltip("Limit initial alpha to this value.")]
[Range(0, 1)]
public float maxInitialAlpha
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

positionExtrapolationFrames

Number of frames ahead to predict particle emission position. Higher values spawn particles ahead of the object for better visual continuity.

Declaration

```
[Tooltip("Script will try to predict where the object will be in the next n frames.")]
public int positionExtrapolationFrames
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

renderQueue

Render queue of the particle material.

Declaration

```
[Tooltip("Render queue of the particle material.")]
public int renderQueue
```

Field Value

Type	Description
int (https://learn.microsoft.com/dotnet/api/system.int32)	

sleepThresholdVelocity

Velocity object has to have to emit particles.

Declaration

```
[FormerlySerializedAs("sleepTresholdVelocity")]
[Tooltip("Velocity object has to have to emit particles.")]
[Range(0.1, 5)]
public float sleepThresholdVelocity
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

startSize

Initial size of the particle.

Declaration

```
[Tooltip("Initial size of the particle.")]
[Range(0, 64)]
public float startSize
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	

surfaceElevation

Elevation above water at which the particles will spawn. Used to avoid clipping.

Declaration

```
[Tooltip("Elevation above water at which the particles will spawn. Used to avoid clippin
g.")]
[Range(0, 0.1)]
public float surfaceElevation
```

Field Value

Type	Description
float (https://learn.microsoft.com/dotnet/api/system.single)	