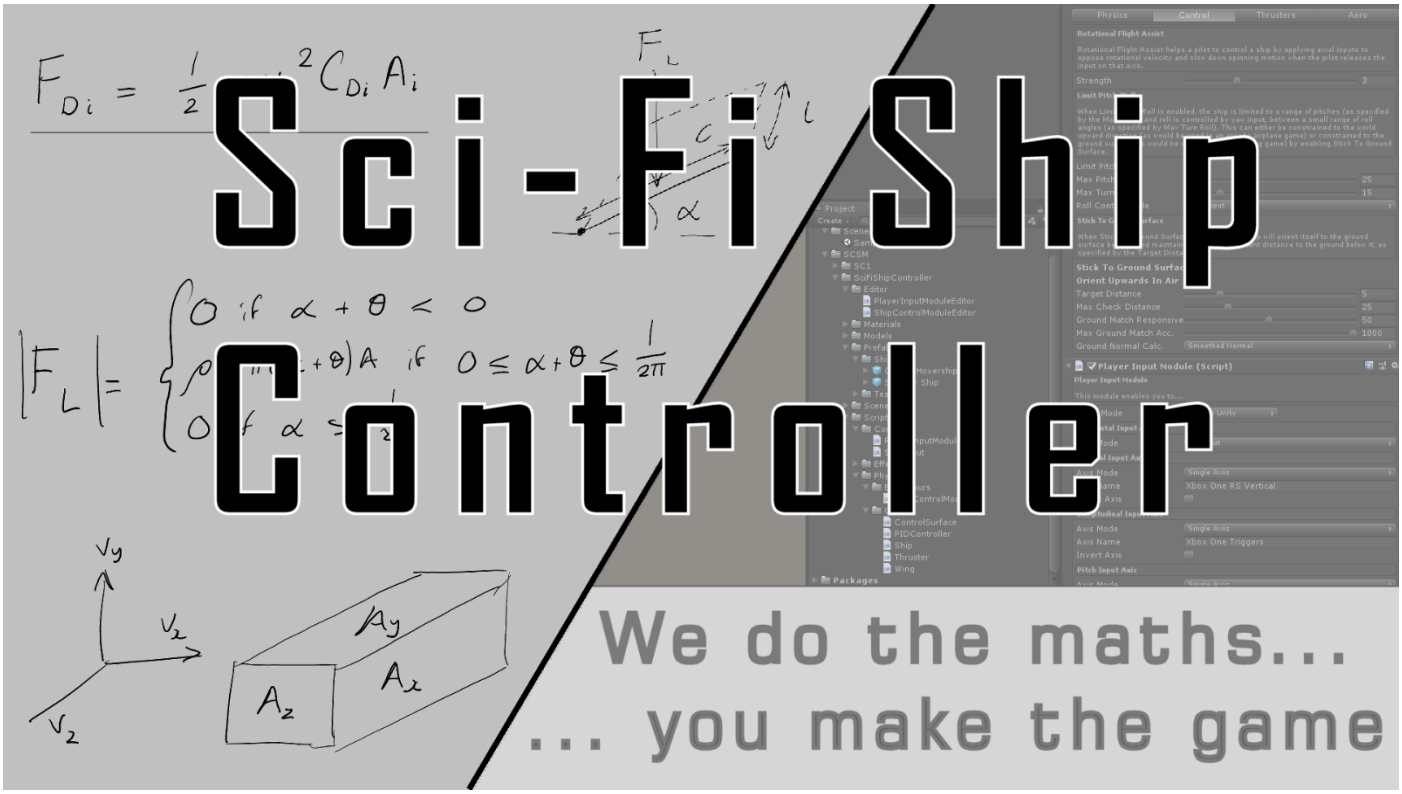


Popularized by the Star Wars saga, complex space planes which can move between planet surfaces and space with ease, are now the staple diet of gamers worldwide. The maths is extreme, which is why we take care of this for you.

NASA would call this a Space Plane controller. It can control a "craft" like an airplane while inside a planet's atmosphere, but can also act like a spaceship while in space.

What is Sci-Fi Ship Controller?

Sci-Fi Ship Controller is an asset that allows you to quickly and easily turn your ship models into fully-functioning, flying ships.

How do I use it in Unity?

Currently, it consists of two main modules: The Ship Control Module and the Player Input Module. The Ship Control Module is a script that can be added to ship models to turn them into flyable ships, complete with all the parameters needed to tweak their behaviour to your liking. The Player Input Module is a script that can be added to any ship with the Ship Control Module already attached to map inputs from the Unity input manager and other input sources to the Ship Control Module in order to let a player control the ship.

Why should I use Sci-Fi Ship Controller instead of another asset?

One of the main things we've worked really hard on with Sci-Fi Ship Controller is its ease of use. All the parameters in the modules are arranged logically and have headers describing their functionality. In addition to this, every editable parameter has an associated tooltip, so if you're unsure about what something does you can simply hover your mouse over it and get a brief description. We've also tried to write Sci-Fi Ship Controller in a way that makes sense to all game developers, not just ones that have a degree in aerodynamic engineering or physics.

One of the other strengths of Sci-Fi Ship Controller is its versatility. It isn't an asset JUST for arcade spaceships, JUST for aircraft or JUST for hover-ships: If it flies, then there's a good chance you can make it with our asset (and if you can't, feel free to let us know so that we have to opportunity to improve Sci-Fi Ship Controller in that regard).

As well as this, Sci-Fi Ship Controller is completely physics-based. All movements are driven by Unity's built-in physics, which provides a great feel for players and ensures you won't encounter any strange behaviour caused by our asset fighting with the physics engine.

Finally, Sci-Fi Ship Controller is designed from the ground up for performance. We've tried as much as possible to follow best practice and avoid expensive allocations and function calls, so Sci-Fi Ship Controller should only comprise a minimal part of your performance budget. The Combat system supports DOTS (Entities, Jobs, Burst Compiler in Unity 2019.1+).

What versions of Unity do we support?

We following a sliding window which roughly matches the versions supported by Unity. We currently support Unity 2017.2-4, 2018.x, 2019.1, and 2019.2.

Table of Contents

Getting Started

SSC comes bundled with a number of demo scenes and prefabs to get you started quickly. The demo scenes are set up with a single player ship, with keyboard input, to allow you to simply open the scenes in the Unity editor and hit play.

SSC has a lot of flexibility in dealing with many, many different scenarios. However, if you have a specific use-case in mind, the demo scenes may help you select a starting Ship Control Module configuration. Most people will want to start with one of the ship prefabs from the SCSM\SciFiShipController\Prefabs\Ships folder. Each of these prefabs has a basic ship model attached so you can quickly test it in your scene. You can add your own space craft models later.

After dropping one of the SSC ship prefabs into the scene, optionally drop in the PlayerCamera prefab (SCSM\SciFiShipController\Prefabs\Environment) and/or Celestrials (SCSM\SciFiShipController\Demos\Prefabs). Both of these require a small amount of configuration. You will see warnings in the Unity Console at runtime in the Unity Editor if you forget to configure them.

Throughout this manual we refer to anything that can be flown by a player or an AI-player as a “ship”. It could be any of the following:

- Aircraft / airplane which flies within a planet’s atmosphere or is affected by gravity
- Hover ship / craft or land-based speed racer
- Space ship that travels through empty space or around a planet
- Space plane (acts like an airplane while inside a planet's atmosphere, but acts like a spaceship while in space)
- Fast 1-2-seater fighter
- Large space battle cruiser which includes turret-like weapons

Videos and Tutorials

Name	URL
Initial Release Trailer	https://youtu.be/o_K_PTU5Vo4
Getting Started / Setup Basics Tutorial	https://youtu.be/-IvOhk_4P6k
Control Tab Basics Tutorial	https://youtu.be/8mNqLeun5eQ
Damage Basics Tutorial	https://youtu.be/6tfwtm9b1oU
Weapon Basics Tutorial	https://youtu.be/lIpttO-UUyE
Physics-Based Tutorial	https://youtu.be/EPbzfG_2Ltw
Anti-Gravity Racer Demo	https://youtu.be/reYNcztvt3I
SSC with Unity Mega City Tech Demo **	https://youtu.be/fIb8XhI_v5g

** Unity Mega City assets are not included with Sci-Fi Ship Controller. This Tech Demo is only to show that SSC can be integrated with other systems.

Demos

There are five demo scenes available currently, which can be found by navigating to the folder SCSM\SciFiShipController\Demos\Scenes. Each of them contains one ship which can be controlled by the player (just run the scene to start the demo). Apart from the "Endless Flier Demo", each demo uses the same basic control scheme:

- Up arrow key is forwards thrust, down arrow key is the brakes
- A/D keys are used to turn left and right
- W/S keys are used to pitch up and down
- Left and right arrow keys are used to roll clockwise and anticlockwise
- Additionally, in the "Explorer (Physics) Demo", Q/E keys are used to move up and down

Ship Control Module

Ship Control Module – Overview

This module enables you to implement ship behaviour on the object it is attached to. It contains the core configuration required to get your ship flying.

The module includes the following configuration categories. They appear in the logical order you'll typically set up a ship.

Category	Purpose
Physics	Where you select (realistic) Physics mode or Arcade mode. Also determines how gravity and mass affects a ship's behaviour.
Control	Includes options for over-riding the player's input behaviour.
Thrusters	Used to control the ship by adding forces to move the ship's position and torques to rotate the ship. Can also add Particle System effects and audio effects.
Aero	Determine how your ship interacts with the air around it (its aerodynamic properties).
Combat	Includes how the ship takes damage, how the ship is re-spawned, and how weapons are configured.

In-Scene Editing

Sci-Fi Ship Controller has a number of features that support in-scene editing for more intuitive development. This enables you to select different attributes of your ship and move them around within the scene view. The values in the SSC module editors will automatically be updated.

In the Ship Control Module editor, whenever you see a small "G" button it indicates that the Gizmos in the scene for that feature can be shown or hidden. Some items like Centre of Lift and Direction, and Centre of Thrust and Thrust Direction are non-selectable. Others like Centre of Mass, Wings, Control Surfaces, Thrusters, and Weapons, are selectable and can be manipulated in the scene view.

To edit the location, size, and/or rotation of an item associated with a ship perform the following actions:

1. In the scene Hierarchy, select the parent gameobject of the ship
2. In the Inspector, navigate to the appropriate tab in the Ship Control Module and make sure the Gizmos are enabled. If in doubt, click the "G" button in the editor to toggle them on/off.
3. In the scene view, move your mouse of the gizmo for the item and click it to select or unselect it.
4. Move, Rotate, or Resize the component with the standard Unity Editor tools. Note: Not all items have all 3 manipulation methods enabled. For example, it doesn't make sense to be able to rotate the ship's centre of mass however it does for a control surface or wing.

Physics Tab

The Physics tab is usually a good place to start when setting up a ship, as it contains a lot of the basics. Here you can specify the mass of the ship, as well as the strength and direction of gravity acting upon the ship. You can also specify whether Unity should set the centre of mass of the ship automatically or if you want to set it manually.

Probably the most important value to start with is the one at the top: The physics model. The physics model determines what options are available for ship control and behaviour. There are two options currently available: Physics-based and Arcade.

Physics-based mode is best employed for games aiming to achieve a large degree of realism, or at the very least evoke a sense of realism from players. In general, only physically realistic options are available. Ships can only be turned using thrusters and control surfaces.

Arcade mode, on the other hand, provides a number of extra options to enhance ship feel and gameplay while removing certain behaviours entirely in order to make ship setup and control easier. For example, pitch/roll/yaw

acceleration and turn acceleration options are only available in arcade mode. Pitch/roll/yaw acceleration values directly determine how quickly the ship rotates on each axis - in physics-based mode rotations can only be achieved indirectly via the use of thrusters. Turn acceleration adds force inputs to a ship based on its motion in order to make the ship's velocity more in line with the way it is facing, which is common in a lot of flight games as it makes flight control and movement a lot more intuitive for players.

Property	Description
Initialise On Awake	If enabled, the InitialiseShip() will be called as soon as Awake() runs for the ship. This should be disabled if you are instantiating the ship through code.
Physics Model	The physics model determines which options are available for ship control, as well as how physics for certain things such as thrusters is handled. Select Physics-Based mode to create physically realistic ships and aircraft, or select Arcade mode to replicate the behaviour of more fictitious craft.
Mass	The property of 'mass' determines how heavy the ship is (in kilograms). The point through which all of the mass of the ship seems to act is known as the 'centre of mass' (CoM) and is indicated by the grey sphere in the scene view. When 'Set CoM Manually' is disabled, the centre of mass will be automatically determined by Unity based on the position of any colliders attached to the rigidbody. Otherwise, the centre of mass can be adjusted by modifying the value of 'Centre of Mass' in the inspector or by selecting the grey sphere in the scene view and dragging it with the move tool. The 'Reset Centre Of Mass' button resets the centre of mass to the position automatically determined by Unity.
Set CoM Manually	When enabled, the centre of mass (the point through which all forces on the ship act) can be edited manually. When disabled, the centre of mass will be positioned at the default position specified by Unity - this is determined by the ship's colliders.
Centre of Mass	The position of the centre of mass in local space.
Gravity	Acceleration changes the strength of gravity. Increasing it increases the pull of gravity. Direction changes the direction in which gravity acts. This direction is indicated by the grey arrow in the scene view, and can be adjusted by selecting the grey sphere and using the rotation tool.
Acceleration	The acceleration due to gravity in metres per second squared. Earth gravity is approximately 9.81 m/s ^2.
Direction	The direction in which gravity acts on the ship in world space.
Pitch/Roll/Yaw Acceleration	Use Pitch/Roll/Yaw Acceleration to determine how quickly the ship can turn on each axis.
Roll Acceleration	How fast the ship accelerates when rolling left and right in degrees per second squared. Increasing this value will increase how fast the ship can be rolled left and right by pilot and rotational flight assist inputs.
Yaw Acceleration	How fast the ship accelerates when turning left and right in degrees per second squared. Increasing this value will increase how fast the ship can be turned left and right by pilot and rotational flight assist inputs.
Turn Acceleration	Use turn acceleration to add a more 'arcade' feel to ship movement. Flight Turn Acceleration adds force inputs to cause the ship to move in the direction it is facing while in the air, while Ground Turn Acceleration does the same while the ship is near the ground.
Flight Turn Acceleration	How quickly the ship accelerates in metres per second squared while in the air to move in the direction the ship is facing.
Ground Turn Acceleration	How quickly the ship accelerates in metres per second squared while near the ground to move in the direction the ship is facing.

Control Tab

For some ships, the control setup is very simple, while in others (such as the classic hover ship) the control setup is more complicated. Essentially, the Control tab is used for used for setting up any input control that the computer will do instead of the player. This includes input assists (such as the rotational flight assist) which make flight easier for players, as well as control modifiers that allow more interesting behaviour to occur (such as stick to ground surface, which controls the pitch, roll and vertical inputs of the ship to allow it to orient itself to the ground surface and maintain a given distance from it). The Control tab determines a lot of the behaviour of the ship related to gameplay.

Property	Description
Rotational Flight Assist	Rotational Flight Assist helps a pilot to control a ship by applying axial inputs to oppose rotational velocity and slow down spinning motion when the pilot releases the input on that axis.
(Rotational) Strength	Increasing this value will increase how quickly rotational flight assist slows down spinning motions. Setting it to zero will disable rotational flight assist entirely.
Translational Flight Assist	Translational Flight Assist helps a pilot to control a ship by applying translational inputs to oppose movement on non-forward axes when the pilot releases the input on those axes, which aligns the ship's velocity more with its forward direction, making turning easier and more intuitive.
(Translational) Strength	Increasing this value will increase how quickly translational flight assist slows down movement in a particular direction. Setting it to zero will disable translational flight assist entirely.
Limit Pitch/Roll	When Limit Pitch/Roll is enabled, the ship is limited to a range of pitches (as specified by the Max Pitch) and roll is controlled by yaw input, between a small range of roll angles (as specified by Max Turn Roll). This can either be constrained to the world upward direction (as would be used in an arcade airplane game) or constrained to the ground surface (as would be used in a hover-racing game) by enabling Stick To Ground Surface.
Max Pitch	The maximum pitch in degrees that the ship is allowed to attain.
Pitch Speed	How fast the ship pitches in degrees per second.
Max Turn Roll	The maximum roll in degrees that the ship is allowed to attain.
Turn Roll Speed	How fast the ship rolls in degrees per second.
Roll Control Mode	How roll is controlled. When Yaw Input is selected, roll is determined by the yaw input (turning left will make the ship roll left and vice versa). When Strafe Input is selected, roll is determined by the strafe input (strafing left will make the ship roll left and vice versa).
Responsiveness (Pitch/Roll)	How fast the ship pitches/rolls to match the target pitch/roll (for instance the ground surface if Stick To Ground Surface is enabled).
Stick To Ground Surface	When Stick To Ground Surface is enabled, the ship will orient itself and maintain a relatively constant distance to the ground below it, as specified by the Target Distance.
Orient Upwards In Air	Enable this to revert to the default behaviour of Limit Pitch/Roll (constraining the ship to a limited range of pitch and roll, facing upwards) when there isn't a detectable ground surface below the ship (i.e. when the ship is high up in the air).
Target Distance	The distance the ship will attempt to maintain to the ground surface below it.
Max Check Distance	The max distance the ship will check below it to find the ground surface. When the ship is further than this distance away from the ground surface it will either revert to the default behaviour of Limit Pitch/Roll (if Orient Up In Air is enabled) or it will revert to standard six-degrees-of-freedom behaviour (if Orient Up In Air is disabled).
Responsiveness (Target Distance)	How responsive ship is to sudden changes in the distance to the ground. Increasing this value will allow the ship to match the Target Distance more closely but may lead to a juddering effect if increased too much.

Property	Description
Max Acceleration (Target Distance)	The limit to how quickly the ship can accelerate to maintain the Target Distance to the ground. Increasing this value will allow the ship to match the Target Distance more closely but may look less natural.
Ground Normal Calc.	How the normal direction (orientation) is determined. When Single Normal is selected, the single normal of each face of the ground geometry is used. When Smoothed Normal is selected, the normals on each vertex of the face of the ground geometry are blended together to give a smoothed normal, which is used instead. Smoothed Normal is more computationally expensive.
Ground Layer Mask	Only geometry in the specified layers will be detected as being part of the ground surface.
Roll Power	How much power of the ship's roll thrusters is used to execute roll manoeuvres. Increasing this value will increase the roll speed and responsiveness of the ship.
Pitch Power	How much power of the ship's pitch thrusters is used to execute pitching manoeuvres. Increasing this value will increase the pitch speed and responsiveness of the ship.
Yaw Power	How much power of the ship's yaw thrusters is used to execute yaw manoeuvres. Increasing this value will increase the yaw speed and responsiveness of the ship.

Thrusters Tab

The Thrusters tab is exactly what it sounds like: It's where you set up your ship's thrusters. Each thruster has a vector direction and a force amount specified, and is linked to the various player inputs via the selection of which force direction and (in physics-based mode) moment directions (read: rotations) the thruster is able to move the ship in (this can be set up automatically using the auto-populate forces and moments button).

An effects object can also be specified for each thruster to link it with effects to be triggered when the thruster is in use.

As more thrust is applied (either from a human player via the Input Player Module, or in code via an AI-player), the Particle System effect increases. If an Audio Source is attached to the effect, the volume increases or decreased based on the amount of thrust that is being applied.

Property	Description
Name	The name of the thruster
Max Thrust (kN)	The max thrust in kilonewtons this thruster can generate.
Relative Position	The position of the thruster in local space.
Thrust Direction	The direction of thrust of the thruster in local space.
Thrust Input	Which input activates this thruster. This determines what input/s are linked with this thruster (if you don't know how to use this, the Auto-Populate Forces and Moments button can be used to calculate this automatically).
Primary Moment Input	The rotational input which is primarily used to activate this thruster. This determines what input/s are linked with this thruster (if you don't know how to use this, the Auto-Populate Forces and Moments button can be used to calculate this automatically).
Secondary Moment Input	The rotational input which is secondarily used to activate this thruster. This determines what input/s are linked with this thruster (if you don't know how to use this, the Auto-Populate Forces and Moments button can be used to calculate this automatically).
Damage Region	Which damage region this part belongs to.
Min Performance	The minimum possible performance level of this thruster (i.e. what performance level the thruster will have when its health reaches zero). The performance level affects how much thrust this thruster generates.

Property	Description
Starting Health	The initial health value of this part. This is the amount of damage that needs to be done to the damage region this part is associated with for the part to reach its min performance.
Effects Object	The object which has the effects that should be enabled when the thruster is used (e.g. sounds, particle effects, etc.) attached. Set the Volume of the AudioSource to represent the maximum volume at full thrust. This should typically be a child gameobject of the ship's parent gameobject.
Minimum Effects Rate	The 0.0 to 1.0 value that indicates the minimum normalised amount of any particle effects or audio sources that are applied when a non-zero thrust input is received for this thruster. The default is 0 which will apply a linear particle emission rate or audio volume based on the amount of thrust input received. If the full particle emission rate or audio volume should be applied when any input is received, set the value to 1.0.

Aero Tab

The Aero tab is where you determine how your ship interacts with the air around it (its aerodynamic properties). This can be through altering environmental properties (the medium density), the drag properties (how moving through the air slows the ship down on each axis) or by adding wings and control surfaces. Wings simulate the effect of parts moving through the air generating lift (upwards force); their properties are controlled by changing the size and angle of attack (inclination) of the wing, as well as the stall effect (how much the effect of stalling affects the wings of the ship). Control surfaces are moving parts that change the aerodynamic properties of the ship in order to let the pilot control it, such as ailerons, rudders and air brakes. Control surfaces are automatically linked to the correct player inputs based on the type of control surface and where it is positioned relative to the centre of mass of the ship.

The drag properties of the ship determine how the airflow around the ship affects the movement of the ship. After making any mesh changes, click the Calculate Drag Properties to recalculate the internally stored drag properties of the ship. Then use the Drag X/Y/Z Coefficients to alter how much drag the ship has on each axis. More streamlined axes of the ship should have a lower drag coefficient while flatter axes should have a higher drag coefficient. In Arcade mode, you can also use the Angular Drag Factor to alter how quickly angular drag will slow down any spinning motion and Disable Drag Moments to prevent the ship from rotating due to moments caused by drag.

Property	Description
Medium Density	The medium density property defines the density of the medium (in kilograms per cubic metre) the ship is travelling through (generally in air). In less dense atmospheres this value should be lower and in more dense atmospheres this value should be higher. At low altitudes in Earth's atmosphere the value is approximately 1.293 while in space (where there is virtually no air) it should be set to zero to achieve realism. The medium density affects all aerodynamics (i.e. drag, lift, etc.).
Drag X Coefficient	The coefficient of drag of the ship on the x-axis. Increasing the coefficient of the drag will increase the effect of drag.
Drag Y Coefficient	The coefficient of drag of the ship on the y-axis. Increasing the coefficient of the drag will increase the effect of drag.
Drag Z Coefficient	The coefficient of drag of the ship on the z-axis. Increasing the coefficient of the drag will increase the effect of drag.
Angular Drag Factor	How strong the effect of angular drag is on the ship. Setting this to 1 will make it physically realistic.
Disable Drag Moments	This will prevent drag causing the ship to rotate.

Wings

Property	Description
Wing Stall Effect	How much the effect of stalling affects ship flight. Setting this to zero will make the effect of stalling very minimal.
Name	Name of the wing. E.g. Front Wing, Tail Wing, Left Front Wing
Angle Of Attack	The angle of attack (inclination above the x-z plane) of the wing in degrees.
Length	The span (length on the z-axis) of the wing in metres.
Width	The chord (width on the x-axis) of the wing in metres.
Relative Position	The position of the wing in local space.
Lift Direction	The direction of the lift force of the wing in local space.
Damage Region	Which damage region this part belongs to.
Min Performance	The minimum possible performance level of this wing (i.e. what performance level the wing will have when its health reaches zero). The performance level affects how much lift the wing generates.
Starting Health	The initial health value of this part. This is the amount of damage that needs to be done to the damage region this part is associated with for the part to reach its min performance.

Control Surfaces

Control surfaces allow you to simulate the effect of moving parts changing the lift and drag properties of the ship in order to control the movement of the ship. Control surfaces require the ship to be moving relative to air around it in order to operate.

Property	Description
Type	The type of control surface this is. The type determines how the control surface moves and what inputs control it. Ailerons control the roll of the ship, and are generally required to be placed symmetrically on opposite sides of the ship. Elevators control the pitch of the ship, and should generally be placed behind the centre of mass of the ship. Rudders control the yaw of the ship and should generally be placed in the middle of the ship (not to the left or to the right). Air brakes are used to slow the ship down.
Length	The span of the control surface in metres.
Width	The chord of the control surface in metres.
Relative Position	The position of the control surface in local space.
Damage Region	Which damage region this part belongs to.
Min Performance	The minimum possible performance level of this control surface (i.e. what performance level the control surface will have when its health reaches zero). The performance level affects how effective the control surface is.
Starting Health	The initial health value of this part. This is the amount of damage that needs to be done to the damage region this part is associated with for the part to reach its min performance.
Use Brake Component	Whether a brake component is used.
Brake Strength	The strength of the braking force.
Ignore Medium Density	Whether the strength of the brake force ignores the density of the medium the ship is in (assuming it to be a constant value of one kilogram per cubic metre).
Min Acceleration	The minimum braking acceleration (in metres per second) caused by the brake when the brake is fully engaged. Increase this value to make the ship come to a stop more quickly at low speeds.

Combat Tab

Includes how the ship takes damage, how the ship is re-spawned, and how weapons are configured.

Damage

There are three damage modes: simple, progressive, and localised.

Damage Model	Description	Damage Regions
Simple	Ship has a single health value. An effects object, like an explosion prefab, can be assigned for when the health of the ship reaches 0.	Only 1 with or without shielding
Progressive	Allows the performance of certain ship parts (i.e. thrusters, wings, weapons) to decrease as the ship takes damage. Individual parts can optionally take Progressive Damage. They can have different Starting Health values.	Only 1 with or without shielding
Localised	Allows 1 or more parts to be assigned to a Damage Region of the ship. The performance of those parts in that region are decreased as the region takes damage. Each region can be assigned an effects object for when its health reaches 0.	1 or more regions with or without shielding.

Progressive or localised damage can be assigned to parts including thrusters, wings, control surfaces and weapons.

Property	Description
Damage Model	Determines how damage is calculated and applied to the ship. In Simple mode, the ship has a single health value. The only effects of damage are "visual until the health value reaches zero, at which point the ship is destroyed and optionally respawns. In Progressive mode, as the ship takes damage the performance of parts is affected. In localised mode different parts can be damaged independently of each other.
Starting Health	How much 'health' the ship has initially.
Use Shielding	Whether the main damage region uses shielding. Up until a point, shielding protects the ship from damage (which can affect the performance of parts on the ship).
Damage Threshold	Damage below this value will not affect the shield or the ship's health while the shield is still active (i.e. until the shield has absorbed 'amount' damage from damage events above the damage threshold).
Amount	How much damage the shield can absorb before it ceases to protect the ship from damage.
Col. Damage Resistance	Value indicating the resistance of the ship to damage caused by collisions. Increasing this value will decrease the amount of damage caused to the ship by collisions.
Effects Object	The particle and/or sound effect prefab that will be instantiated when the ship is destroyed. See also Effects Module.

Local Damage Regions can define areas of a ship that you want to receive individual damage. Examples could be “Engines” or “Bridge” or “Forward Weapons”. It could also be a strategic area of your ship that is critical for completing missions like “(Food) Gallery”, “Armory”, “Hanger”, “Landing Gear”, “Tractor Beam” etc.

Property	Description
Name	The name of the damage region
Relative Position	Position of this damage region in local space relative to the pivot point of the ship. Together with the size it determines what area the damage region encapsulates. This is the area that damage must occur at to impact this damage region.

Property	Description
Size	Size of this damage region (in metres cubed) in local space. Together with the relative position it determines what area the damage region encapsulates. This is the area that damage must occur at to impact this damage region.
Starting Health	How much 'health' the damage region has initially.
Use Shielding	Whether this damage region uses shielding. Up until a point, shielding protects the ship from damage (which can affect the performance of parts on the ship).
Damage Threshold	Damage below this value will not affect the shield or the damage region's health while the shield is still active (i.e. until the shield has absorbed 'amount' damage from damage events above the damage threshold).
Amount	How much damage the shield can absorb before it ceases to protect the damage region from damage.
Col. Damage Resistance	Value indicating the resistance of the damage region to damage caused by collisions. Increasing this value will decrease the amount of damage caused to the damage region by collisions.

Respawning

When a ship's health is reduced to 0 and it is destroyed, respawning options allow you to determine what happens next.

Property	Description
Respawning Mode	How respawning happens after the ship is destroyed. Options include "Don't respawn", "Respawn at Original Position", "Respawn at Last Position", "Respawn at Specific Position"
Respawn Time	How long the respawn process takes (in seconds). This lets you delay the respawning so that the ship doesn't immediately reappear after being destroyed. You might want to allow enough time for an effect to run (like an explosion).
Respawn Position	The position in world space that the ship respawns from.
Respawn Rotation	The rotation in world space that the ship respawns with.
Col. Respawn Delay	The time (in seconds) between updates of the collision respawn position. Hence when the ship is destroyed by colliding with something, the ship respawn position will be where the ship was between this time ago and twice this time ago. Only relevant when Respawning Mode is set to "Respawn At Last Position".
Respawn Velocity	The velocity in local space that the ship respawns with.

Weapons

Currently we support two kinds of ship-mounted weapons: Fixed Projectile canons and Turrets that fire Projectiles. Each weapon type can have multiple cannons/barrels/fire positions. All fire positions on the same weapon must fire in the same direction.

The first step to setup a Projectile weapon is to create a Projectile prefab. There are a couple of samples in SCSM\SciFiShipController\Prefabs\Projectiles to get you started. To create your own, use one of the samples as a template. Every Projectile requires a Projectile Module script attached to the parent gameobject. See the section called "Projectile Module" later in this manual.

Property	Description
Name	Name of the weapon. E.g. Front Cannon, Front Right Turret
Type	The type or style of weapon. Current we support Fixed Projectile or Turret Projectile weapons.
Relative Position	The position of the weapon in local space.
Multiple Fire Positions	If this weapon has multiple cannons or barrels

Property	Description
Fire Position Offsets	The positions of the cannon or barrel relative to the position of the weapon.
Fire Direction	The direction in which the weapon fires projectiles in local space. +ve Z is fire forwards, -ve Z is fire backwards.
Projectile Prefab	Prefab template of the projectiles fired by this weapon. Projectile prefabs need to have a Projectile Module script attached to them.
Reload Time	The minimum time (in seconds) between consecutive firings of the weapon.
Firing Button	The firing button or mechanism to use for this weapon. For a Player ship, this matches the Player Input Module. Fixed weapons can use the Primary or Secondary fire buttons from the Player Input Module, while Turrets also have the option of Auto-Fire.
Unlimited Ammo	Can this weapon keep firing and never run out of ammunition?
Ammunition	The quantity of projectiles or ammunition available for this weapon if there isn't unlimited ammo.
Damage Region	Which damage region this part belongs to.
Turret Pivot Y	For turret weapons, the transform of the pivot point around which the turret turns or rotates on the local y-axis
Turret Pivot X	For turret weapons, the transform on which the barrel(s) or cannon(s) elevate up or down on the local x-axis
Turret Min. Y	The minimum angle on the local y-axis the turret can rotate to
Turret Max. Y	The maximum angle on the local y-axis the turret can rotate to
Turret Min. X	The minimum angle on the local x-axis the turret can elevate to
Turret Max. X	The maximum angle on the local x-axis the turret can elevate to
Turret Move Speed	The rate at which the turret can rotate

To create laser beam style weapons, use a Projectile Module prefab like ProjectileBasic5 which is included with SSC. It uses a Particle System component with a Trails item. To vary the length of the laser "beams", modify the Particle System's Start Speed.

Player Input Module

This module enables you to map inputs from input sources to the six axes of the ship:

- Horizontal Input (left/right movement)
- Vertical Input (up/down movement)
- Longitudinal Input (forwards/backwards movement)
- Pitch Input (rotation on the local x-axis)
- Yaw Input (rotation on the local y-axis)
- Roll Input (rotation on the local z-axis)

It also enables you to map inputs for the weapon system.

The Player Input Module is very flexible and can take input from multiple sources including:

- Direct Keyboard
- Legacy Unity (input system)
- Oculus API
- Rewired
- Vive

The Player Input Module script should be added to the parent gameobject of your player "Ship" prefabs along with a Ship Control Module script. If you are creating your own ship setup, we recommend copying the Player Input Module

component from one of the ship prefabs included and “Paste Component as New” onto your ship parent gameobject. Modifying an existing Player Input Module component can be faster than starting from scratch.

At runtime in the Unity Editor, the Player Input Module has a Debug Mode which will help you identify what input data is being set to the Ship Control Module based on your configuration.

Direct Keyboard

When "Direct Keyboard" mode is selected, input can be modified by simply changing the key on the keyboard bound to each input.

Legacy Unity

When "Legacy Unity" is selected the Unity input manager will be used to receive input, requiring input axes to be set up in the Unity input manager and then referenced in the Player Input Module to bind input. For information on how to use the Unity input manager, see the Unity manual pages:
https://docs.unity3d.com/Manual/ConventionalGameInput.html

https://docs.unity3d.com/Manual/class-InputManager.html

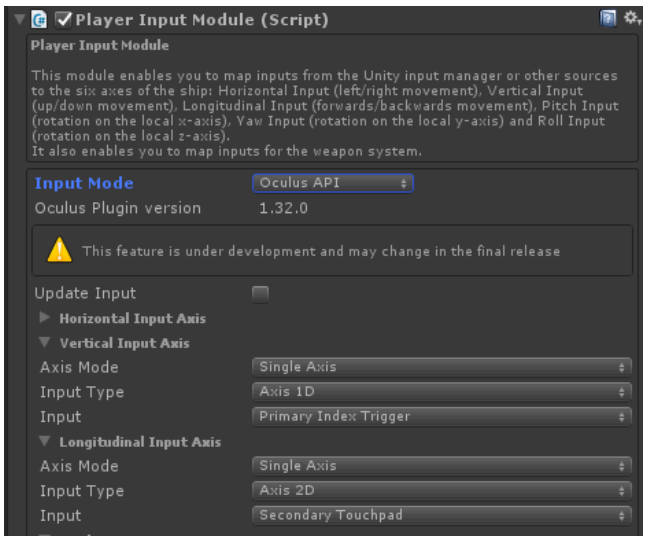
Virtual Reality (VR) Input

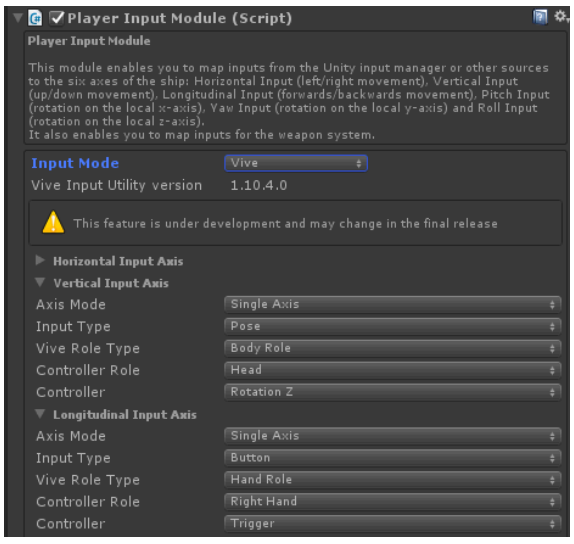
The Player Input Module can be configured to take input from a wide variety of VR controllers and devices. There are two options: Oculus API or Vive Input Module. Both require some 3rd party components which are not included with SSC.

VR System	Description
Oculus API	This requires the downloading and setup of the 3rd party Oculus package. https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022 See also https://www.oculus.com/setup/ https://docs.unity3d.com/Manual/OculusControllers.html SSC was tested with version 1.32.0.
Vive Input Module	This requires the downloading and setup of the 3rd party Vive package. https://assetstore.unity.com/packages/tools/integration/vive-input-utility-64219 SSC was tested with version v1.10.4.

If you have a newer version of the plug-ins and see any issues, please report them to us. The best place to do this is in our Unity forum.

Once the Oculus or Vive packages have been installed and you have restarted Unity, by selecting the appropriate “Input Mode” you should be able to see the 3rd party version number and the “Input Type” for each axis or button.





To enable VR support in Unity, it must be enabled in the Unity Player Settings under “XR”. XR is an umbrella term, encompassing Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR) applications. For more information see:

https://docs.unity3d.com/Manual/XR.html

Rewired

This popular 3rd party package supports a vast array of input controllers. To use Rewired with SSC you need to have separately purchased it from the Unity Asset Store and imported it into your project.

For more information on Rewired see:

https://assetstore.unity.com/packages/tools/utilities/rewired-21676

Before configuring the Player Input Module with Rewired, you first need to setup the Rewired Input Manager in the scene. If you are not familiar with Rewired, here are the basic steps.

1. Add Rewired Input Manager to scene
2. Open Input Manager
3. Add a Player
4. Add Action Categories
5. Add Actions to those Categories
6. Add Joystick Maps (for gamepad use [T] Gamepad Template - see below)
7. Add Keyboard Maps
8. Add Joystick Maps to a Player (ensure one is set to Start Enabled)
9. Add Keyboard Maps to a Player (ensure one is set to Start Enabled)
10. For the Player, ensure Assign Keyboard on Start is enabled

For convenience, if you are using Unity 2019.1 with Rewired 1.25.2 or newer, we have included a quick setup prefab in the SCSM\SciFiShipController\Demos\3rdParty folder. Make sure you check out the Rewired_Readme file in the same folder first.

For PC with some kind of gamepad (e.g. Xbox One, Sony Dual Shock 4), we’d suggest setting up a Gamepad Template like in the table below.

SSC Action	**[T] Gamepad Template**	**Element Properties**	**Xbox One Controller**	**Xbox 360**	**Sony Dual Shock 4**
Yaw	Left Stick X	Full	Left Stick X	Left Stick X	Left Stick X
Pitch	Left Stick Y	Full	Left Stick Y	Left Stick Y	Left Stick Y
Roll	Right Stick X	Full	Right Stick X	Right Stick X	Right Stick X
Vertical	Right Stick Y	Full	Right Stick Y	Right Stick Y	Right Stick Y
Longitudinal +ve	Right Shoulder 2	Range +ve, Contribution +ve	Right Trigger	Right Trigger	Right Trigger
Longitudinal -ve	Left Shoulder 2	Range +ve, Contribution -ve	Left Trigger	Left Trigger	Left Trigger
Primary Fire	Action Bottom Row 1		A Button		X Button
Secondary Fire	Action Top Row 2		Y Button		Triangle Button

The Player Input Module works for both Axis and Button input. It has been tested with Keyboard Maps and Joystick Maps in Rewired. If you see any issues, please let us know.

Using Sony Dual Shock 4 with SSC on PC

If you don't have an Xbox One controller, but have a Sony Dual Shock 4, you are all set. You can use the legacy Unity Input method or, if you have the excellent Rewired asset, then the Sony controller will work with the suggestion config here.

To connect your Sony controller to a Windows 10 PC, follow these steps:

Wired

Use a micro-USB to USB cable, plug the small end into the controller and the regular USB into your PC. Your controller is now ready! How easy is that?!

Wireless

1. Make sure the Sony Dual Shock 4 controller is fully charged
2. Turn the controller off
3. On the Windows 10 PC make sure Bluetooth is enabled
4. On Windows 10 PC type "Bluetooth and other device settings" in search bar (bottom left of screen)
5. Click "Add Bluetooth or other device"
6. Click "Bluetooth"
7. On Sony controller, press the "Playstation" button (button between joysticks) and the "Share" button (just to the right of left arrow pad on the top left of the controller).
8. The light bar on the front of the controller should start flashing.
9. When "Wireless Controller" or "Input" appears, click on it. Windows 10 will then automatically pair the controller with the PC.
10. Now, you're good to go!

Ship Camera Module

This provide basic camera control. So instead of the fixed camera position you can deliver smooth camera movement out of the box by adding this script to an empty gameobject in the scene.

For convenience, a prefab has been included SCSM\SciFiShipController\Prefabs\Environment called “PlayerCamera”. Drag and drop it into a scene, assign the Target Ship to your Player ship, and you’re ready to test.

Property	Description
Target Ship	The target ship for this camera to follow.
Lock Camera Pos	If enabled, the camera will stay at the same position and rotate towards the target. Only available when Camera Rotation Mode is set to Aim To Target.
Target Offset	The offset from the target (in local space) for the camera to aim for.
Lock To Target Pos	If enabled, the camera will stay locked to the optimal camera position.
Move Speed	How quickly the camera moves towards the optimal camera position.
Lock Target To Rot	If enabled, the camera will stay locked to the optimal camera rotation.
Turn Speed	How quickly the camera turns towards the optimal camera rotation.
Camera Rotation Mode	How the camera rotation is determined. • FollowVelocity: The camera rotates to face in the direction the ship is moving in. • FollowTargetRotation: The camera rotates to face the direction the ship is facing in. • AimAtTarget: The camera rotates to face towards the ship itself.
Follow Velocity Threshold	Below this velocity (in metres per second) the forwards direction of the target will be followed instead of the velocity.

Property	Description
Orient Upwards	If enabled, the camera will orient with respect to the world up direction rather than the target's up direction.
Update Type	When the camera position/rotation is updated. • FixedUpdate: The update occurs during FixedUpdate. Recommended for rigidbodies with Interpolation set to None. • LateUpdate: The update occurs during LateUpdate. Recommended for rigidbodies with Interpolation set to Interpolate. • Automatic: When the update occurs is automatically determined.

Projectile Module

Every Weapon that fires a projectile requires a prefab with a Projectile Module script attached. For details on using projectile weapons on a Ship, see the “Combat Tab” under the “Ship Control Module” section in this manual.

To create a new Projectile:

1. In a scene, create an empty gameobject
2. Rename the empty gameobject. E.g. MyProjectile1
3. Either add a mesh and mesh renderer directly to the gameobject, or as a child gameobject
4. Remove any colliders that may have been added in step 3 (SSC doesn’t depend on projectile colliders for projectile collisions)
5. Add a Projectile Module script to the parent gameobject
6. Create a prefab from the gameobject by dragging the parent gameobject into a folder in the Project pane
7. Reset the prefab parent transform position and rotation to 0,0,0
8. Delete the gameobject from the scene

The Projectile Module script has three options for spawning projectiles from a weapon’s canon or fire position.

Projectile system	Description
Standard	The standard behaviour doesn’t have pooling or DOTS enabled. It relies on the standard Unity object instantiation and destruction methods. It works well when there are only a few projectiles in the scene.
Pooling	An object pooling system to manage create, re-use, and destroy projectiles. This is more efficient than the “standard” projectile system.
DOTS	This is a high-performance method of creating and destroying projectiles that utilises the new Unity Data-Orientated Technology Stack (DOTS). Projectiles are Entities rather than GameObjects. SSC converts the projectile prefabs to entities at runtime. For games that require high-performance, low overhead projectiles, this is the preferred choice.

How to Setup Projectiles to use DOTS/ECS

1. In Unity 2019.1.0f2 or newer open Package Manager
2. Install the following packages (or newer)
   - Collections 0.0.9 preview 17
   - Mathematics 1.0.0 preview 1
   - Jobs 0.0.7 preview 10
   - Entities 0.0.12 preview 30
   - Hybrid Renderer 0.0.1-preview.10
   - Burst 1.0.0-preview.12
3. On the projectile prefab, in the Projectile Module script inspector, enable “Use DOTS”

For Unity 2019.2+, from the Package Manager, install Entities 0.0.12 preview 33, then Hybrid Renderer 0.0.1 preview 13. Unity will automatically install all other dependencies.

Property	Description
Start Speed	The starting speed of the projectile when it is launched.
Use Gravity	Whether gravity is applied to the projectile.
Damage Amount	How much damage the projectile does on collision with a ship.
Use DOTS	Use Data-Oriented Technology Stack which uses the Entity Component System and Job System to create and destroy projectiles. Has no effect if Unity 2019.1, ECS, and Jobs is not installed.
Use Pooling	Use the Pooling system to manage create, re-use, and destroy projectiles.
Min Pool Size	When using the Pooling system, this is the number of projectile objects kept in reserve for spawning and despawning.
Max Pool Size	When using the Pooling system, this is the maximum number of projectiles permitted in the scene at any one time.
Despawn Time	If the projectile has not collided with something before this time (in seconds), it is automatically despawned or removed from the scene
Effects Object	The particle system and/or sound effect prefab that will be instantiated when the projectile hits something and is destroyed. This does not fire when the projectile is automatically despawned. See also Effects Module.

When “Use Pooling” is enabled, the optimal Min/Max Pool Size can be determined by:

1. Running the game in the editor
2. Pausing the game, then finding SSCManager in the scene hierarchy
3. Enabling “Debug Mode”
4. Taking note of the current pool sizes for Projectile Templates

If the current pool size never approaches the Max Pool Size, then the Max Pool Size can be reduced.

Important

When modifying existing prefabs that come with SSC, we recommend creating a new prefab (i.e. a copy of the prefab) and placing it in your own project folder outside the SciFiShipController folder. This way, when you download an SSC update, your changes won’t be overwritten.

Effects Module

This module enables you to implement effects behaviour on the object it is attached to. This can include multiple child particle systems and/or an audio source attached to the parent gameobject. Create a prefab so that it can be assigned within the Ship Control Module.

When “Use Pooling” is enabled, and an Audio Source is attached to the prefab’s parent gameobject, ensure that “Play On Awake” is not enabled on the Audio Source.

Property	Description
Use Pooling	Use the Pooling system to manage create, re-use, and destroy effects objects.
Min Pool Size	When using the Pooling system, this is the number of effects objects kept in reserve for spawning and despawning.
Max Pool Size	When using the Pooling system, this is the maximum number of effects objects permitted in the scene at any one time.
Despawn Time	After this time (in seconds), the effects object is automatically despawned or removed from the scene.

When “Use Pooling” is enabled, the optimal Min/Max Pool Size can be determined by:

1. Running the game in the editor
2. Pausing the game, then finding SSCManager in the scene hierarchy
3. Enabling “Debug Mode”
4. Taking note of the current pool sizes for Effects Templates

If the current pool size never approaches the Max Pool Size, then the Max Pool Size can be reduced.

Common Issues

Common Issues – Player Input

1. Rewired is configured but nothing happens in play mode. In the Player Input Module, when the “Input Mode” is set to “Rewired”, the player needs to be assigned a Rewired player number. This can be done in code, by calling PlayerInputModule.SetPlayerNumber(..), or by setting the Player Number in the Player Input Module editor to a non-zero value, like 1,2,3 or 4.
2. My ship doesn’t move or respond correctly to player input. Although there are several causes, a handy feature in play mode in the Unity editor, is enable “Debug Mode” on the Player Input Module. This can help to determine which input is actually being received and is being sent to the ship.

Common Issues – Ship Behaviour

1. Large ships turn too quickly in Arcade mode. In the Ship Control Module editor, on the "Physics" tab, adjust the Pitch, Roll, and/or Yaw acceleration rates. Small numbers will give the user the impression they are piloting a large, heavy space ship. Default values around 100 are more suited to fighter-style craft.
2. One or more ships seem to jitter or stutter as they move. Ensure that the Interpolation setting on RigidBody components for ships in your scene near the player ship have the same setting.
3. In Arcade mode, when turning sharply at speed, the ship

Common Issues – Demo Scenes

1. Everything is pink. This occurs when the material shaders in the scene cannot be rendered. The most common cause is that the project was created with a different render pipeline like Lightweight Render Pipeline (LWRP) or High Definition Render Pipeline (HDRP). To apply the correct materials, load the appropriate package from the SciFiShipController\SRP folder. See the readme file in this folder for more details.

Common Issues – Effects

1. Sound on my effects prefab always ignores Volume Rolloff and Max Distance settings. This may be because Spatial Blend is set to 0. Try setting it to 1.
2. Some or all of my particle effects only play once when “Use Pooling” is enabled. This may be because one or more particle systems have a “Stop Action” other than “None”.
3. My thruster produces too few particles when only a small amount of thrust input is received. If at maximum thrust the effect looks correct, in Ship Control Module, on the Thrusters tab, increase the “Minimum Effects Rate” for that thruster.

Common Issues – Weapons

1. When DOTS is enabled for projectiles, they hit colliders with “Is Trigger” enabled. Either add those objects they hit to the “Ignore Raycast” Unity Layer in the scene, or switch to Pooling.
2. Particle effects on a projectile prefab don’t appear when DOTS is enabled on the Projectile Module. Currently there is little or no performance benefit for adding Particle Systems (and therefore GameObjects) to DOTS-enabled projectiles. To use particle systems attached to projectiles, switch to Use Pooling. As DOTS matures, we’ll be investigating new solutions in this area.

Runtime

SSC is designed to be used in your games. We expect your code to interact with ours. For example, you may wish to do any of the following:

- Spawn Non-Player-Character (NPC) friendly or enemy ships
- Move NPC ships with your own Artificial Intelligence (AI) code
- Set targets for turret weapons on ships
- Update game scoring whenever a ship is destroyed
- Fire weapons automatically from code
- Change the behaviour of a ship as it enters, exits a planet's atmosphere
- Enable or disable "Stick To Surface" in code
- Prevent weapons from firing during certain gameplay
- Assign or unassign a ship from a squadron

If there is anything that you'd like to control at runtime but cannot, please talk to us about that, we're want to provide you the best runtime experience possible.

Changing Variable at Runtime

Many public variables are modifiable at runtime from within your own code. Variables are commented so that (a) you know what they do, and (b) you can see if they require a method to be called after changing at runtime. For example, if you change ship.groundMatchResponsiveness or maxGroundMatchAccelerationFactor at runtime, you also need to call ship.ReinitialiseGroundMatchVariables().

Demo Scripts

We have included a collection of helpful scripts that show how certain features can be used in your games or projects. They are subject to change with version upgrades, so are not meant to be used directly in your projects. Instead, the intention is to help you build games with Sci-Fi Ship Controller by providing coding examples.

Script name	Description
BeaconLight	A script to rotate a light on the y-axis. If an audiosource (and clip) is attached to the same gameobject, it will activate and deactivate at the same time as the light. If including an audiosource, turn off "Play On Awake" for the audiosource. Assumes original rotation y is between -359.999 and +359.999
HideCursor	Sample script hide/show the mouse pointer due to mouse (in)activity. Drop it onto a gameobject in the scene.
SampleShipSpawner1	Sample ship spawner that leverages the in-build ShipSpawner class. In your project, you'll want to include "using SciFiShipController" namespace and write your own code.
SampleShowShipHealth	Sample script to show Health of a ship in the UI. This is only sample to demonstrate how API calls could be used in your own code.
SampleThrusterFXOverride	Sample script to show how to override the thruster particle effects and audio based on the speed of the ship. Attach it to the parent of a ship prefab. This assumes that the particle effects attached to thrusters have Loop enabled.
SampleWeaponAssignTarget	Sample script to assign targets to turret weapons on a ship at runtime. This is only a code segment to demonstrate how API calls could be used in your own code. Place it on an empty gameobject in the scene to see how it works.

Useful Runtime Methods or Properties

Custom runtime methods should be a lightweight to avoid performance issues. These single-cast delegates can have a single instance of a call-back method. This is useful when you want to take some (custom) action when something occurs, like when a ship is destroyed or is hit or damaged by a projectile.

Property or Method	Description

CallbackOnDestroy
callbackOnDestroy
	The name of the custom method that is called immediately before the ship is destroyed when it reaches a health of 0. Your method must take 1 parameter of class Ship. It could be used to update a score or remove a ship from a squadron. Create your own method which takes a parameter of the ship class.
	E.g.
	Public void MyPreDestroyMethod(Ship ship)
	{
	If (ship.someshipvariable == 'something') { DoSomething(); }
	}
	..
	shipControlModule.callbackOnDestroy = MyPreDestroyMethod;

CallbackOnHit callbackOnHit	The name of the custom method that is called immediately after the ship is hit by a projectile. Your method must take 2 int parameters (int sourceShipId, int sourceSquadronId). It could be used to take evasive action while being pursued by an enemy ship. It could be also used to detect friendly fire. shipControlModule.callbackOnHit = MyOnHitMethod;
CallbackOnDamage callbackOnDamage	The name of the custom method that is called immediately after damage has changed. Your method must take 1 float parameter (mainDamageRegionHealth). It could be used to update a HUD or take some other action. ship.callbackOnDamage = MyOnDamageMethod;

Support

For any other questions you have (or any suggestions on how we can improve Sci-Fi Ship Controller), feel free to contact us via our Unity forum (https://forum.unity.com/threads/594448/) or on our Discord channel (https://discord.gg/CjzCK4b).

Version History

Initial Release: August 2019