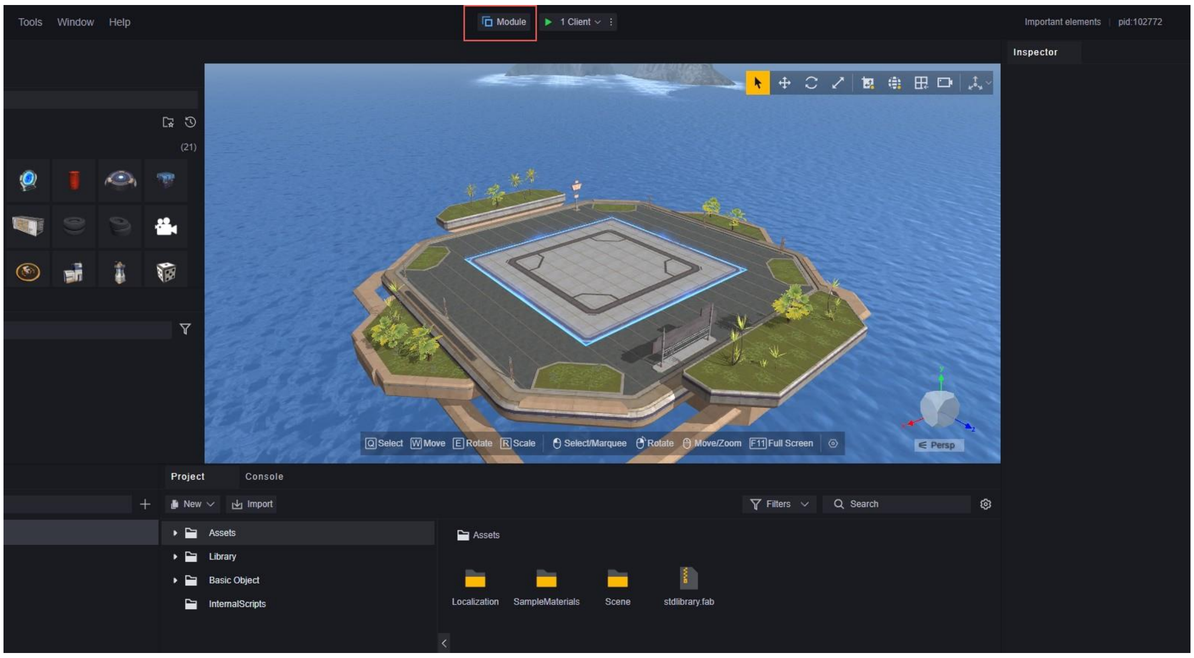
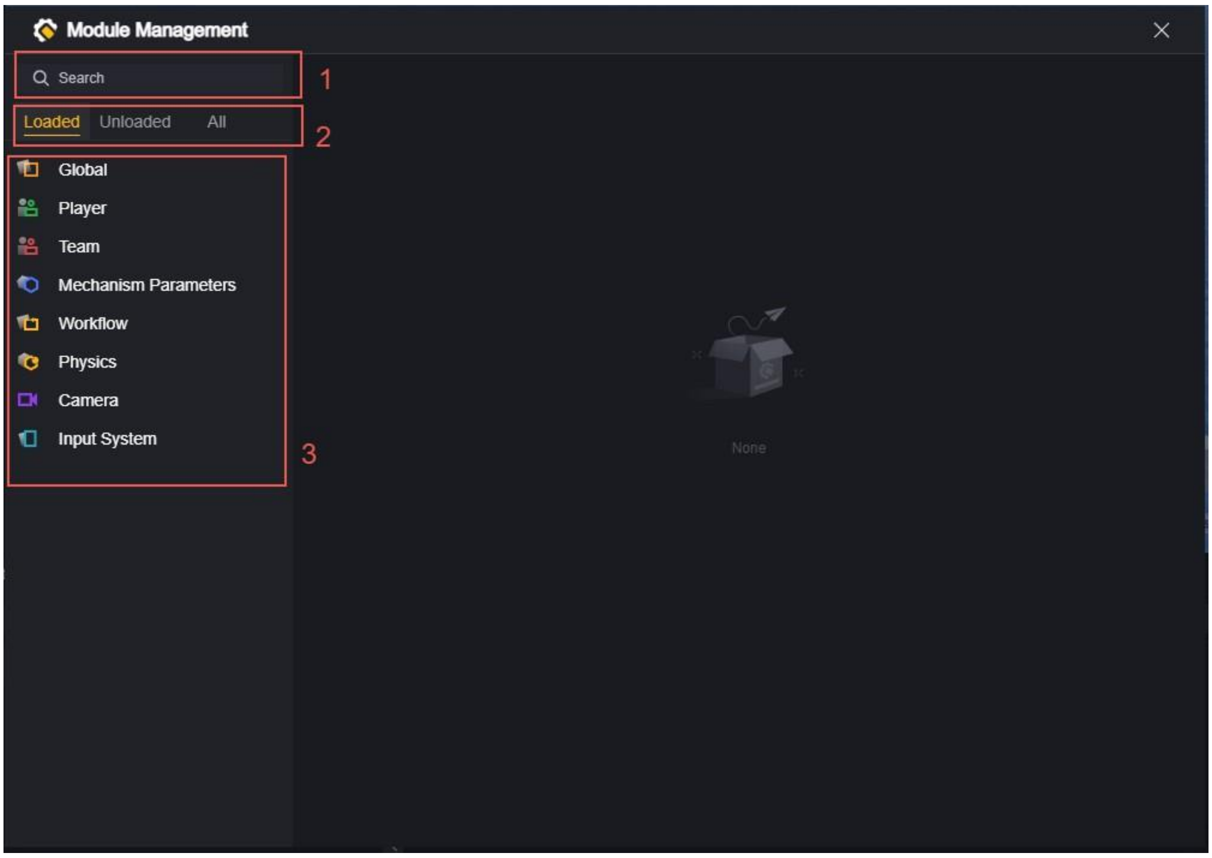
Important Concepts - User Manual

module (computer)

A module is a logically closed-loop functional block that encapsulates internal complexity and provides a simple interface to the outside world.
Modules generally consist of a class of configurations, assets, features, etc. that are related to each other, and can be loaded to set up game-related elements.

The Module Manager can be opened from the upper position in the editor:





1. All modules can be searched.

> The scope of the search is all modules, regardless of whether the module is loaded with the currently selected category or not, the corresponding module can be searched directly.

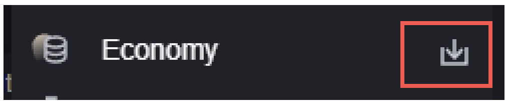
2. Modules are categorised into loaded, unloaded and all.
3. Module list, show all the modules under the category, click the module to view the configuration of the module.

Loading and unloading of modules

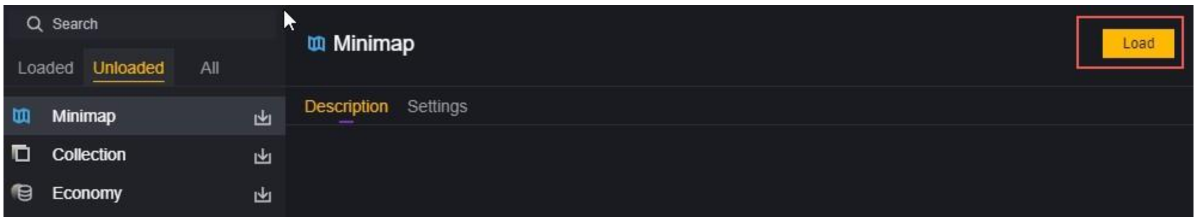
By default, we have loaded several modules for you. Modules under the unloaded category are optional.

Load Module

The flag at the back of the module indicates that the module has not yet been loaded. Click on the symbol to load the module.

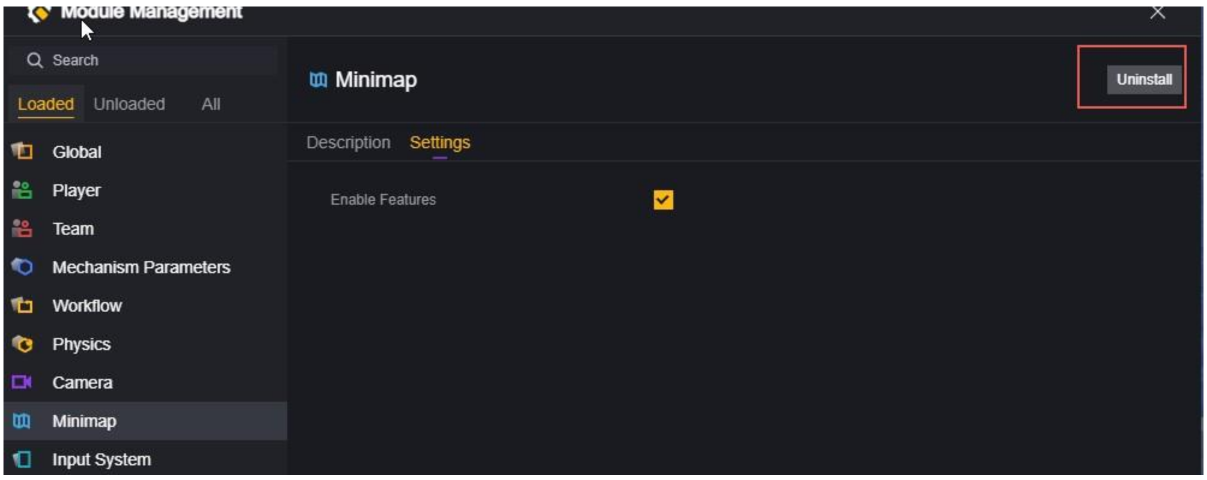


It can also be loaded via the Load button in the module configuration screen.



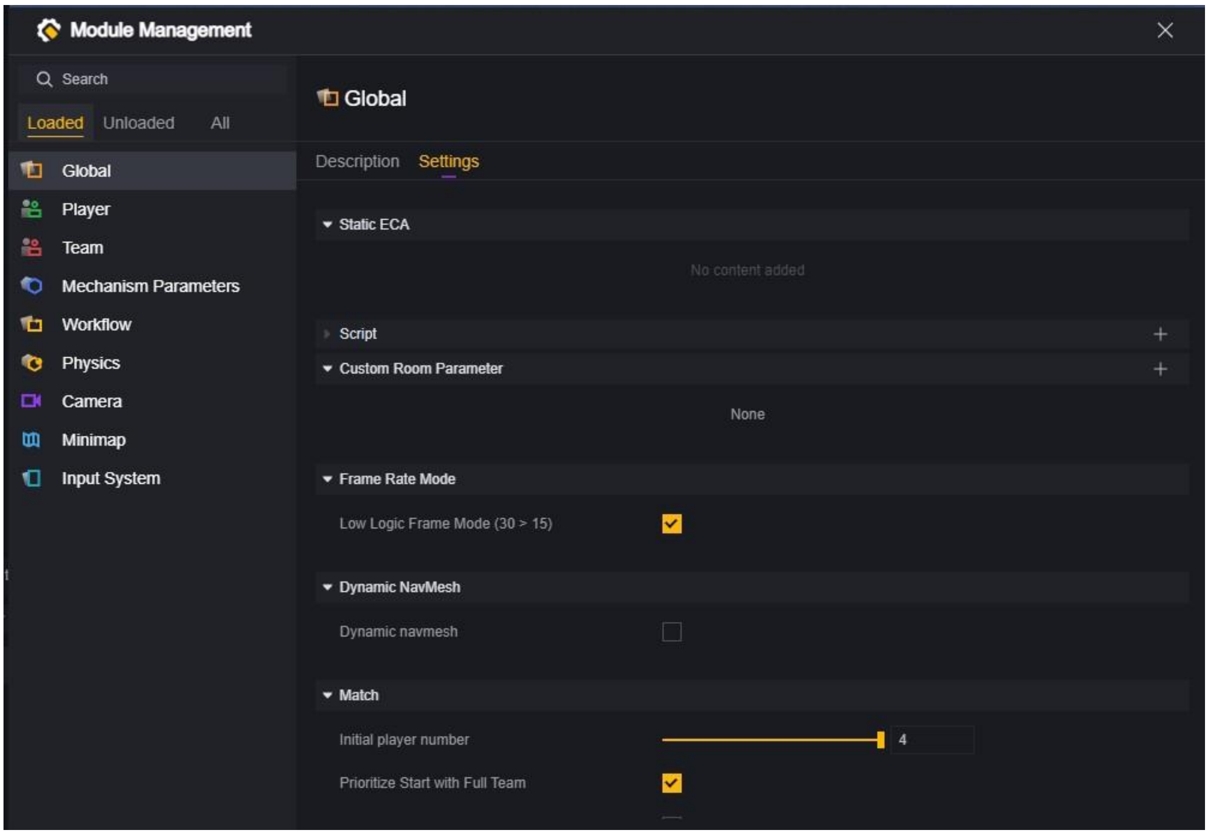
Unloading Module

For those modules that can be uninstalled, there will be an uninstall button on their configuration screen that can be clicked to uninstall them.



Enable Features ✓

Some modules are essential and cannot be uninstalled, only their configuration can be changed. These modules do not have an uninstall button.


Module Management

Search

Loaded Unloaded All

Global
Player
Team
Mechanism Parameters
Workflow
Physics
Camera
Minimap
Input System

Global

Description Settings

Static ECA

No content added

Script

Custom Room Parameter

None

Frame Rate Mode

Low Logic Frame Mode (30 > 15) ✓

Dynamic NavMesh

Dynamic navmesh ☐

Match

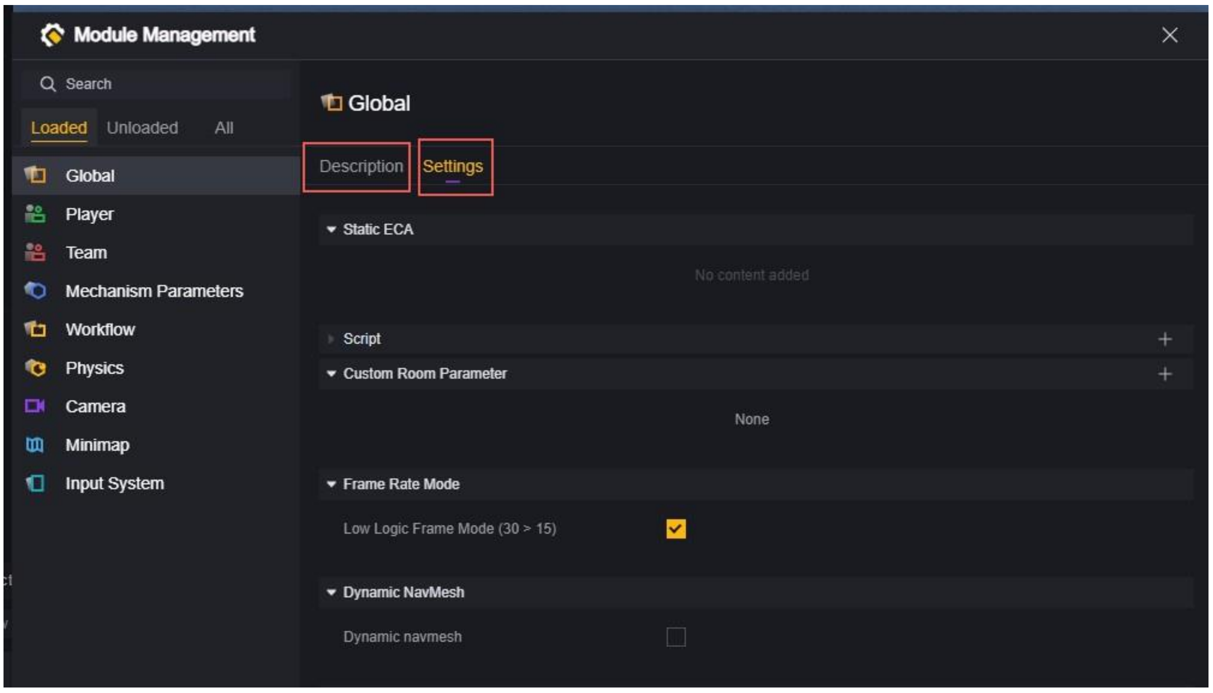
Initial player number 4

Prioritize Start with Full Team ✓

> Loading/unloading modules will not be reflected in the category display immediately, it may appear that the loaded module appears in the unloaded category column, which will be updated after refreshing the category tab.

Configuration of modules

For unloaded modules, there is only one available action: load. The loaded module can be configured in a specific way.


Module Management

Search

Loaded Unloaded All

Global
Player
Team
Mechanism Parameters
Workflow
Physics
Camera
Minimap
Input System

Global

Description Settings

Static ECA

No content added

Script

Custom Room Parameter

None

Frame Rate Mode

Low Logic Frame Mode (30 > 15) ✓

Dynamic NavMesh

Dynamic navmesh ☐

After selecting a loaded module, you can view the description of the module, as well as specific configuration items. For important module configurations, please refer to the corresponding manual articles.

Entity

Entity is one of the important concepts of UGC.

An entity is a class of figurative/abstract concepts or a collection of concepts. Entities have attributes, and modifications to the attributes affect the instances created by the entity.

The concept of entities is very broad, and the vast majority of the concepts in this game are in the form of entities.

Taking "player" as an example, the "player" entity refers to the figurative concept of a player, and life value is an

attribute of the player entity, a particular player in a game.

A and B are two instances of the player entity.

Players A and B both have the life-value attribute of the player entity, with the difference that Player A has a life-value of 150 and Player B has a life-value of 200. The life value is a property of the entity, and the 150 and 200 life values are properties of the instance.

In addition to entities that can be visualised like players, vehicles, and weapons, there are also abstract entities that cannot be directly visualised, such as the global and game flow.

Abstract entities also have attributes and instances are created as well.

For example, a game process is an abstract entity of which the preparation phase duration is an attribute, and assuming that two games are played, an instance of the game process is created for each game.
The first game preparation phase lasts 5 seconds and the second game preparation phase lasts 10 seconds. the 5 and 10 second preparation phase durations are instance attributes of the game flow entity.

In most cases, the entities mentioned in the game actually refer to instances of entities, e.g., deleting an entity, getting an entity's properties, modifying an entity's properties. The objects that are actually operated on are instances of entities. In the following and other articles, there is no need to de-emphasise the distinction between entity instances and entities.

Life cycle of an entity

As stated above, in reality, life cycle role objects are instances of entities.

Lifecycle means that an entity will always go through a specific process from creation to destruction, and you can get a handle on the l i f e c y c l e of an entity that you need to edit in order to manipulate it at a specific point in time for design purposes. All entities start their lives as Awake, and start their lives as

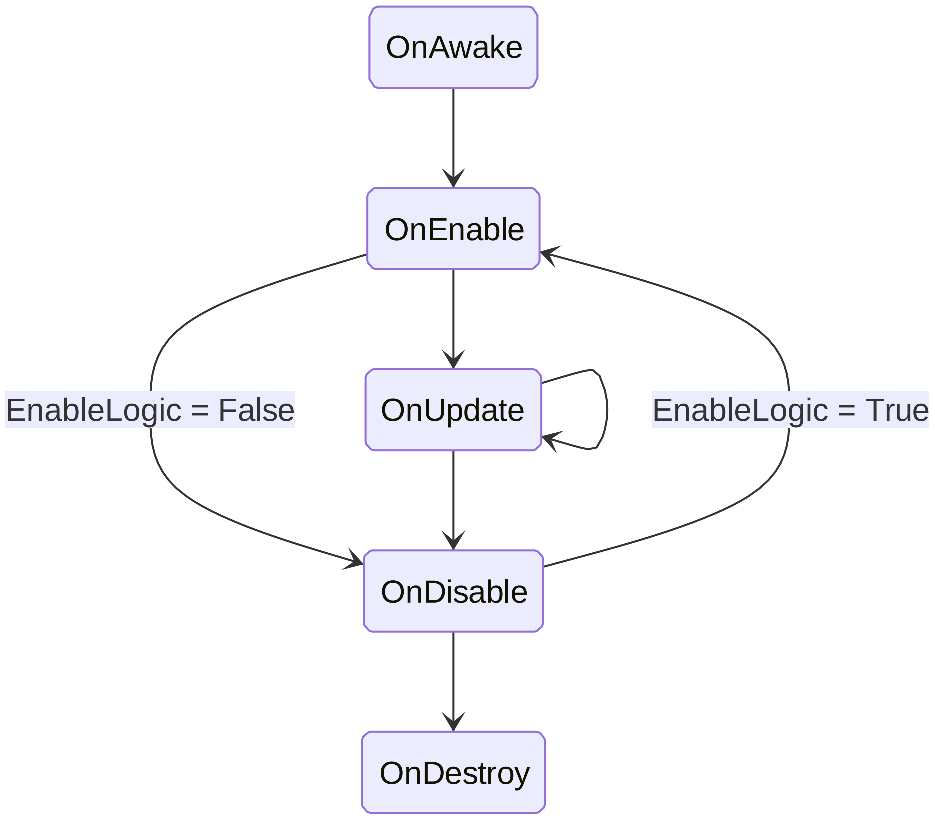
Destroy marks the end of life. The vast majority of entities are activated after Awake by Enable and destroyed

before using Disable

Switch off activation.

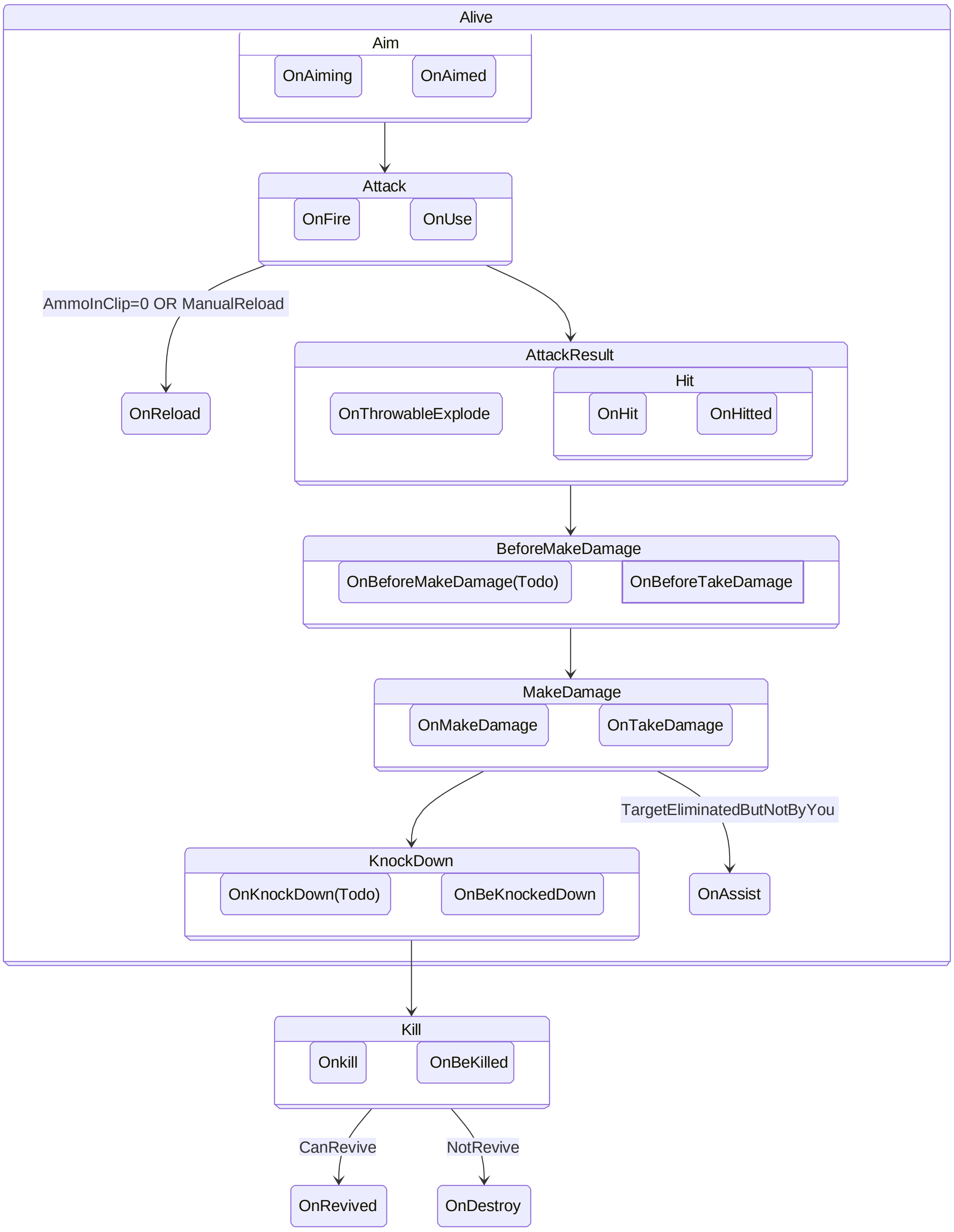
> There may be entities that skip the enable or disable phases, but awake and destroy are always present.

Common Processes

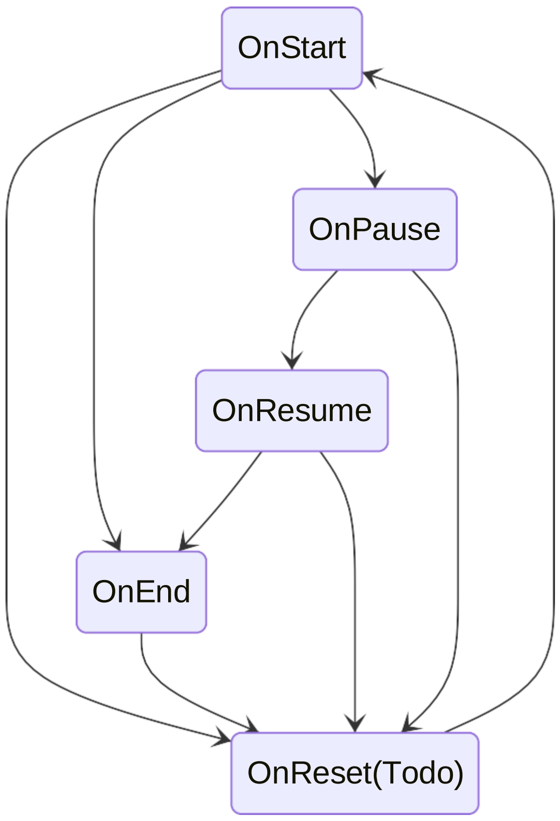


Before Awake, the initial values of all attributes are ready and the corresponding bridging entities are created. The update part of the life cycle of some entities is given below for reference:

Combat unit flow



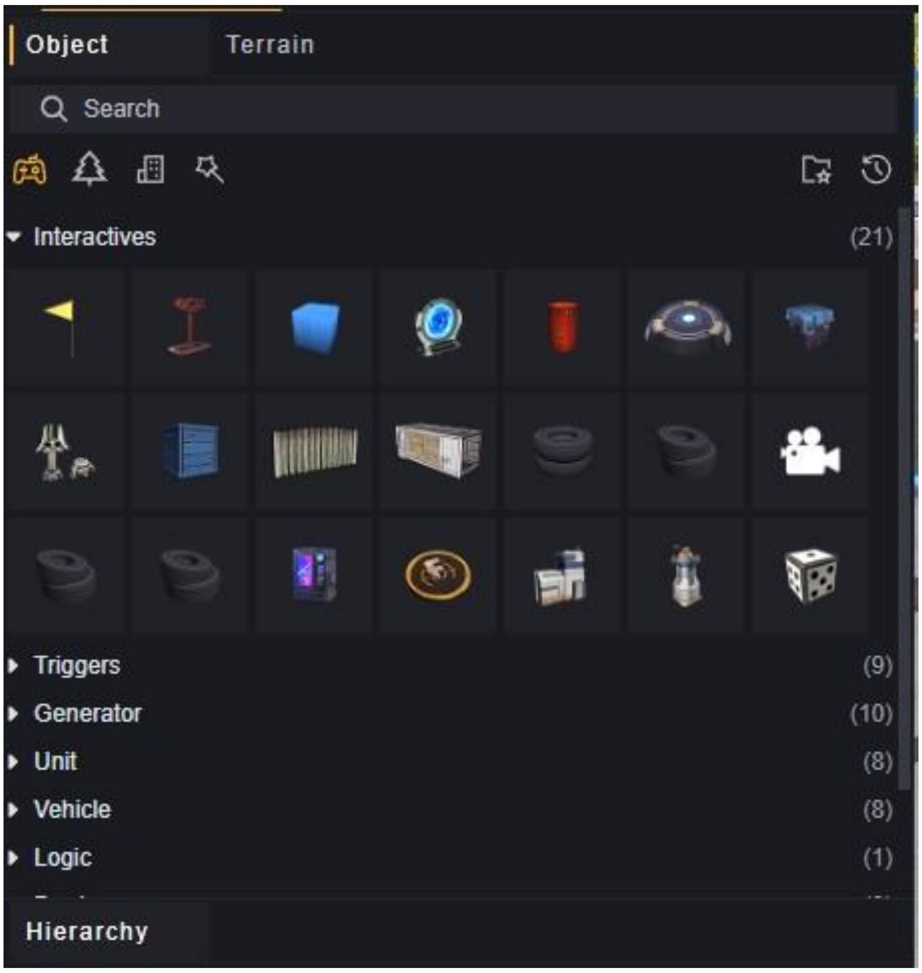
playable process



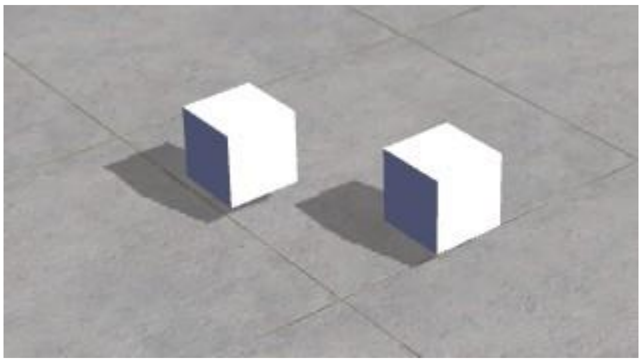
Scene Objects, Prefabs and Components

In the editor, you can place a variety of objects in the scene, which can be purely decorative, triggers, generators, and so on. You can also customise objects by modifying their components and properties.

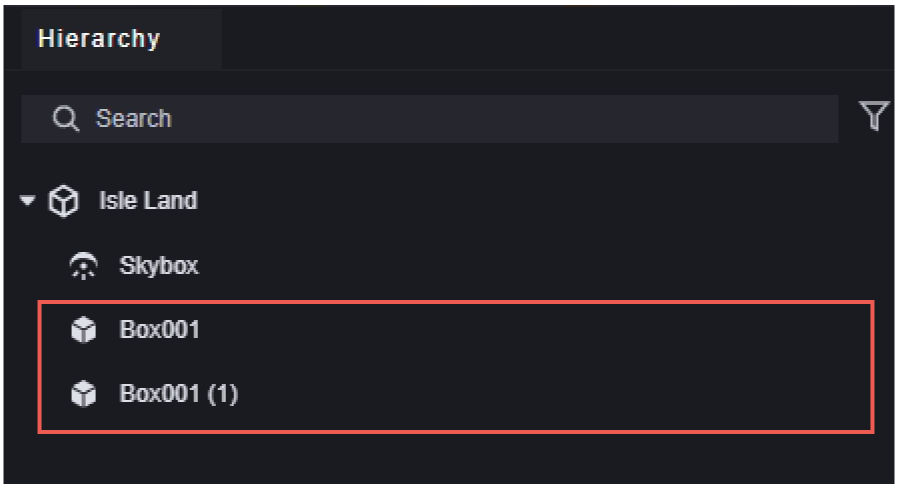
We provide several categories of preset objects in the editor that can be configured by dragging them onto the scene.



Using the basic object cube as an example, we configure two cubes onto the scene.

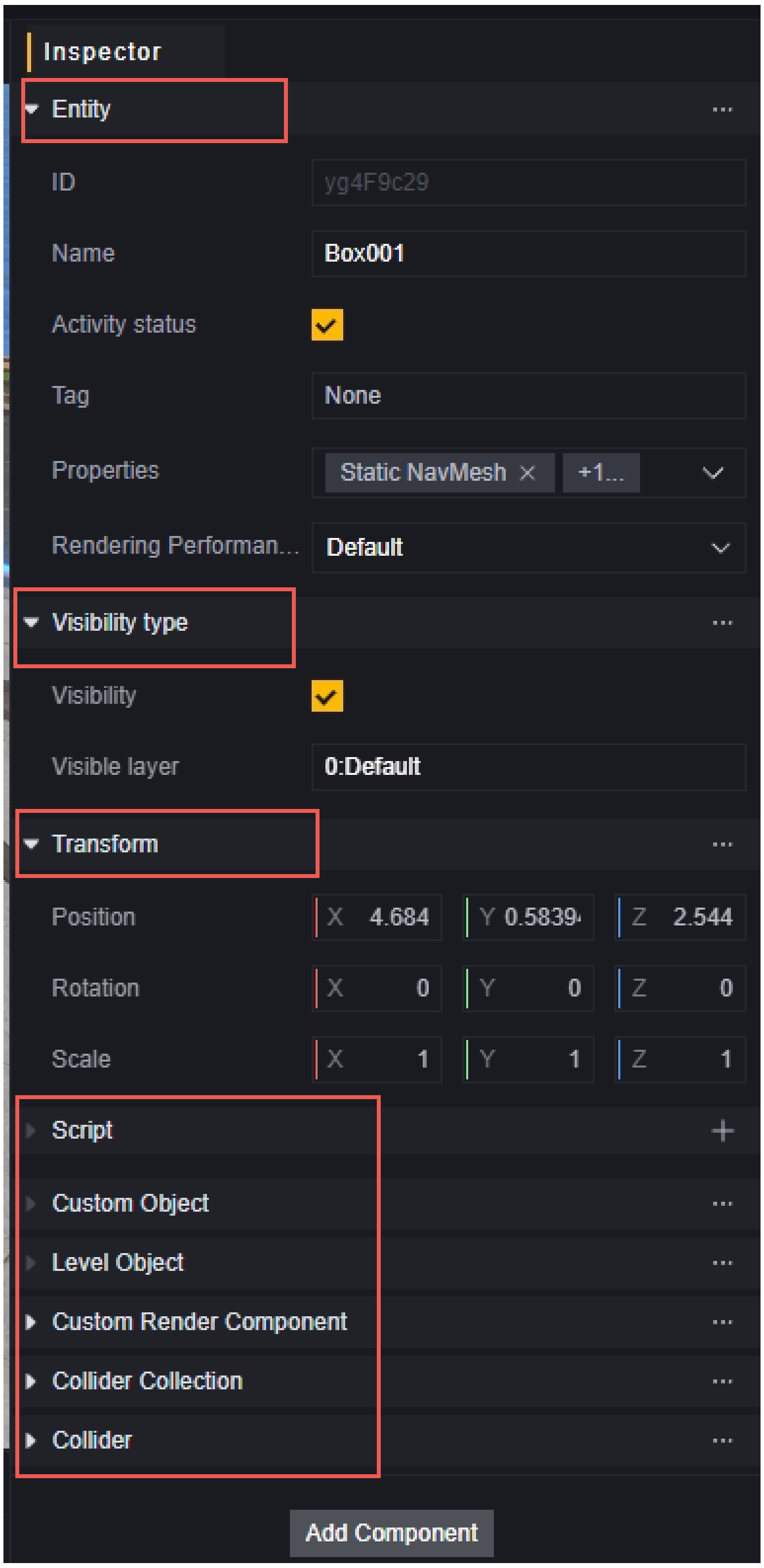


Configured objects are managed on the Hierarchy page.



Components define the behaviour of an object. Some components are required to be loaded on the corresponding object and some are optional. We provide a default configuration of components for objects, which you can modify as needed.

Take the cubes above as an example, select any cube and you can see that it has several components configured by default in the inspector panel.



Inspector

Entity

ID	yg4F9c29
Name	Box001
Activity status	✓
Tag	None
Properties	Static NavMesh × +1...
Rendering Performan...	Default

Visibility type

Visibility	✓
Visible layer	0:Default

Transform

Position	X 4.684	Y 0.5839	Z 2.544
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Script

Custom Object

Level Object

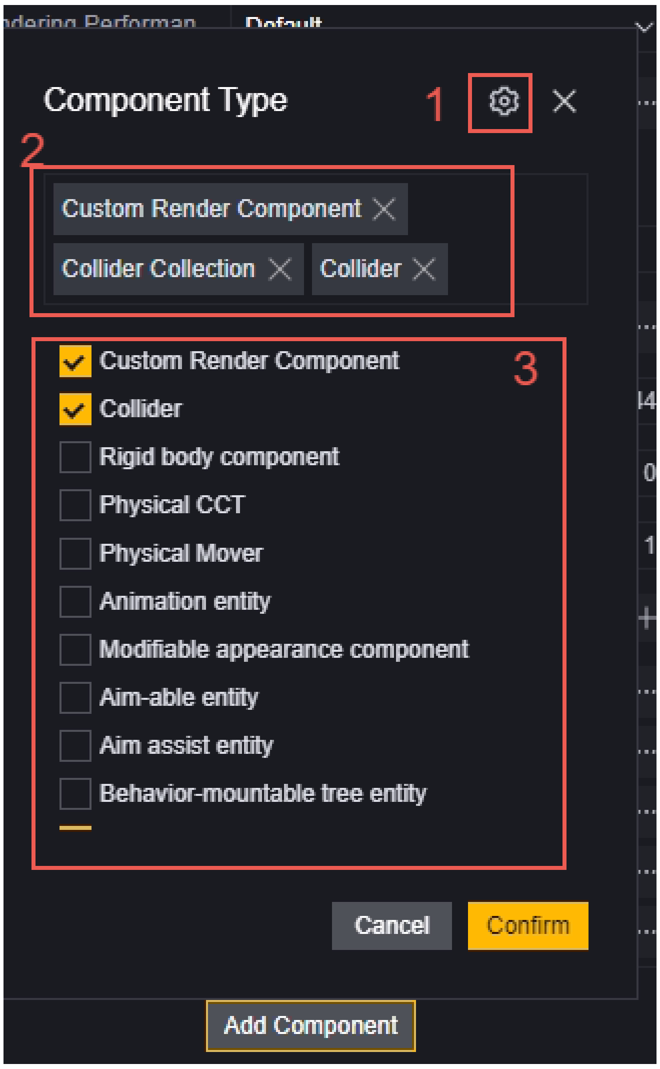
Custom Render Component

Collider Collection

Collider

Add Component

Which components are configured can be edited using the Add Component button.

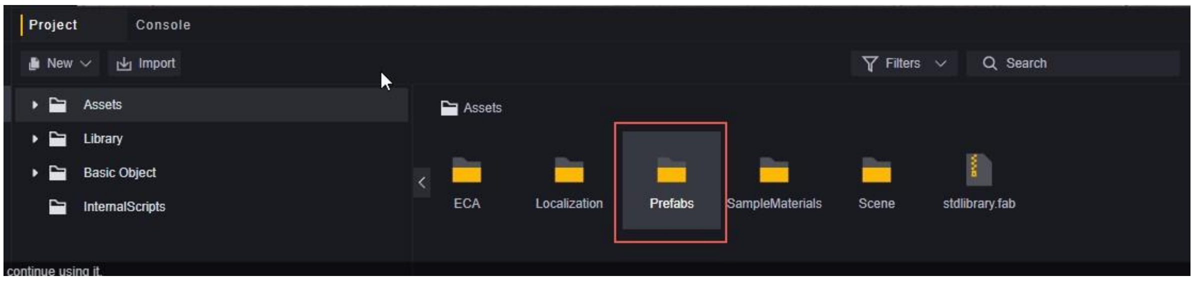


[✓ Custom Render Component]
[✓ Collider]
Rigid body component
Physical CCT
Physical Mover
Animation entity
Modifiable appearance component
Aim-able entity
Aim assist entity
Behavior-mountable tree entity

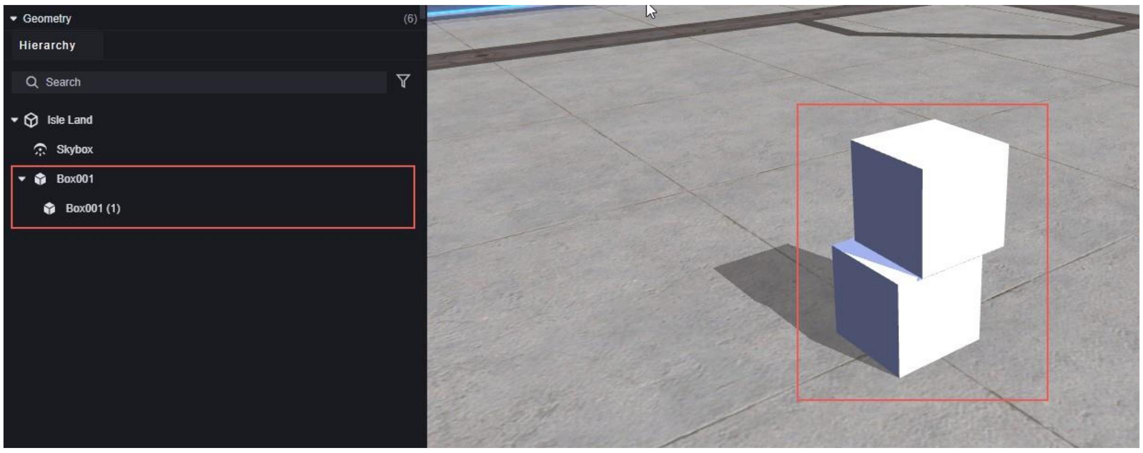
1. Settings are made to the component and the settings for the component are synchronised to all entities that have the component loaded.
2. Components that have been loaded.
3. Optional component, check to load, uncheck to unload.

Prefabs are templates for objects. Prefabs allow you to make changes to all objects configured by a prefab at the same time. A prefab is created by dragging an object on the scene into the asset bar.

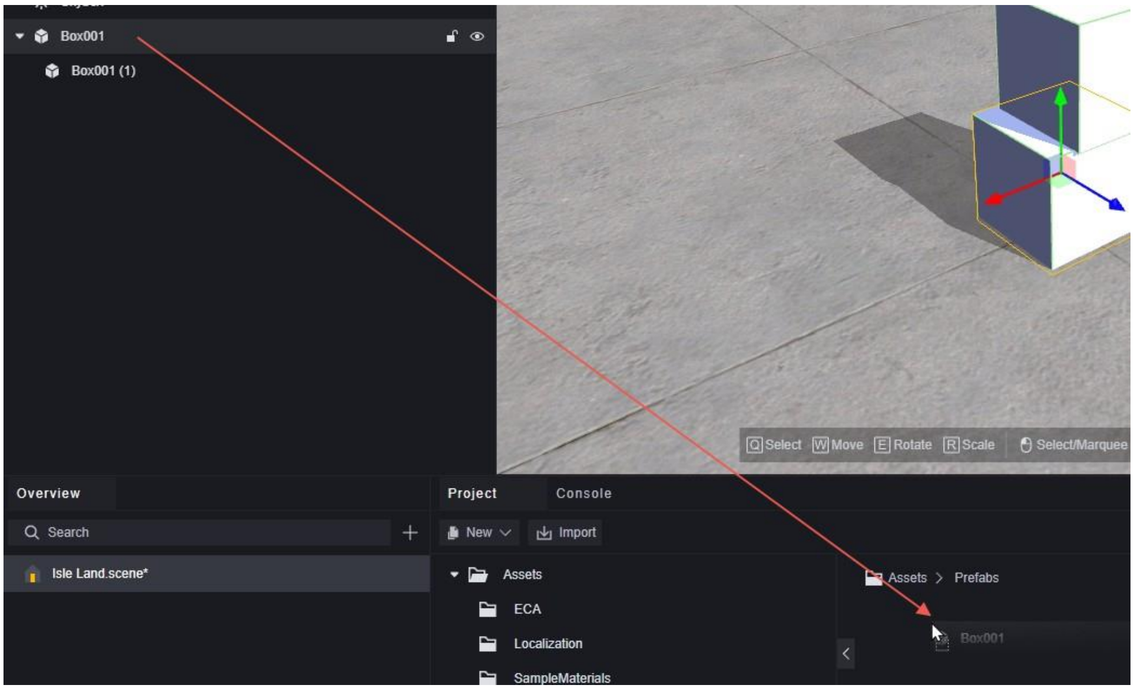
1. Create a folder of prefabricated bodies, this step is not necessary but recommended for easy file management.

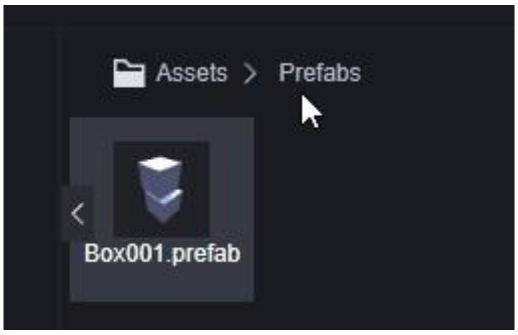


2. Edit objects that you need as prefabs, such as stacking two cubes on top of each other and creating hierarchies to merge them into a whole.



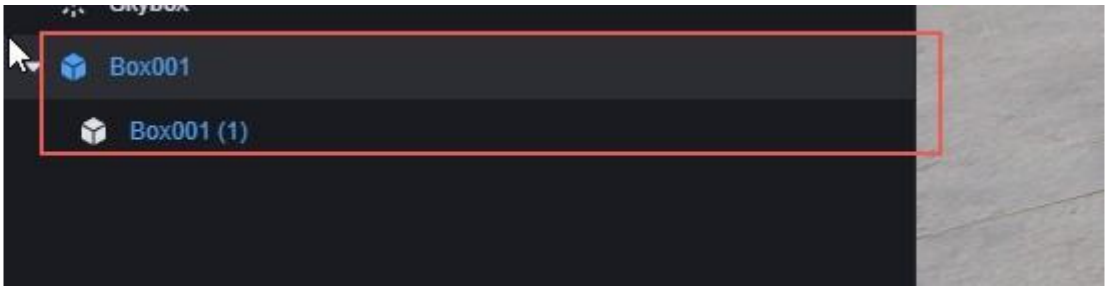
3. Drag the modified objects from the hierarchy screen into the created prefabs folder.



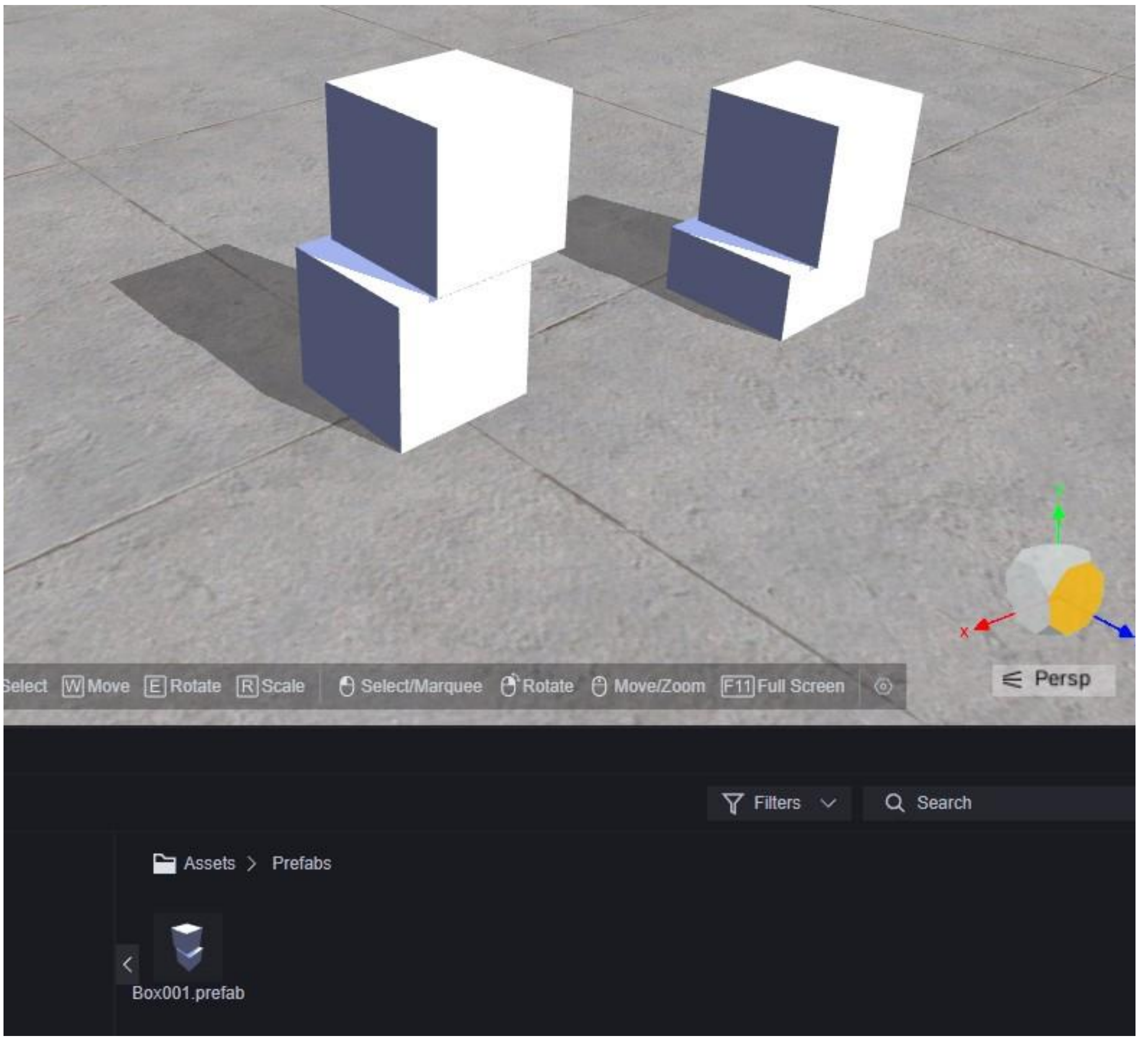


> Multiple prefabs can be created for the same object, but only the most recently created prefab will be used as the prefab for that object.

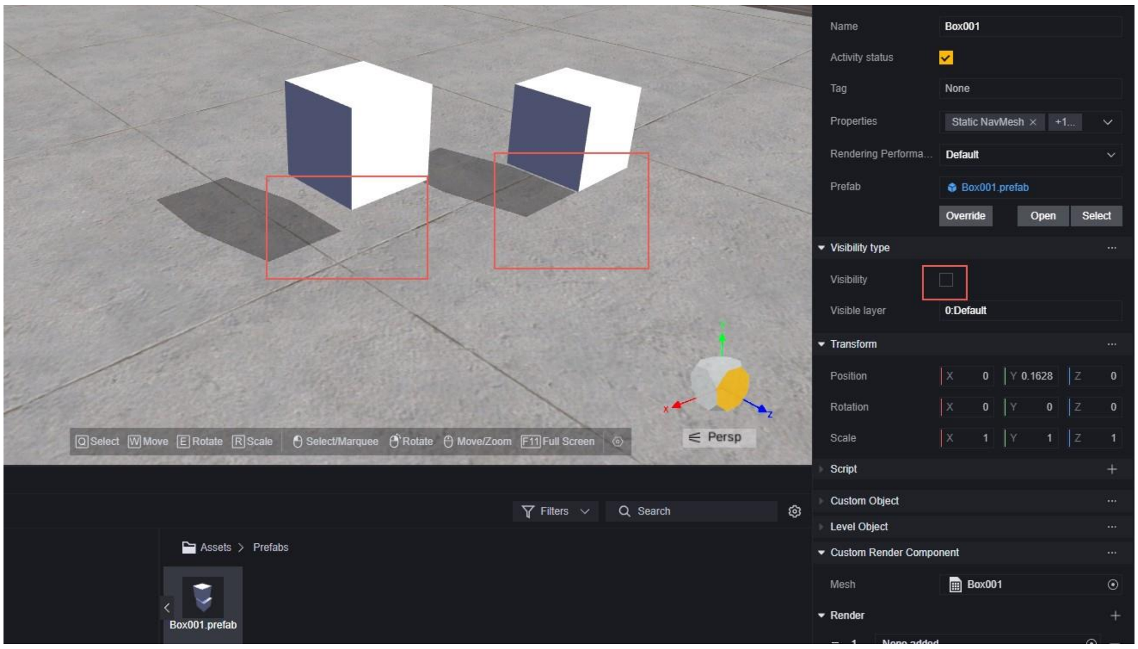
4. Objects with prefabricated bodies are marked blue in the hierarchy.



5. Dragging and dropping a prefab from an asset file onto the scene creates an identical object based on it.



6. Editing a prefab is synchronised to all objects created from it. You can quickly edit a prefab by clicking on it in the Inspector window, or double-click on it to access the detailed editing screen.



Name	Box001
Activity status	✓
Tag	None
Properties	Static NavMesh × +1...
Rendering Performa...	Default
Prefab	Box001.prefab

▾ Visibility type

Visibility	☐
Visible layer	0:Default

▾ Transform

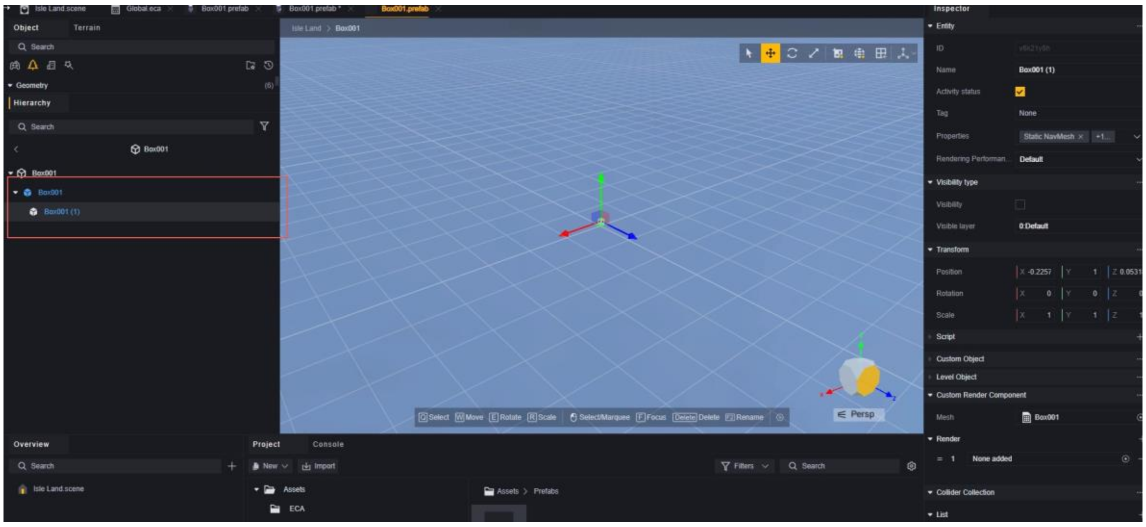
Position	X 0	Y 0.1628	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Script
Custom Object
Level Object
▾ Custom Render Component

Mesh	Box001

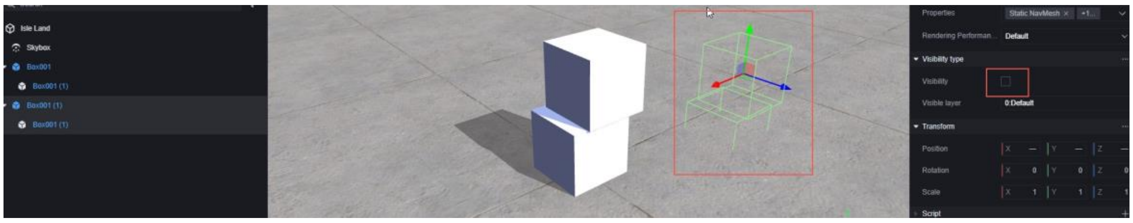
▾ Render

Here only the visibility of the main object is switched off, this is because in the Quick Manipulation screen only the cube that is the parent object in the structure is edited. Complex editing can be done by double clicking on the prefabricated body to access the detail screen.

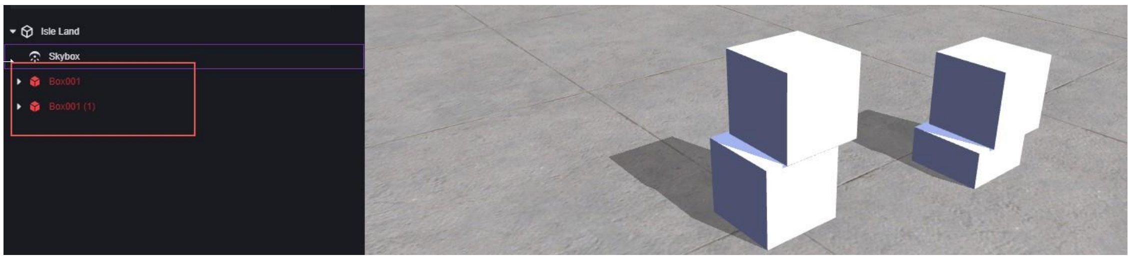


After opening the detail screen, you can edit all the structural bodies of the prefabricated body.

7. Editing an object does not affect the precast body or other objects with the same precast body.



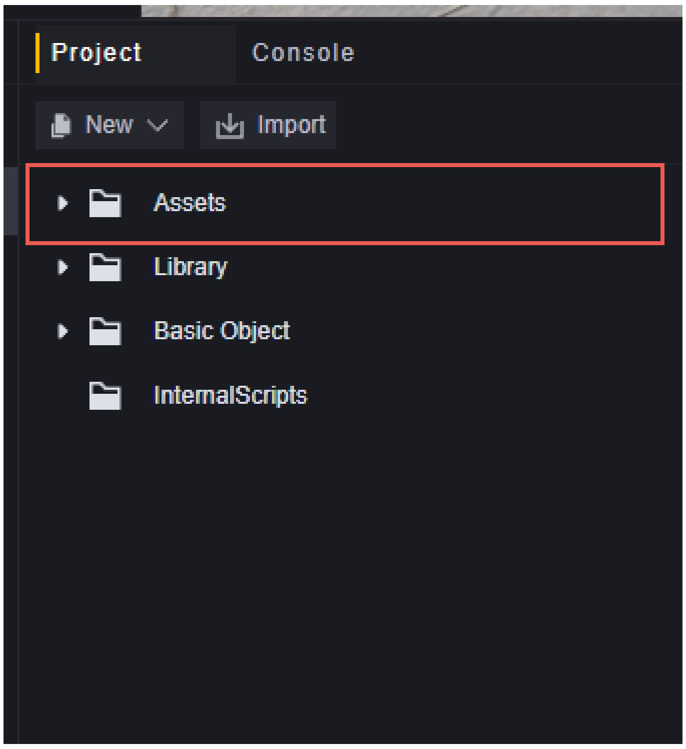
8. Deleting a prefab does not affect objects already created, but objects created from this prefab are marked in red in the hierarchy window.



liabilities

Assets are files used in the project to create game content. Resources must be of a type supported by the editor. Creation within the editor or importing from outside is supported, as well as exporting and using as assets for other projects.

Assets can be confirmed in the project screen, only the contents of the Assets folder are supported for customisation.



Supported asset types

Asset type	descriptions	Operation Introduction
Materia l Files	Materials can change the appearance of an object and are always used through shaders.	[There should be a link here]
physical material	Physical materials can change the physical properties of an object, such as elasticity and friction.	[There should be a link here]
Scene file	Scenarios are interfaces that process game content and contain resources for all or part of the game. For more complex games, you may need multiple scenarios to implement the design.	[There should be a link here]
UI files	A UI is an interface used for user operation or displaying information for the user.	[There should be a link here]
Player Data	Player Data can customise the player character, including its appearance, movements, etc.	[There should be a link here]
Animati on Controll er	The Animation Controller allows you to schedule and maintain a set of animated transitions for a character or object, setting up the playback of each animation clip.	[There should be a link here]
Behavi oural tree files	The Behaviour Tree file allows you to set the behaviour of a character or object.	[There should be a chain here.] Pick up] - The paperwork is done.
Script Related	Scripts can be written to add custom logic to the game, either by means of graphical elements or code.	[There should be a link here]