

UnityFS version 1.0

Component Guide

UnityFS Component Reference

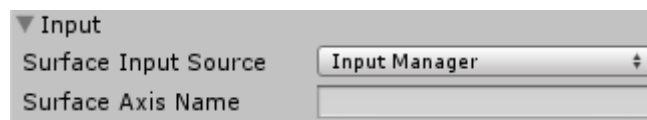
UnityFS version 1.0.....	1
Component Guide.....	1
Input.....	3
Cameras.....	4
CockpitCam.....	5
OrbitCam.....	5
TowerCam.....	6
Cockpit.....	6
AttitudeIndicator.....	7
NeedleInstrument.....	7
RudderPedal.....	8
ThrottleStick.....	8
Yoke.....	9
Dynamics.....	10
Aerofoil.....	11
CenterOfGravity.....	11
ControlSurface.....	12
Engine.....	13
GroundEffect.....	14
PropWash.....	15
Wing.....	16
External.....	18
StallWarner.....	19
SteerableNosewheel.....	19
WheelAudio.....	20
WheelBrake.....	20
WindAudio.....	21
Helpers.....	22
MirrorWing.....	23

Input

UnityFS 1.03 features a brand new input system. Any UnityFS component requiring input now uses this functionality.

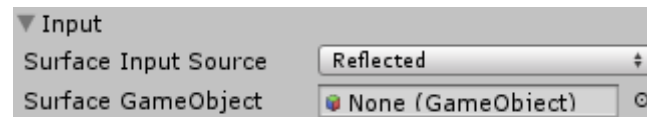
There are three input modes which can be used in UnityFS 1.03.

Input Source – Input Manager



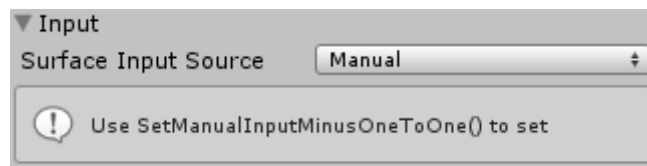
This is much like the previous version of UnityFS and allows you to drive any UnityFS controls directly from the Unity Input Manager. Simply type in your Axis Name as it is defined in the Input Manager and UnityFS will use that as its source of input.

Input Source – Reflected



This input mode allows you to drive any UnityFS control from any public float, Vector2, or Vector3 in your game. To use specify the game object which contains the value you wish to link to. Once selected you can then select the component which you wish to use. Finally you can then select any public float, Vector2 or Vector3 field to use for input. When using Vector2 or Vector3 you will then need to additionally specify which axis to link to.

Input Source – Manual



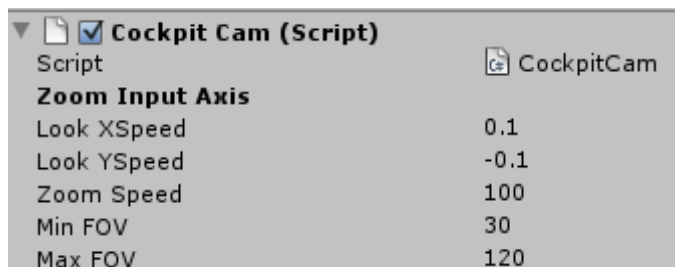
This final input mode allows you to manually set the input on any UnityFS control. To do this get the component for the object you wish to control. Get its InputController and call SetManualInputMinusOneToOne(). For example:

```
MyControlSurface.Controller.SetManualInputMinusOneToOne(0.5f);
```

Cameras

CockpitCam

Simulates a virtual cockpit style camera.



Requires

Camera

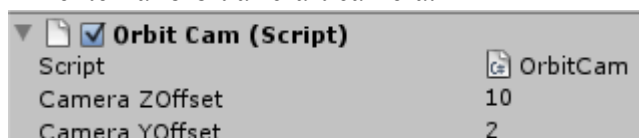
Usage

To use add to an empty GameObject and parent to the cameras folder in your aircraft hierarchy. Position as desired. Camera will be automatically picked up from the Aircraft component script.

Cameras can be switched using the ChangeCameraInputButton input axis defined in the Aircraft component window.

OrbitCam

An external orbit aircraft camera.



Requires

Camera

Usage

To use add to an empty GameObject and parent to the cameras folder in your aircraft hierarchy. This camera can not be positioned via the editor as it is positioned programatically. To position set the Z and Y offsets in the component editor window. Camera will be automatically picked up from the Aircraft component script.

Cameras can be switched using the ChangeCameraInputButton input axis defined in the Aircraft component window.

TowerCam

A fixed position camera which will always look at the aircraft it is attached to.

Requires

Camera

Usage

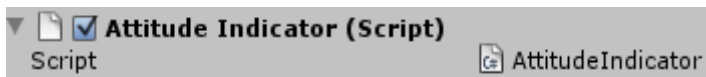
To use add to an empty GameObject and parent to the cameras folder in your aircraft hierarchy. Position as desired. The camera will not move from this position even if the parent objects move – I.e when the aircraft flies around the scene.

Cameras can be switched using the ChangeCameraInputButton input axis defined in the Aircraft component window.

Cockpit

AttitudeIndicator

A simple attitude indicator



Requires

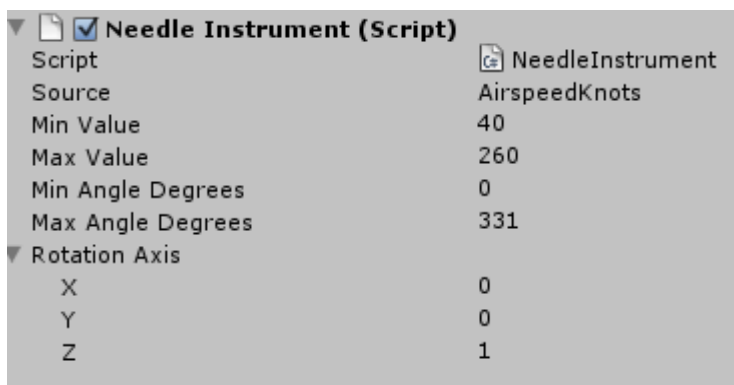
Nothing.

Usage

To use simply attach to the attitude indicator mesh that you wish to use. There are no parameters for configuration – orientation will be calculated automatically from the Aircraft's orientation.

NeedleInstrument

A needle instrument to drive virtual cockpit dials.



Requires

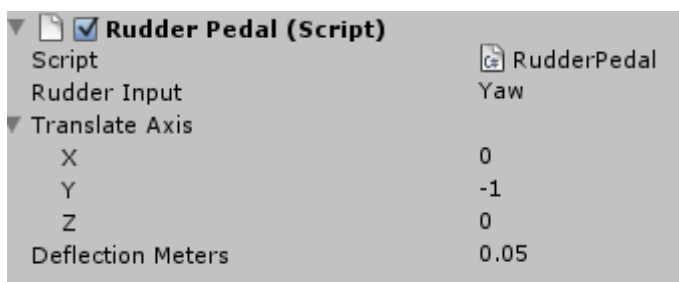
Nothing

Usage

To use attach to your needle mesh or a parent pivot about which the needle will rotate. Source for the needle can be selected from the Source enum in the component window. Min and Max values specify the minimum and maximum values that the instrument can read whilst Min and Max angle degrees specify the physical rotation of the instrument.

RudderPedal

A virtual cockpit rudder pedal.



Requires

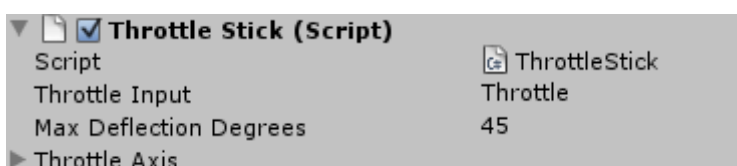
Nothing

Usage

To use attach to your rudder pedal mesh or a parent pivot about which the rudder pedal will translate. Rudder Input should be set with the keyboard or joystick input axis that is used to drive the ruder pedal. The -1 to 1 input is converted to a physical translation of the pedal which can be specified by setting the Deflection Meters parameter. (Assumes Rudder pedal is in the neutral position I.e 0 rudder input)

ThrottleStick

A virtual cockpit rotating throttle stick.



Requires

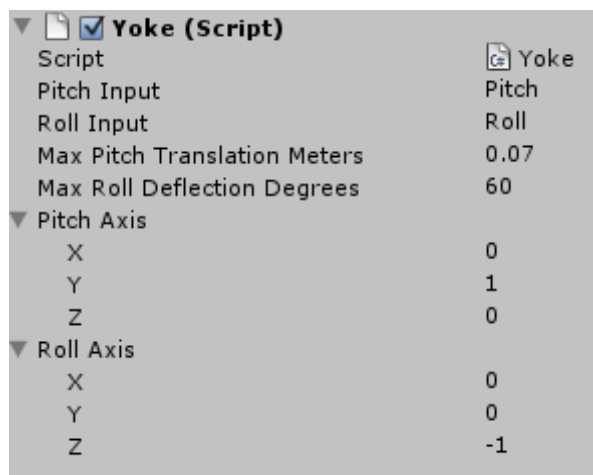
Nothing

Usage

To use attach to your throttle mesh or a parent about which the throttle stick will rotate. Throttle Input should be set with the keyboard or joystick input axis that is used to drive the Throttle. Max deflection degrees specifies the maximum rotation of the throttle mesh for 100% throttle. The throttle mesh is assumed to be in the 0% Throttle position by default.

Yoke

A virtual cockpit control yoke.



Requires

Nothing

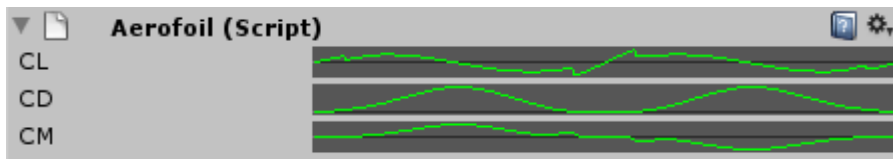
Usage

To use attach to your yoke pedal or a parent pivot about which the yoke will translate and rotate. Pitch Input should be set with the keyboard or joystick input axis that is used to drive the elevator/pitch. Roll input should be set with the keyboard or joystick input axis that is used to drive the ailerons/roll. Max Pitch Translation Meters and Max Roll Deflection Degrees specify the translation and rotation for the Yoke in relation to full pitch and roll input. The Yoke is assumed to be in the neutral input (0 pitch 0 roll) position by default.

Dynamics

Aerofoil

An aerofoil.



Requires

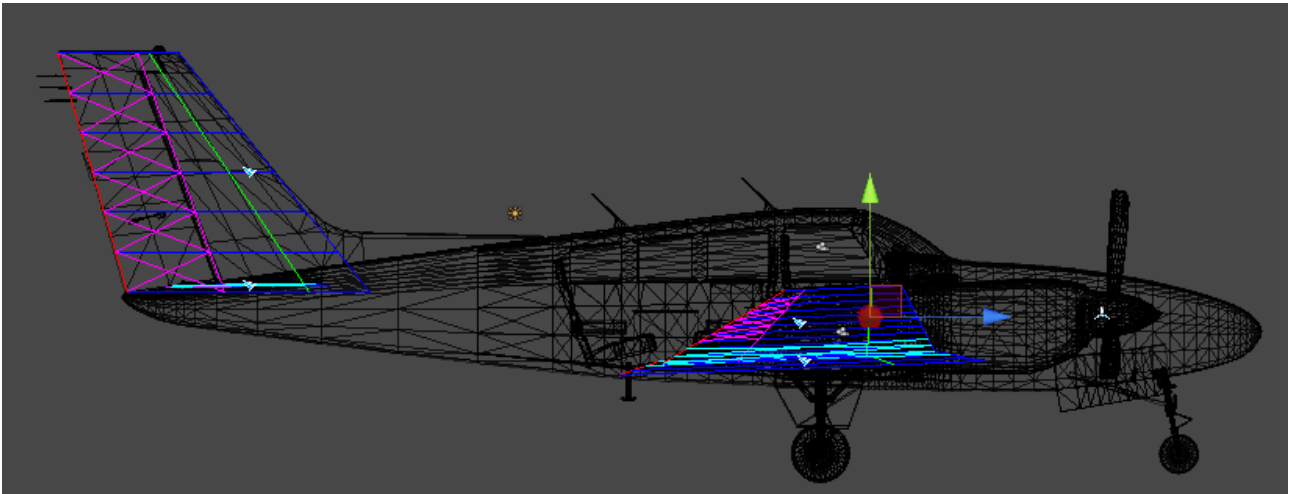
Nothing

Usage

Aerofoil specifies the lift drag and moment coefficients used by a particular wing. There is no need to have any instances of aerofoils in the scene. Instead use the existing aerofoil prefabs or create new prefabs for easy reuse. To use an aerofoil, drag and drop the aerofoil on to the desired Wings aerofoil field in the Wing Component.

CenterOfGravity

A helper object to override the default Rigidbody center of gravity.



Requires

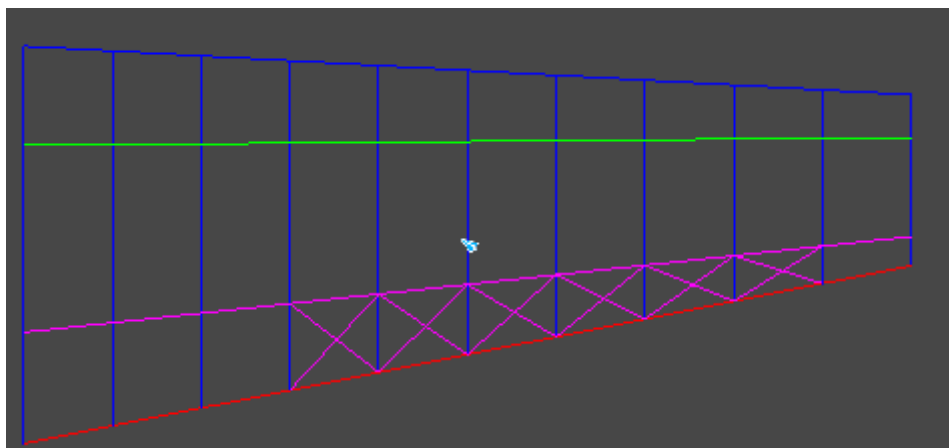
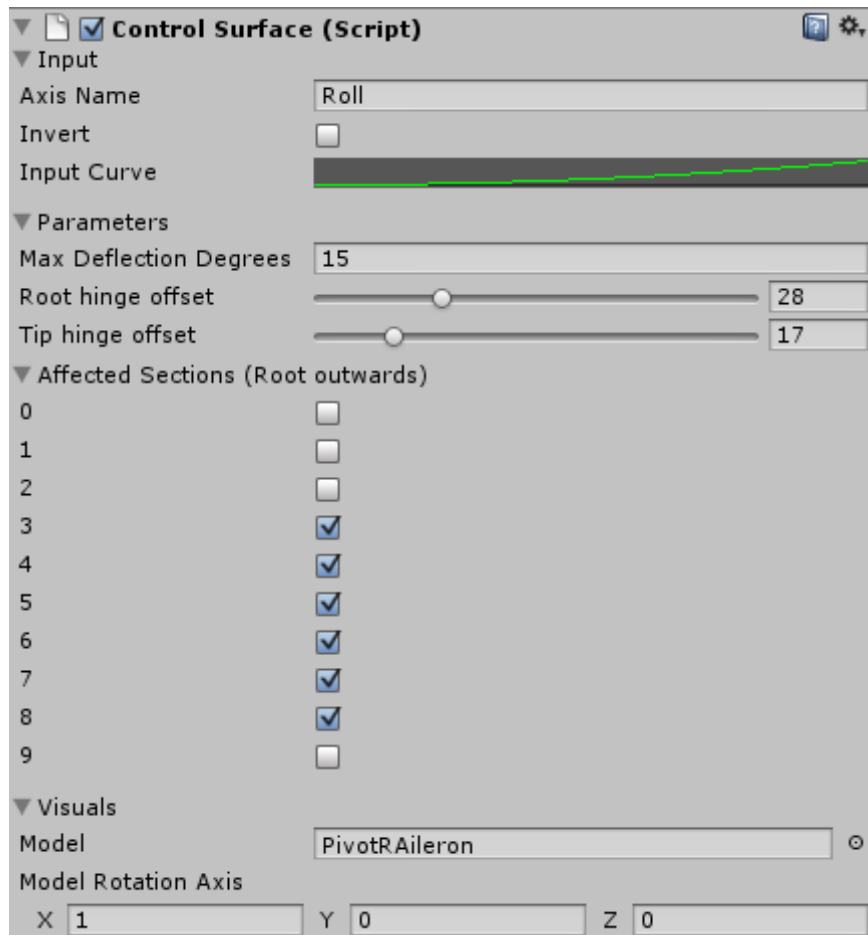
Nothing

Usage

To use add to an empty GameObject and parent to the Dynamics folder in your aircraft hierarchy
CG position is critical to aircraft longitudinal stability. As a rough guide CG should be positioned centrally in line with the wings lift line (25% back along the wing chord) as per screenshot above.

ControlSurface

An angle of attack control surface.



Requires

Wing





Usage

To use add to desired wing. Once attached you will notice in the Scene view that the wing now has a purple line running along it, this is the hinge line for the control surface. Root hinge offset and tip hinge offset can be used to set the size of the control surface. Affected Sections can be used to select the sections which will move, these are indicated in the scene view via a purple X. Max deflection degrees specifies the total control surface deflection available e.g 30 degrees = 15 degrees up and 15 degrees down. The rotation is driven by a control input specified in the Axis Name field. For non linear response the input curve can be used to remap from 0-1. For inverted input check the Invert box.

A visual mesh can be attached to the control surface for animation – simply select the desired mesh in the Model field and specify the axis of rotation. The model is assumed to be in the neutral / 0 input position by default.

Engine

A simple engine/thruster for version 1.0 – More advanced thrust models coming soon.

▼  Engine (Script)	 Engine
Script	PivotPropL (Transform)
Animated Propeller Pivot	
▼ Animated Propeller Pivot Rotate Axis	
X	0
Y	0
Z	-1
Slow Propeller	baron69_pivot
Fast Propeller	Prop_rotate_L_pivot
RPMTo Use Fast Prop	400
Idle RPM	400
Max RPM	2800
Force At Max RPM	4400
Percentage Force Applied VSAirspeed K	
RPMTo Add Per KTOF Speed	10
RPM Lerp Speed	1.5
Throttle Axis	Throttle
Engine Start Clip	 leftstart
Engine Run Clip	 leftengine
Pitch At Idle RPM	0.65
Pitch At Max RPM	1.1
Current Engine State	Running
Engine Start Input Button	Ignition

Requires

Nothing

Usage

To use add to an empty GameObject and parent to the dynamics folder in your aircraft hierarchy. Position as desired. Thrust can be set using the Force At Max RPM field, with the RPM range set via IdleRPM and MaxRPM. For a non linear thrust response the Percentage Force Applied VS Airspeed KTS curve can be tweaked. RPM Lerp Speed controls the speed at which the RPM spools up or down in relation to the input. Input is defined via the Throttle Axis field.

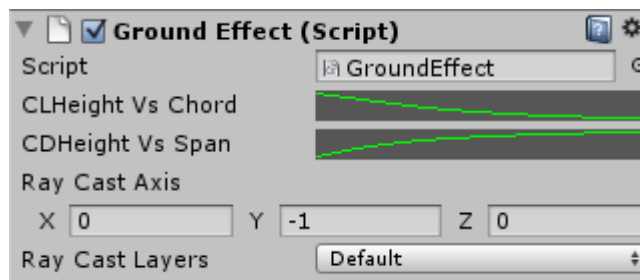
Engines can be turned on or off via a keypress. This key/button can be defined using the Engine Start Input Button field. The engine has three states – off, starting and on. The default state can be selected from the Current Engine State enum.

For propeller animation an object can be attached to spin in relation to RPM. This can be specified via the Animated propeller pivot field. For an advanced effect separate models can be turned on and off for differing RPM speeds. I.e a propeller mesh for 0 to slow RPM's and a blurred quad for faster RPM's these can be defined via the Slow Propeller and Fast Propeller fields. (This requires that the slow and fast meshes are parented to the Animated Propeller Pivot.

For audio both an engine start and engine run sound can be defined. The engine run sound pitch is adjusted based on rpm. This can be controlled via the Pitch At Idle RPM and Pitch at Max RPM fields. (Note for engine start, the start key has to be pressed and held until the start audio clip has finished)

GroundEffect

Ground effect simulation when close to the ground.



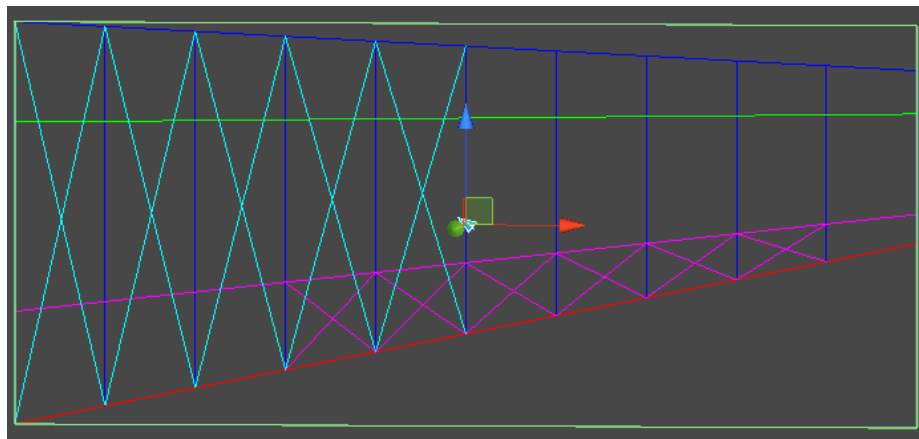
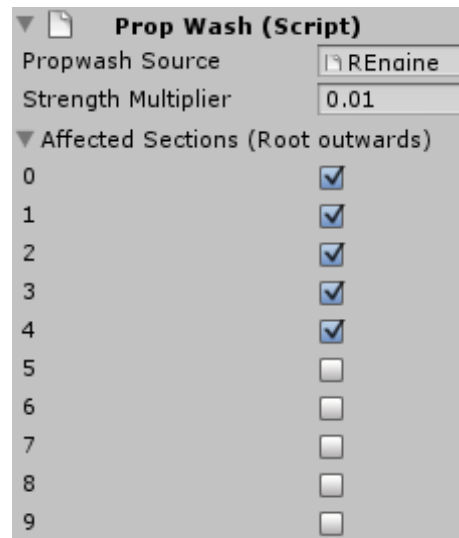
Requires

Wing

To use add ground effect components to the main wing shapes you wish to use to simulate ground effect. Ground effect both increases lift and decreases drag as you approach the ground. The strength of the ground effect can be controlled by adjusting the lift and drag curves on the component. For UnityFS to find the ground it must use raycasts. The direction of the raycast (in local space) can be specified using the RayCastAxis vector3 field. Raycast layers can be used to define what layers to test against. Note – for complex aircraft and arrangements it is important to check that the aircraft itself does not have any colliders which the raycast would hit. This can be checked by enabling debug drawing and looking for the red and green debug lines from the raycast result.

PropWash

A rough propeller wash simulation for version 1.0 – More advanced model coming soon.



Requires

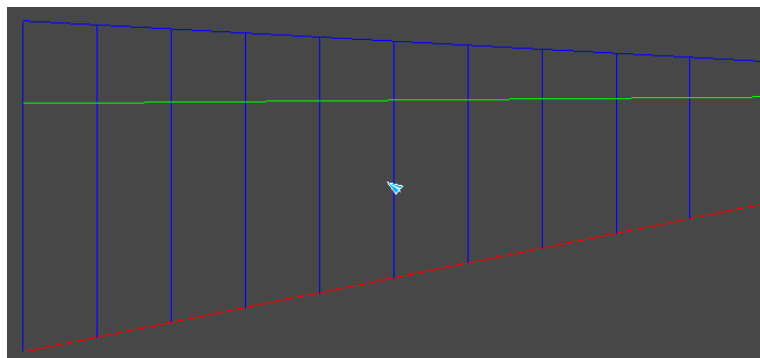
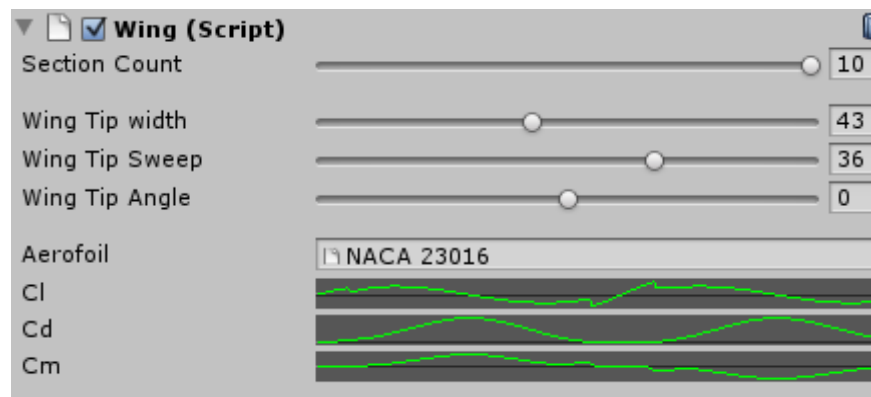
Wing

Usage

To use add to wing and select the Engine source of the prop wash in the Propwash Source field. The strength of the Propwash is equal to the Thrust multiplied by the Strength Multiplier field. It is likely you want this to be a very small value otherwise your wing will produce lots and lots of lift. Similar to control surfaces the affected wing sections can be selected by checking the sections boxes, these will be represented in the scene view by blue crosses.

Wing

A wing.



Requires

BoxCollider

Usage

To use add to an empty GameObject and parent to the dynamics folder in your aircraft hierarchy.

The top blue line in the above scene view is the leading edge of the wing – this should be facing forwards (+z). The Green line is the lift line (25% chord line) and the red line is the trailing edge.

You will note that the above screen shot features a right wing. For a left wing simply rotating the wing 180degrees will not work as the trailing edge will now be at the front of the wing. For left wings set the X axis scale to be negative (-) this will ensure that the trailing edge remains in the correct direction, or better still use the Mirror tool (see Helpers section).

The wing is divided into a number of sections, this is how the algorithm breaks down the wing and performs it's lift and drag calculations. More sections = greater accuracy but more CPU cost. The default and highest number of sections is 10. This can be lowered all the way to 1 for much quicker calculations at the expense of accuracy. (This is worth considering for mobile platforms.)

Wing tip width, sweep and angle can be set using the sliders, whilst scale can be defined by scaling the wing component (using the scale tool in the scene editor.) **Note wings should never be rotated around Y (that is to say the wings z axis should always be facing forwards) with regards to the**

aircraft. If the wing you are modelling is swept use the sweep slider – this way the aerofoil will always be facing the correct way. (Even if a wing appears to be sweeping drastically backward the aerofoil cross section shape is always facing forwards)

By default UnityFS will perform basic aerofoil approximation which will give users a chance to get their aircraft into the air quickly, but for advanced realism it is possible to use real aerofoil data to drive the wing physics. These can be selected by dragging an aerofoil prefab into the Aerofoil box. Unity FS ships with over 1000 example aerofoils. To find out specific aerofoils for specific aircraft wings the following website offers a wealth of information:

<http://www.aerofiles.com/airfoils.html>

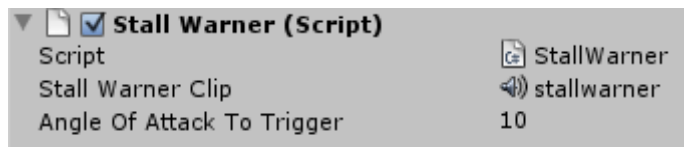
It is worth noting that most aerofoils are not symmetrical and also will not generate 0 lift and 0 degrees angle of attack this is often desirable for main wings, it is however not desired for the rudder and elevator as this would result in differing amounts of lift depending on which way the control surface is deflected. I therefore recommend that symmetrical aerofoils are used in these instances – again good reference material will give you specifics on all surface aerofoils for the aircraft you are modelling.

The wing component also automatically uses a Box Collider for collision. This can not be scaled as it is controlled via the script. Instead for scaling scale the whole GameObject using the scale tool in the scene view. If you do not wish to use collision simply turn off the Box Collider in the scene view.

External

StallWarner

An audible stall warner alarm.



Requires

Wing

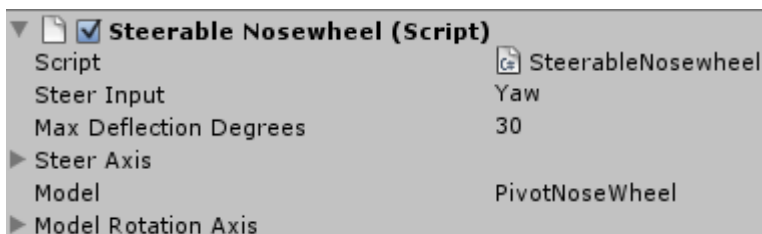
Usage

To use add to the wing you wish to have the stall warner attached . (Usually an inward section of the wing as wings often have washout which means that the inner section of the wing is at a higher angle of attack and stalls before the outer section, meaning you can still maintain aileron authority)

The angle at which the sound warner horns is set via the Angle Of Attack To Trigger field. Stall Warner Clip defines the sound to be used.

SteerableNosewheel

A steerable nosewheel control.



Requires

Nothing




Usage

To use add to the WheelCollider you wish to control. Steer Input can be used to define which input controls the steering – this is often the same as the Rudder. Max Deflection Degrees specifies the total nosewheel steering range. I.e 30degrees = 15 left and right.

As with control surfaces a model can also be driven by the SteerableNosewheel component. This can be defined in the Model field along with the Model Rotation Axis. (Note wheel colliders should be kept separate from wheel meshes so that the wheel meshes can be rotated separately.)

WheelAudio

Wheel rumble audio for when wheel is in contact with ground.

▼  <input checked="" type="checkbox"/> Wheel Audio (Script)	 WheelAudio
Script	 groundroll
Wheel Roll Clip	1000
RPMFor Max Volume	0.25
Max Volume	

Requires



WheelCollider

Usage

To use add to desired WheelCollider. Roll sound can be specified via the Wheel Roll Clip field. The volume of the wheel roll effect is dependent on wheel RPM. RPM can be set via the RPM For Max Volume field.

WheelBrake

A wheel brake to slow aircraft down when on ground.

▼  <input checked="" type="checkbox"/> Wheel Brake (Script)	 WheelBrake
Script	Brake
Wheel Brake Input Axis	10000
Brake Torque	

Requires

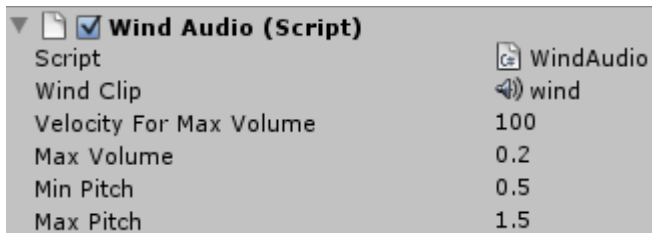
Wheel Collider

Usage

To use add to desired WheelCollider. Breaking torque can be set via the Brake Torque field with the input used to drive it set via Wheel Brake Input Axis.

WindAudio

Wind sound as aircraft travels through the air.



Requires

RigidBody

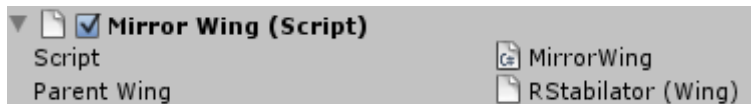
Usage

To use add to parent aircraft object (to which Aircraft and Rigidbody components are attached)
The wind sound effect can be specified by the Wind Clip field. Wind volume and pitch are controlled by the velocity. These can be specified by the Velocity For Max Volume, Max Volume, Min Pitch and Max Pitch fields.

Helpers

MirrorWing

A helper to mirror a Wing and all of its attached components.



Requires

Nothing

Usage

To use add to an empty GameObject and parent to the Dynamics folder in your aircraft hierarchy. From the component window then fill in the Parent Wing field with the wing which you wish to mirror. The GameObject will then be automatically renamed and all of the attached components and positions copied and mirrored.

Note the following parameters are not copied to allow some control over mirrored wings and will still need to be set manually..

Wing

- All are copied

Control Surface

- Invert not copied.
- Model not copied.
- ModelRotationAxis not copied.

Propwash

- All are copied

GroundEffect

- All are copied