

Ship Controller Manual

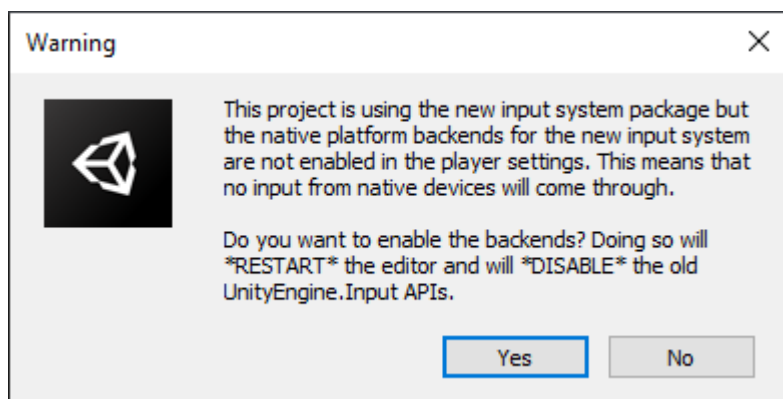
Input

- Ships retrieve user input through Input class which retrieves input from active InputProvider and fills the InputStates struct with the retrieved data.
- InputProviders are split into *ShipInputProviders* and *SceneInputProviders*. *ShipInputProviders* relay ship input (throttle, brakes, etc.) while *SceneInputProviders* take care of scene input (ship changing, camera changing, camera movement and the rest of the inputs not directly related to ship. One of each needs to be present (e.g. InputSystemShipInputProvider and InputSystemSceneInputProvider).
- Multiple different InputProviders can be present in the scene (v1.0 or newer required). E.g. InputSystemProviders and MobileInputProviders can be used in the same scene. The resulting input will be a sum of inputs from all InputProviders in case of numeric inputs and logical OR operation of all inputs in case of boolean inputs.
- Input is stored inside InputStates object and can be copied over from one ship to another.
- To manually set the InputStates make sure Auto Settable is set to false.

All input providers inherit from either ShipInputProviderBase or SceneInputProviderBase, but differ in their implementation.

Input System Warning

When importing the asset for the first time this message will pop up:



Both Yes or No can be selected but it is important to set the *Project Settings* ⇒ *Player* ⇒ *Input Handling* to Both afterwards. This way both new InputSystem and the old InputManager will work. If this setting is set to InputManager only errors might appear as the demo scenes of the asset rely on InputSystem.

If a message *This Unity Package has Package Manager dependencies.* appears, click Install/Upgrade.

Available Bindings

Ship Input Provider Bindings

Out of the box gamepad bindings are only available for InputSystem.

Name	Type	Keyboard Defaults	Gamepad Defaults	Description
Steering	axis [-1,1]	A/D	Left Stick - Left/Right	Steering/rudder. Positive right.
Throttle	axis [-1,1]	S/W	Left/Right Trigger	Throttle. Positive forward.
Throttle2	axis [-1,1]			Throttle 2.
Throttle3	axis [-1,1]			Throttle 3.
Throttle4	axis [-1,1]			Throttle 4.
BowThruster	axis [-1,1]	Q/E	Left Stick - Left/	Bow thruster input.
SternThruster	axis [-1,1]	Z/C		Stern thruster input.
SubmarineDepth	axis [-1,1]	K/I		
EngineStartStop	Button	E		
Anchor	Button	T		

Scene Input Provider Bindings

Name	Type	Keyboard Defaults	Gamepad Defaults	Description
ChangeCamera	button	C	Start	Changes camera.
CameraRotation	2D axis	Mouse Delta	Right Stick	Controls camera rotation.
CameraPanning	2D axis	Mouse Delta	Right Stick	Controls camera panning.
CameraRotationModifier	button	Mouse - LMB	Right Stick Press	Enables camera rotation.
CameraPanningModifier	button	Mouse - RMB	Left Stick Press	Enables camera panning.
CameraZoom	axis	Mouse - Scroll	D-Pad Up/Down	Camera zoom in/out.
ChangeVehicle	button	V	Select	Change vehicle or enter/exit vehicle.
FPSMovement	2D axis	WASD	Left Stick	Demo FPS controller movement.
ToggleGUI	button	Tab		Toggles demo scene GUI.

Input Manager (old/classic)

- Type of InputProvider for handling user input on desktop devices through keyboard and mouse or gamepad.
- Uses classic/old Unity Input Manager. It is recommended to use the Unity's new Input System instead for new projects.

InputSystem package is required even if not used. If using the old/classic Unity input set Project Settings ⇒ Player ⇒ Input Handling to Both and proceed as normal. InputSystem package being present installed will not interfere with old/classic Unity input / InputManager.

Installation

When first importing Dynamic Water Physics 2 the project will be missing required bindings. There are two ways to add those:

1. Manually adding each entry to the *Project Settings* ⇒ *Input* following the input bindings table.
2. Copying the contents of *InputBindings.txt* and appending them to the contents of the `[UnityProjectPath]/ProjectSettings/InputManager.asset` file. To do so:
 - Close Unity.
 - Open *InputManager.asset* in Notepad/Notepad++/Visual Studio or any other text editor of your choice.
 - Copy the contents of the provided *InputBindings.txt* file (*Scripts* ⇒ *ShipController* ⇒ *Input* ⇒ *InputProviders* ⇒ *InputManagerProvider* ⇒ *InputBindings.txt*) and paste them at the end of the *InputManager.asset*. Make sure there are no empty lines between the existing content and the pasted content. Also make sure that all the indents are correct (Unity will detect end of file if indent is off). Save the file.
 - Open Unity. Check *Project Settings* ⇒ *Input*. The input bindings for Dynamic Water Physics will appear towards the bottom of the list.

Scene Setup

To set up InputManager-based input in the scene add the following components to the scene:

1. 'InputManagerShipInputProvider'
2. 'InputManagerSceneInputProvider'

Any ship that is present in the scene will now receive input from these providers.

Input System (new)

- InputSystem v1.0 or higher is required. This is available in Unity 2019.3 or newer.

When using DS4Windows, InputSystem will detect button presses twice.

Installation

- Install 'Input System' package through Window ⇒ Package Manager
- Under Edit ⇒ Project Settings ⇒ Player ⇒ Other Settings ⇒ Active Input Handling select Input System Package (New) or Both - the latter in case your project still uses `UnityEngine.Input` somewhere.

Scene Setup

- Add `InputSystemShipInputProvider` and `InputSystemSceneInputProvider` to any object in your scene.
- Default bindings can be modified by double clicking on `.inputactions` files. Save Asset

must be clicked for the changes to take effect.

Mobile Input Provider

- Add `MobileShipInputProvider` and `MobileSceneInputProvider` to the scene.
- Create a few UI ⇒ `Button` objects inside your canvas. Make sure that they are clickable.
- Remove the `UnityEngine.UI.Button` component and replace it with `MobileInputButton`. `MobileInputButton` inherits from `UnityEngine.UI.Button` and adds `hasBeenClicked` and `isPressed` fields which are required for `Mobile Input Provider`
- Drag the buttons to the corresponding fields in the `MobileShipInputProvider` and `MobileSceneInputProvider` inspectors. Empty fields will be ignored.

Scripting

Retrieving Input

Since v1.0 multiple `InputProviders` can be present in the scene, meaning that their input has to be combined to get the final input result. To get the combined input use:

```
float throttle = InputProvider.CombinedInput(i => i.Throttle());  
bool engineStartStop = InputProvider.CombinedInput(i =>  
i.EngineStartStop());
```

Or to get the input from individual `InputProviders` (say to find out if a button was pressed on a keyboard): `float throttle = InputProvider.Instances[0].Throttle;` When using input generated by code (i.e. AI) it is usually handy to have access to a single axis throttle/brake. This can be done like so:

```
shipController.input.Throttle = 0.5f;  
shipController.input.Throttle = -0.5f;
```

`shipController.input.states.throttle` is equal to `shipController.input.Throttle`. The latter is just a getter/setter for convenience.

Manually Setting Input

Input in each ship is stored in `InputStates` struct:

```
myShipController.input.states
```

In case input should not be retrieved from user but from another script - as is the case when AI is used - `AutoSettable` should be set to `false`. This will disable automatic input fetching from the active `InputProvider`.

Input now can be set from any script:

```
myShipController.input.Horizontal = myFloatValue; // Using getter/setter.
```

```
myShipController.input.states.horizontal = myFloatValue; // Directly  
accessing states.
```

Custom InputProvider

If a custom InputProvider is needed it can easily be written. Custom InputProviders allow for new input methods or for modifying the existing ones. E.g. if the MobileInputProvider does not fit the needs of the project a copy of it can be made and modifications done on that copy. That way it will not get overwritten when the asset is updated.

Steps to create a new InputProvider:

- Create a new class, e.g. ExampleInputProvider and make it inherit from InputProvider class:

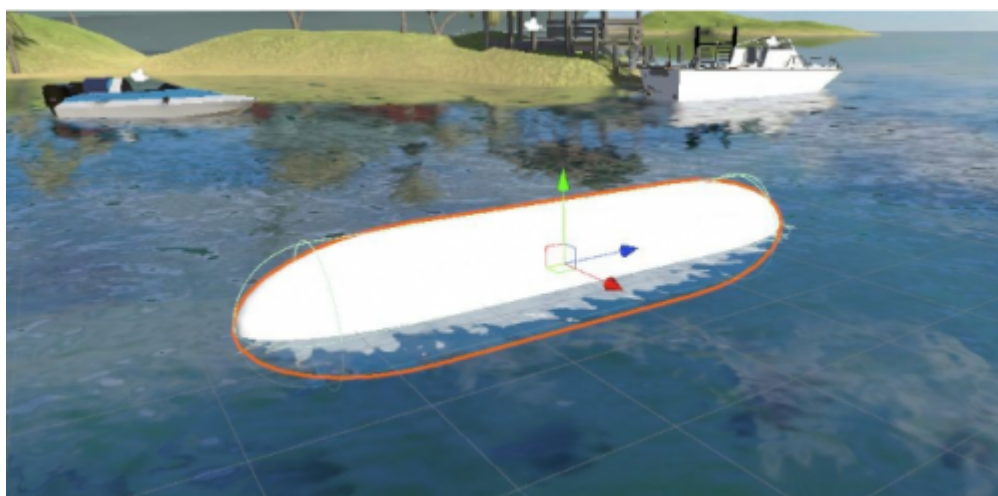
```
public class ExampleInputProvider : InputProvider {}
```


- Implement missing methods. Most IDEs can do this automatically.
- The required methods are *abstract* and will need to be implemented. There are also *virtual* methods such as ToggleGUI() which are optional and will be ignored if not implemented.
- Methods that are not used should return false, 0 or -999 in case of ShiftInto() method.

2020/07/17 12:24 · Aron Rescec

Quick Start

For the example setup a primitive capsule will be used (GameObject menu > 3D Object > Capsule).



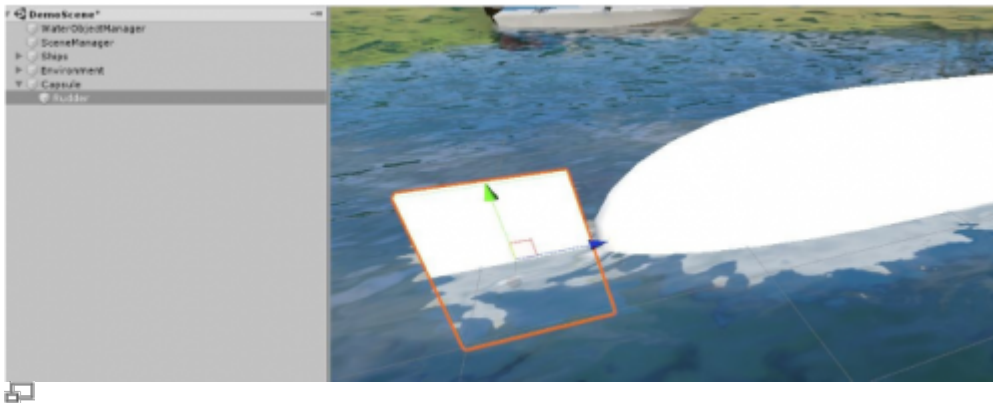
 Capsule representing a ship.

1. Add the ship object into the scene and tag it Ship (add a new tag if it does not exist). The tag is only necessary so that the ship changer can find your ship.
2. Add WaterObjectWizard component to the ship object. Tick the particle system option and click on Auto-Setup button. Click play - the object will float now. For manual WaterObject

setup follow the guide [here](#).

3. Add `AdvancedShipController` component to the parent object (the one containing the `Rigidbody`).
4. Add `CenterOfMass` component to the parent object and adjust center of mass to be near the bottom of the ship (green sphere). If this is not done the ship will most likely tilt to the side since the center of mass in Unity is calculated as a center of volume of all of the `Rigidbody` colliders, which is unrealistic for ships which generally have the center of mass near the keel.
5. Make sure that the input has been set up as per Input guide above.

Rudders



As an example a simple primitive cube will be used.

1. Add a rudder (in this case just a scaled cube) to the ship.
2. Assign the rudder transform (in this example the scaled cube) to the `Rudder Transform` field under *Rudders* foldout.
3. Add `WaterObject` component to the rudder so it too can interact with water.
4. Add a `Camera` of any type to the ship object (as a child) and tag it `ShipCamera`.
5. Press play and cycle to your ship using the `V` button (default change ship button). The boat is now floating and the rudders turn. If the rudders turn around wrong axis the rotation of the model needs to be fixed. Check this [link](#) for a guide on how to fix the model rotation.

Engines

Engines

Size

▼ Left Outboard

Name

Is On ☐

Input Mapping

Engine

Min RPM

Max RPM

Max Thrust

Spin Up Time

Starting Rpm

Start Duration

Stop Duration

Propeller

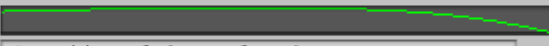
Thrust Position X Y Z

Thrust Direction X Y Z

Apply Thrust When Above Water ☐

Reverse Thrust Coefficient

Max Speed

Thrust Curve 

Rudder Transform

Animation

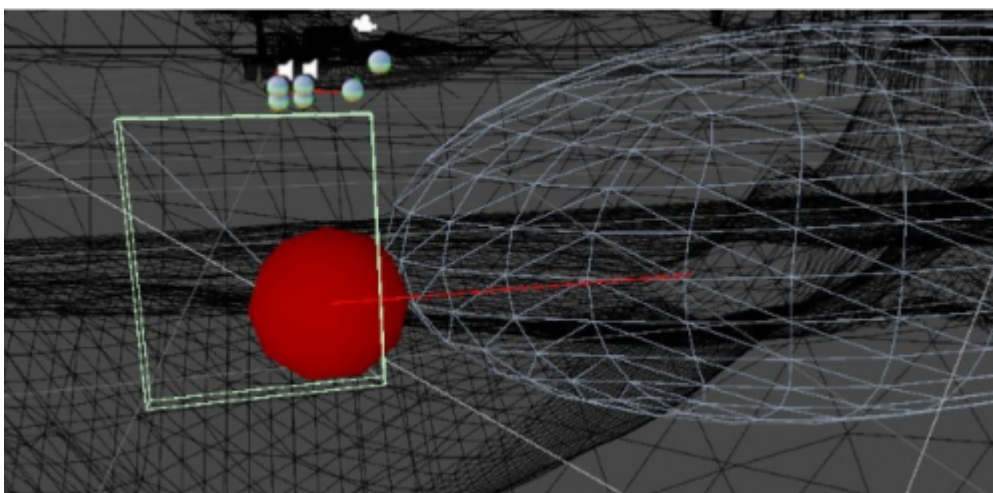
Propeller Transform

Propeller Rpm Ratio

Rotation Direction

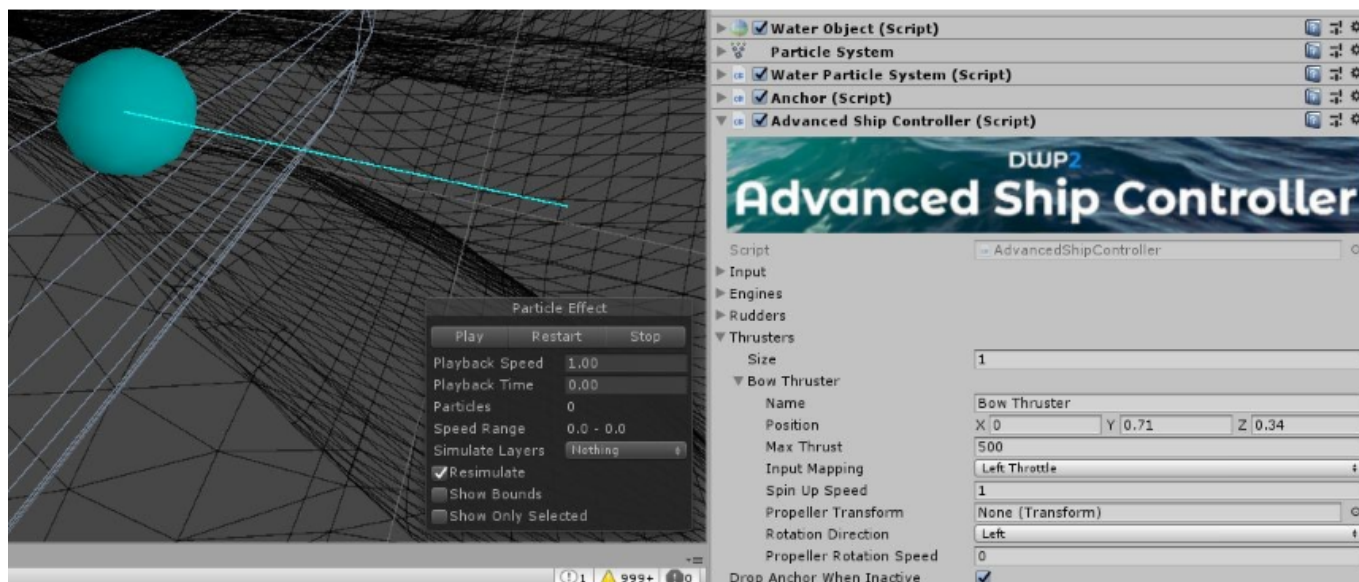
Engine section of AdvancedShipController.

1. To make ship move an engine and propeller are needed. Ship Controller assumes that there is one propeller per engine. Add an engine under Engines tab and set the wanted values (hover over each value to see what it does). If thrust position is above the water thrust will not be applied (if Apply Thrust When Above Water is left unchecked).
2. Default thrust position is at [0,0,0]. Make sure to adjust this value to fit your ship.



An example engine setup with thrust point shown as red sphere and thrust direction as a red line.

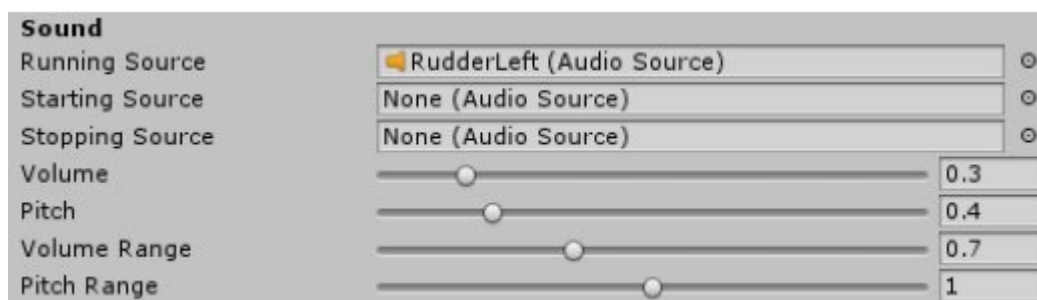
Thrusters



An example thruster with thrust point shown as a sphere and thrust direction as a blue line.

Larger ships usually have bow and/or stern thrusters to help them maneuver. Thruster is indicated as a blue sphere with a line indicating direction of thrust when positive input is pressed.

Sound



Sound section of AdvancedShipController

Engine sound is achieved through simple pitch modulation. To set up sound:

- Add an AudioSource to the ship object and assign a looped engine sound clip to it. Drag the AudioSource to the field Running Source under Engine foldout.
- Same should be done for Starting Source and Stopping Source fields. These fields are options. Adjust the Start Duration and Stop Duration fields to be somewhat shorter than the length of starting and stopping clips.

Ship Controller

Engine

Left

Left

Is On

Engine

Min RPM

300

Max RPM

800

Max Thrust

170000

N

Spin Up Time

2

N

Starting RPM

300

Start Duration

1.3

s

Stop Duration

0.8

s

Propeller

Thrust Position

X -1.21

Y -6.5

Z -21

Thrust Direction

X 0

Y 0

Z 1

Apply Thrust When Above Water

Reverse Thrust Coefficient

0.3

Max Speed

20

m/s

Thrust Curve

Rudder Transform

None (Transform)

Animation

Propeller Transform

None (Transform)

Propeller Rpm Ratio

0

Sound

Volume

0.3

Volume Range

0.7

Pitch

0.4

Pitch Range

0.3

Engine inspector.

Engine represent a single inboard or outboard engine. It handles power delivery, propeller rotation and engine sound.

Each ship can have multiple engines.

By default the thrust position of the engine is at [0,0,0]. Be sure to adjust this value to fit your ship.

Fields

- **IsOn** - is the engine currently on.
- **MinRPM, MaxRPM** - minimum and maximum RPM the engine can achieve. Stalling is not supported.
- **Max Thrust** - maximum thrust that the engine can generate at MaxRPM.
- **Spin Up Time** - time needed for the engine to reach MaxRPM.
- **Starting RPM** - RPM at which the engine will run while starting.
- **Start Duration, Stop Duration** - duration of engine starting and stopping. Influences for how long the start sound will be played.
- **Thrust Position** - position at which the thrust force will be applied. Local coordinates.
- **Thrust Direction** - direction in which the thrust will be applied. Local coordinates.
- **Apply Thrust When Above Water** - set to true if you want the propeller to work even when

out of water. False by default.

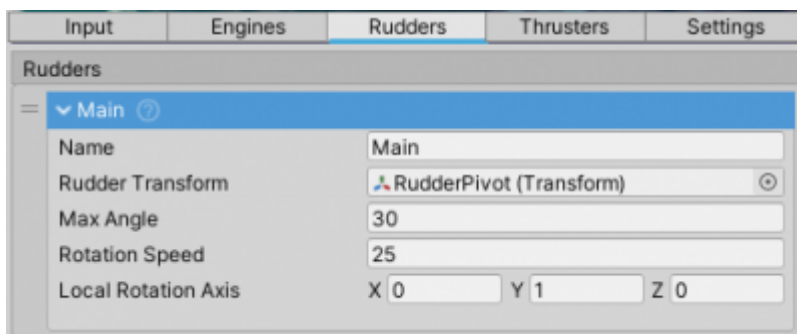
- **Reverse Thrust Coefficient** - thrust coefficient when throttle is negative.
- **Max Speed** - maximum speed a propeller can achieve.
- **Thrust Curve** - thrust percentage (of Max Thrust is represented on the Y axis) and speed is represented on the X axis as a percentage (0,1) of Max Speed). This is essentially a speed/efficiency curve of a propeller.
- **Rudder Transform** - a transform representing the rudder. Can be used when the engine is outboard and the thrust should be applied in the direction of the rudder.
- **Propeller Transform** - a transform which will represent the propeller. Visual only, does not interact with water.
- **Propeller Rpm Ratio** - ratio between engine RPM and propeller RPM.

Setup

1. To make ship move an engine and propeller are needed. Ship Controller assumes that there is one propeller per engine. Add an engine under Engines tab and set the wanted values (hover over each value to see what it does). If thrust position is above the water thrust will not be applied (if Apply Thrust When Above Water is left unchecked).

2020/07/17 13:02 · Aron Rescec

Rudder



Rudder inspector.

Rudder is used for steering the ship.

- Each rudder is a WaterObject and controls the ship through regular interaction with water.
- Invisible rudder can be used in case the visual one is too small. To achieve this parent a very thin Cube to the rudder, use WaterObjectWizard to set it up, resize it to the desirable scale and finally disable MeshFilter. This will result in a rudder that interacts with water but is not visible.

Fields

- **RudderTransform** - the transform that will be rotated.
- **MaxAngle** - maximum angle of rudder rotation to each side (e.g. +/- 30).
- **RotationSpeed** - rotation speed in deg/s for the rudder.
- **LocalRotationAxis** - the local axis around which the RudderTransform will be rotated.

Setup

1. Add a rudder to the ship if it does not already have one. This can be a primitive Cube scaled to a thin shape.
2. Assign the rudder transform (in this example the scaled cube) to the Rudder Transform field.
3. Add WaterObject component to the rudder so it too can interact with water.

2020/07/17 13:06 · Aron Rescec

Thruster

Input	Engines	Rudders	Thrusters	Settings
Thrusters				
= Bow ?				
Name	Bow			
Position	X 0	Y -1.22	Z 2.58	
Max Thrust	500			
Spin Up Speed	2			
Propeller Transform	None (Transform) ⓘ			
Thruster Position	Bow Thruster ▼			

Thruster inspector.

Thrusters can be used to move a ship without using the main engine(s). They can either apply thrust to the port or starboard side of the ship.

Fields

- **Position** - position at which the thrust will be applied.
- **Max Thrust** - maximum thrust in [N] which can be applied.
- **Spin Up Speed** - reaction time needed to reach Max Thrust
- **Thruster Position** - is it a bow or stern thruster. Determines input mapping.
- **Propeller Transform** - transform of the propeller that will be used as a visual representation of the thruster. Has to have Unity-correct rotation and pivot.
- **Propeller Rotation Direction** - rotation direction of Propeller Transform.
- **Propeller Rotation Speed** - rotation speed of the Propeller Transform.

2020/07/17 13:13 · Aron Rescec

Settings

Input	Engines	Rudders	Thrusters	Settings
Settings				
Drop Anchor When Inactive		<input checked="" type="checkbox"/>		
Weigh Anchor When Active		<input checked="" type="checkbox"/>		
Stabilization				
Stabilize Roll		<input checked="" type="checkbox"/>		
Roll Stabilization Max Torque		<input type="text" value="3000"/>		
Stabilize Pitch		<input type="checkbox"/>		

Settings inspector.

These are general ship settings.

Fields

- Drop Anchor When Inactive - if there is an Anchor component attached to this ship the Anchor will be dropped on sleep.
- Weigh Anchor When Inactive - if there is an Anchor component attached to this ship the Anchor will be weighed (raised) when ship wakes up.
- Stabilize Roll - if true a torque will be applied to the ship to try and negate any roll.
- Roll Stabilization Max Torque - maximum torque that will be used for roll stabilization.
- Stabilize Pitch - if true a torque will be applied to the ship to try and negate any pitch.
- Pitch Stabilization Max Torque - maximum torque that will be used for pitch stabilization.

2020/07/17 13:17 · Aron Rescec

Helper Scripts

Anchor

Anchor (Script)	
Anchor ?	
Drop On Start	<input type="checkbox"/>
Force Coefficient	<input type="text" value="10"/>
Zero Force Radius	<input type="text" value="1.5"/>
Drag Force	<input type="text" value="5000"/>
Local Anchor Point	X <input type="text" value="0"/> Y <input type="text" value="0.18"/> Z <input type="text" value="2.82"/>

Anchor inspector.

Anchor is a simple script that keeps a Rigidbody anchored to a point where the anchor was dropped.

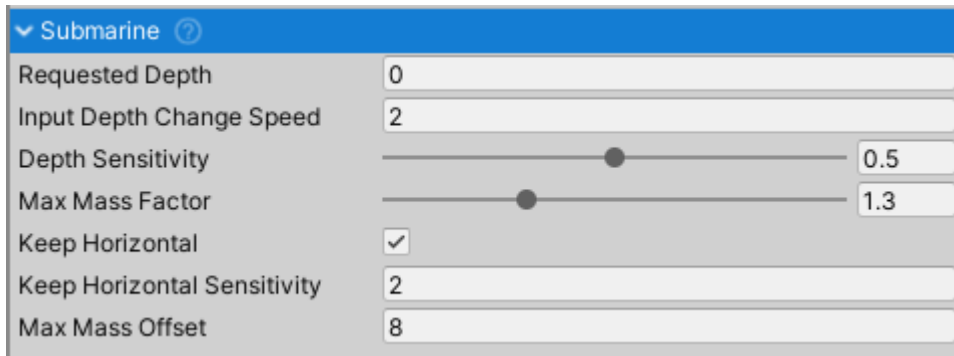
Fields

- Drop On Start - should the anchor be dropped when the scene starts?
- Force Coefficient - coefficient in regards to Rigidbody mass used to calculate the anchor force.

- Zero Force Radius - radius from the drop point in which no force is applied due to slack anchor line.
- Drag Force - force above which the anchor point will drag/move.
- Local Anchor Point - point on ship at which the anchor force is applied. Local coordinates.

2020/07/17 13:33 · Aron Rescec

Submarine



Submarine inspector.

A script that can be attached to AdvancedShipController to control submarine-specific functions.

Fields

- Requested Depth - depth in meters at which the submarine will try to stay.
- Input Depth Change Speed - speed of change of Requested Depth when there is user input.
- Depth Sensitivity - how sensitive depth regulator will be to the changes in depth. Higher value will result in faster reaction times to depth change.
- Max Mass Factor - maximum mass multiplier. Submarines take on water to change depth and this value determines maximum total mass that a submarine can achieve.
- Keep Horizontal - if true the script will try to keep the submarine horizontal / level.
- Keep Horizontal Sensitivity - how fast the script will react to submarine not being level.
- Max Mass Offset - to keep the submarine level the center of mass is offset from the resting center of mass as required. This is the maximum offset that can be achieved.

2020/07/17 13:41 · Aron Rescec

From:

<http://dynamicwaterphysics.com/> - **Documentation for Unity**

Permanent link:

<http://dynamicwaterphysics.com/doku.php/ShipControllerManual>

Last update: **2021/10/25 08:58**

