# Game | Usage of Dummies

## Overview

Dummies (aka anchors or pivots) define special points or volumes in relation to the model. They are specified on the ModelComponentDefinition as a List of ModelDummy classes.

Here is a visual example of Dummies in action:


On the picture you can see a Gatling Block that has multiple dummies specified:

- It has one at its muzzle which is used by the projectile system to spawn bullets at the correct position.
- It has 2 more dummies at its conveyor ports that allow the user to interact with its inventory. They also have another purpose - they tell the conveyor system that there can be a connection in this area.

## ModelDummy Class

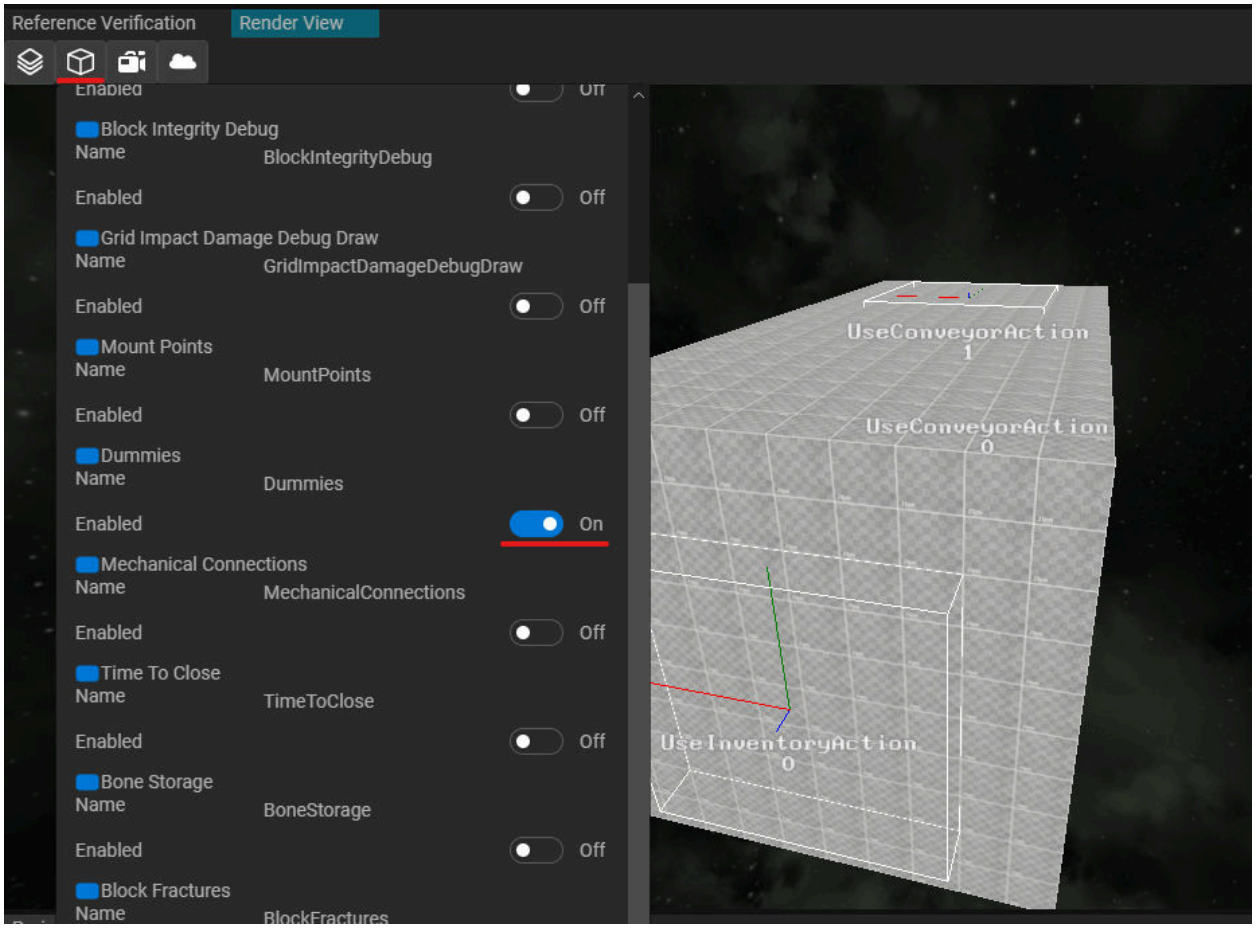
The ModelDummy class itself is documented, but this chapter dives deeper into the meaning of each field.

- Name - string
    - The name of the Dummy. You can call the Dummy however you want.
- Transform and Scale - Integers

    - Representation of the position, rotation, and scale of the dummy. Entered using local coordinates
- Priority - hierarchical
    - Priority for interaction system in the ascending order: the dummy at the top of the list  is the highest priority. The priority is used in cases where there might be multiple dummies occupying the same position.
- Type- Selection
    - Denotes type of dummy which is represented here: Types of Dummy
- SupportingFracture - String
    - Specifies which fracture of the block supports the dummy. If a fracture is broken, the Dummy is disabled. For more information about fractures please refer to the Breakable Block with Fractures chapter of the block addition document.

# Debug Draw

You can visualize the dummies in the Editor by turning on the Dummies debug draw:




# Types of Dummy

## ConveyorSystemConfiguration

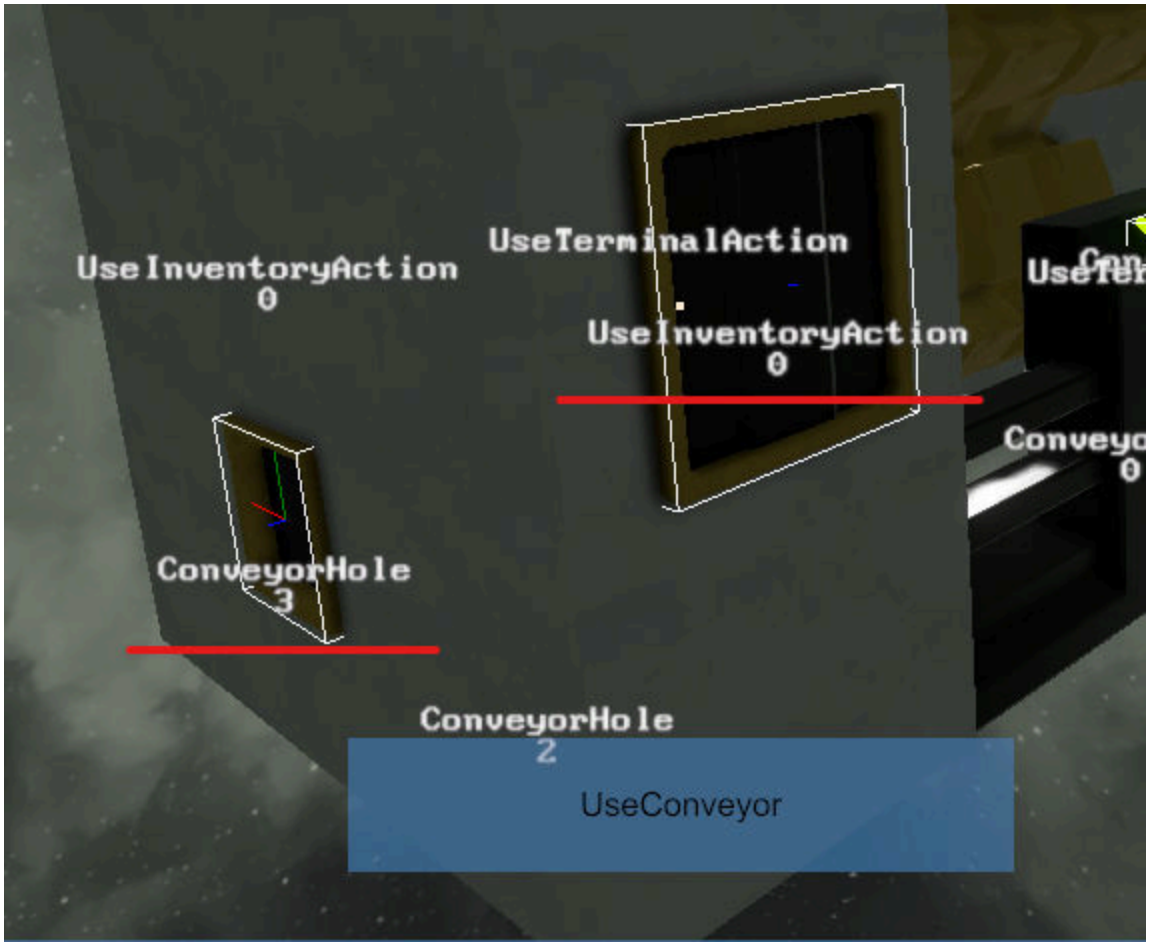
The ConveyorSystemConfiguration contains mapping that specifies which Dummy types should count as a conveyor port. Large conveyor ports serve a double purpose for being both an interaction point and a port while small ports do not have the interaction point.

### ConveyorHole

This is used to define conveyors that operate as ports only

### UseConveyorAction

See: UseConveyorAction




## UseAction Types

The UseAction class is connected to Dummies that are used by the interaction system and allow the user to perform actions like entering the cockpit or interacting with a conveyor port. This is achieved via child classes that are each made for a specific purpose. The most important will be described in detail in the following chapters. You can find the full list in the Interaction namespace.

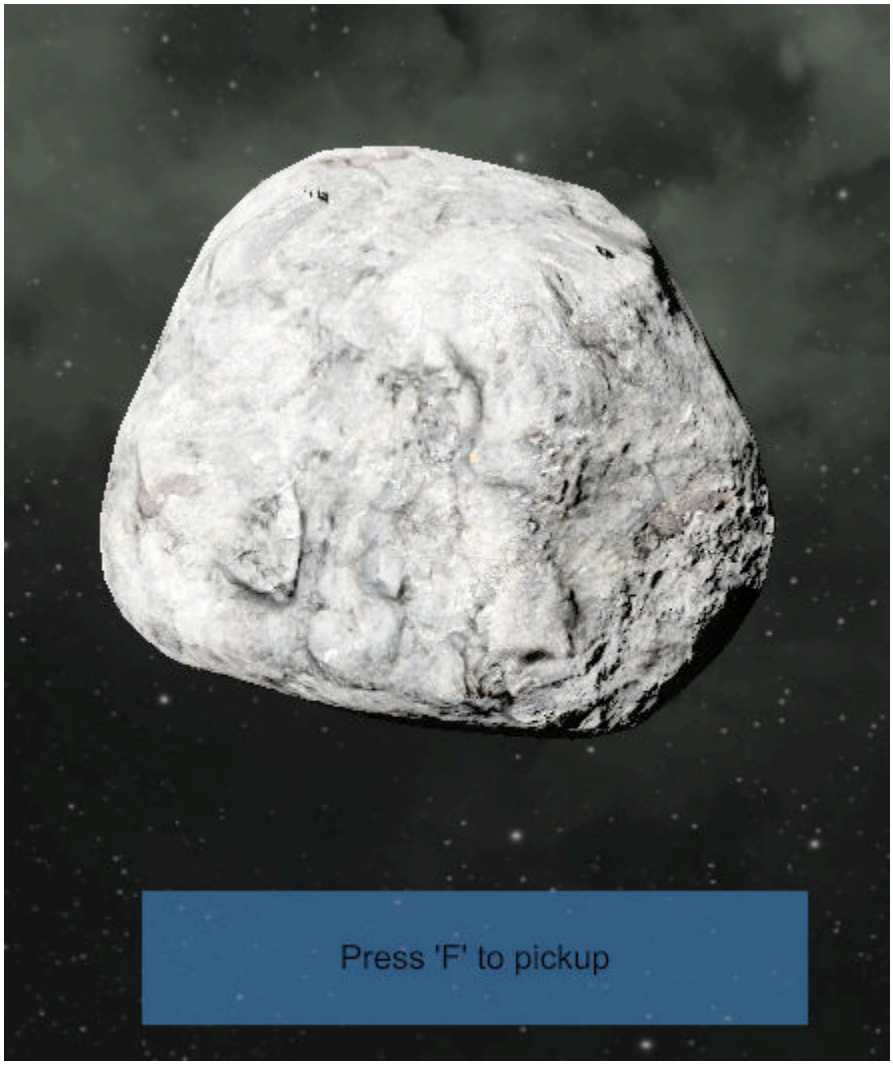
Note that while usually the action name will correspond to its class name, this might not always be the case. The pairings between the action name and the given Type can be found in UseActionBindingsDefinition. At the time of writing this documentation the pairings are:

```
{ "UseAction", UseAction },
{ "UseTerminalAction", UseTerminalAction },
{ "UsePilotableAction", UsePilotableAction },
```

```
{ "UsePowerableAction", UsePowerableAction },
{ "UsePickupAction", UsePickupAction },
{ "UseConveyorAction", UseConveyorAction },
{ "UseInventoryAction", UseConveyorAction }
```

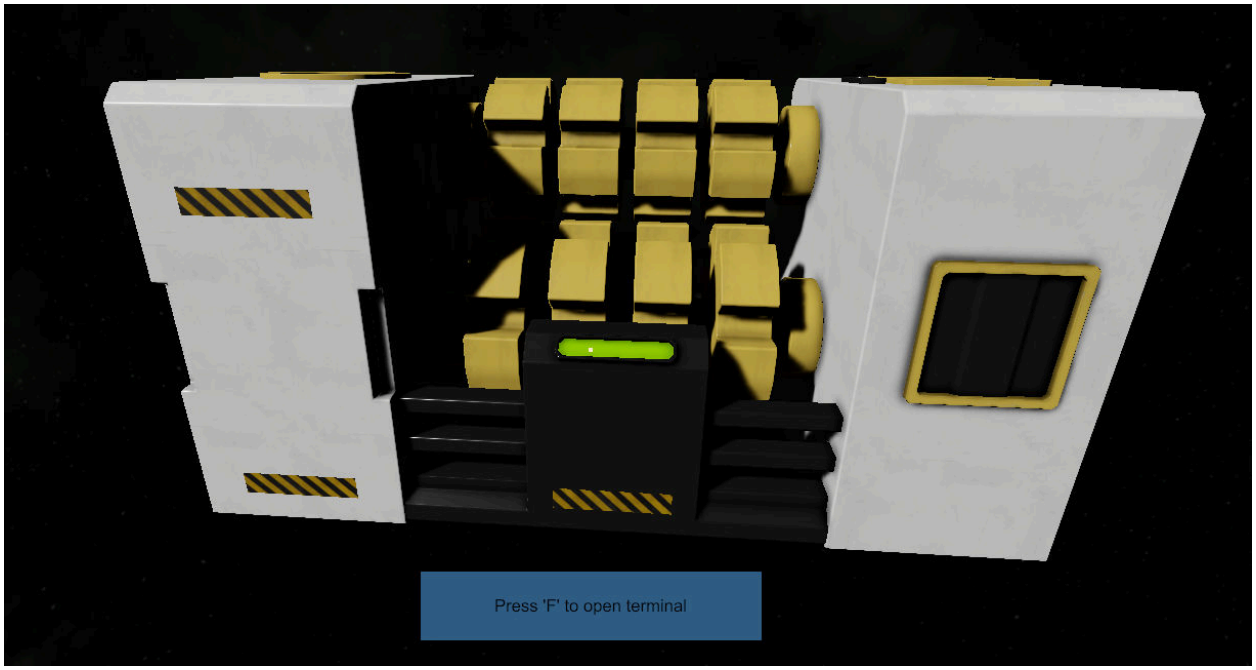
### UsePickupAction

This action indicates that the object can be picked up by the player. Used by floating objects like ores:



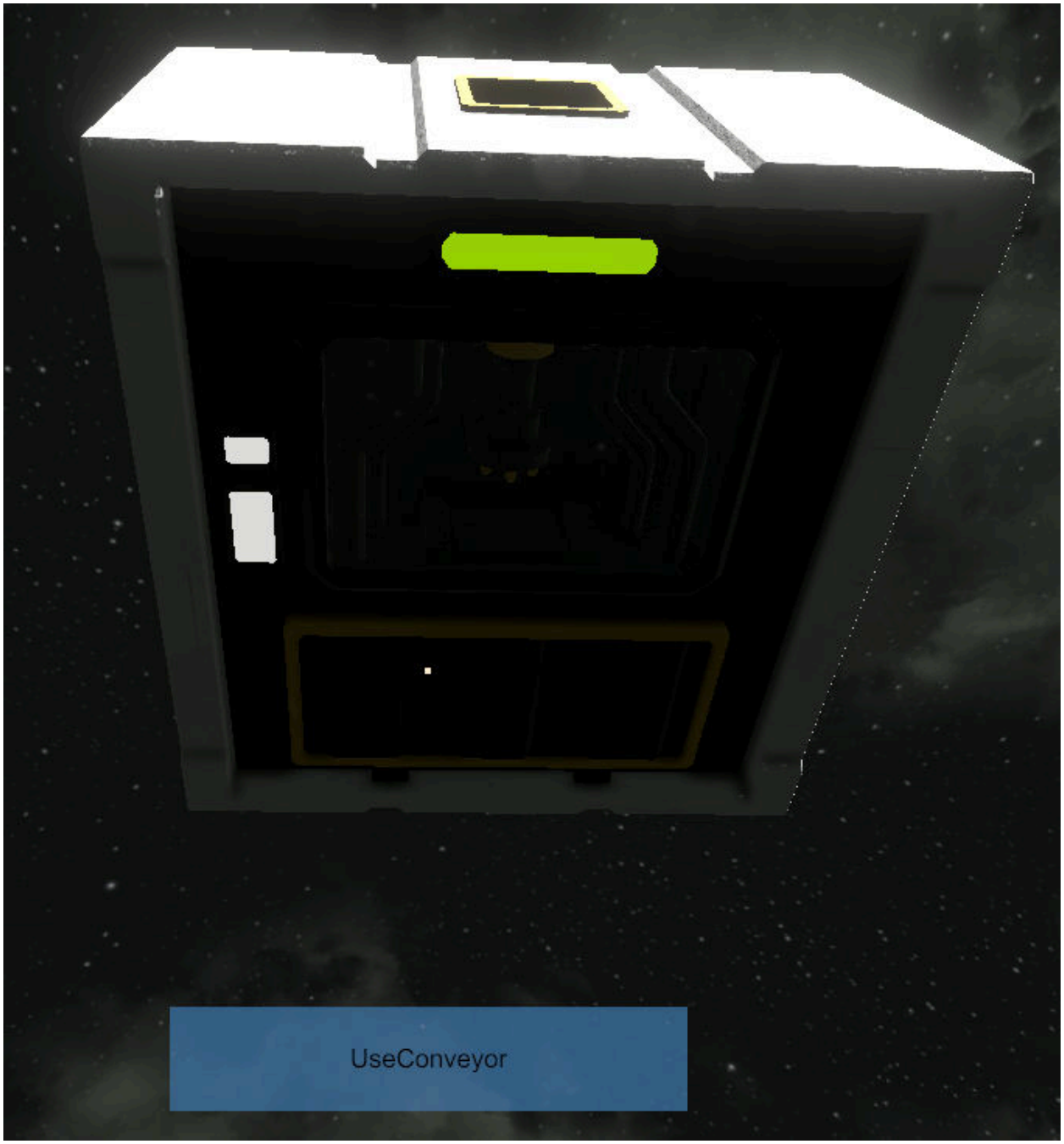

### UseTerminalAction

Opens a full terminal with Control Panel as the focused tab. An example is the Refinery:




### UseConveyorAction

Opens terminal with Inventory as the focused tab. This dummy should be used in cases where you want the user to be able to interact with the inventory AND for the dummy to function as a conveyor port too. For example on an Assembler:

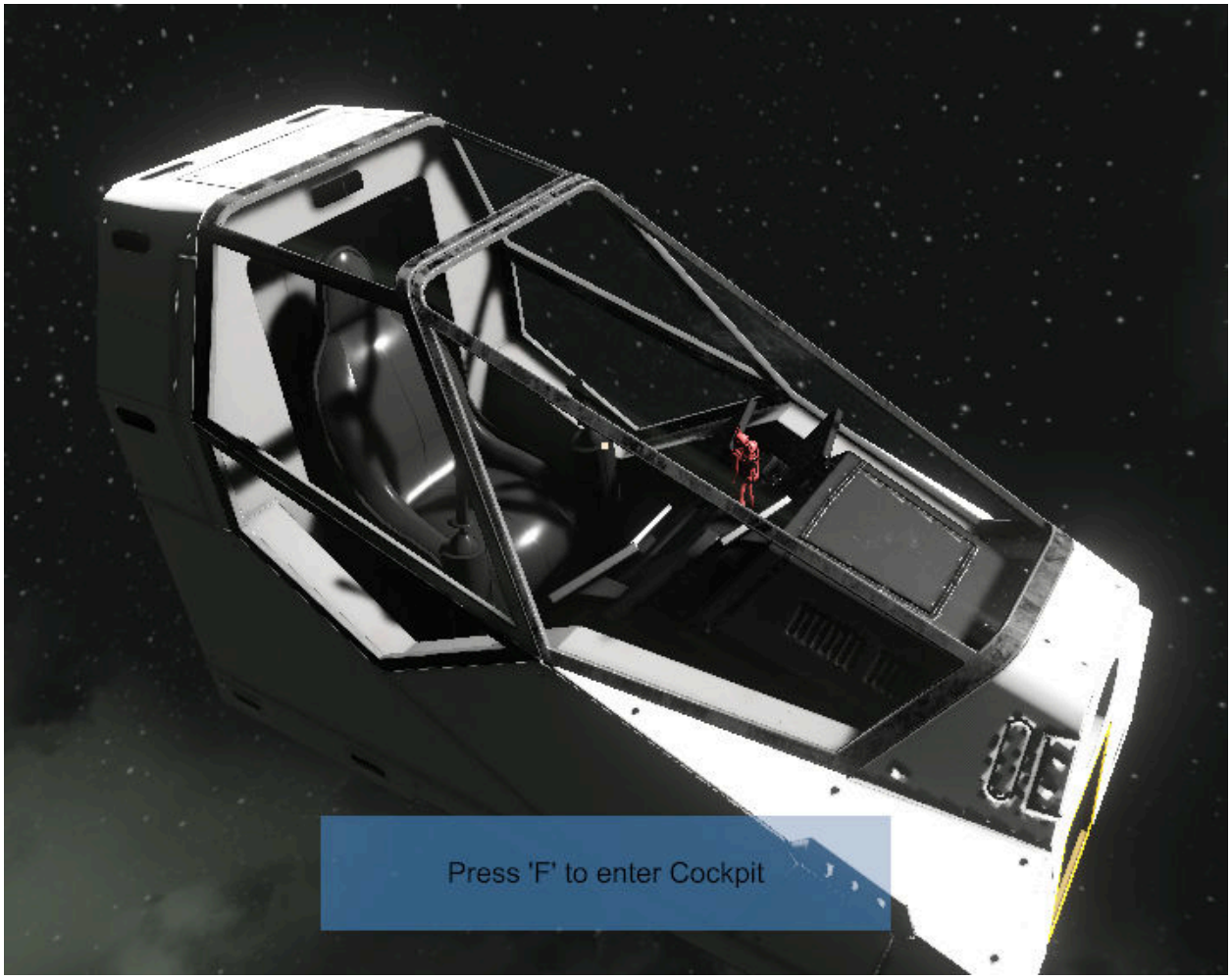



### UseInventoryAction

This dummy should be used in places where you ONLY want the user to be able to interact directly with the block's inventory. The dummy will not work as a conveyor port.

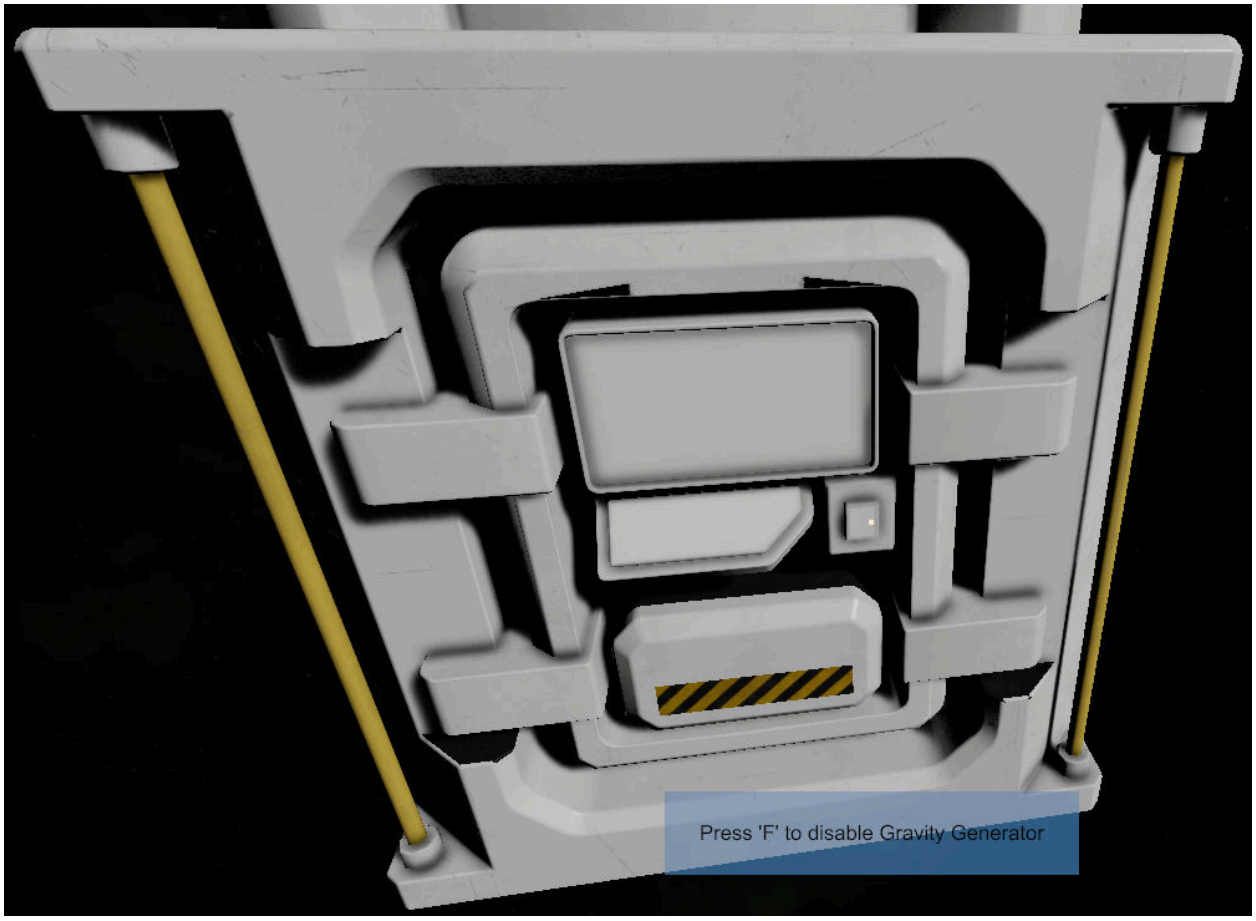
### UsePilotableAction

Allows entering a pilotable object, for example a cockpit:




### UsePowerableAction

Can be used on blocks that have a PowerableBlockComponent to turn the block on and off. One example is the Gravity Generator Block:

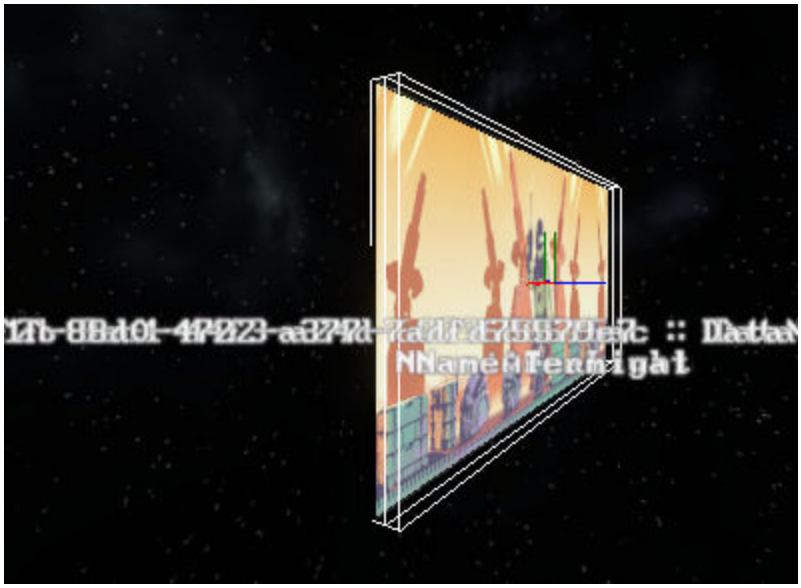



## Lights

A collection of dummies that provide different types of illumination.

### AreaLight

This is used to define the size, origin and direction of light generated by an LCD Block.

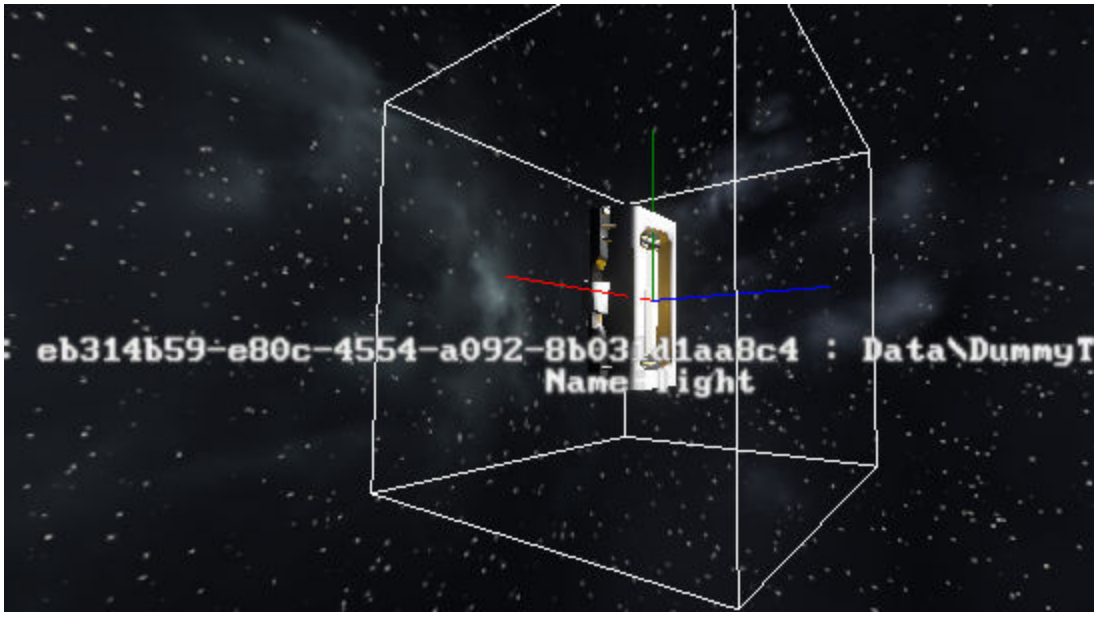


## CapsuleLight

This used to define the size, and origin of lights generated by Neon Tubes blocks.

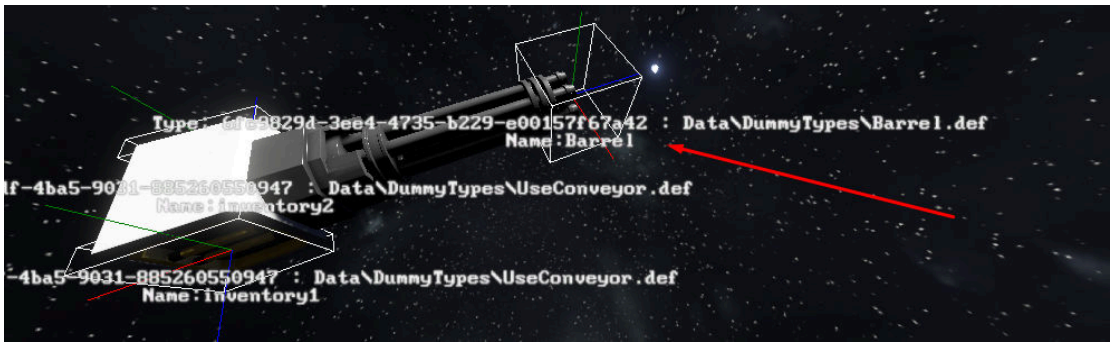

## Light

This is used to define the point of origin and direction of a light source (e.g Interior Light/ SpotLight)



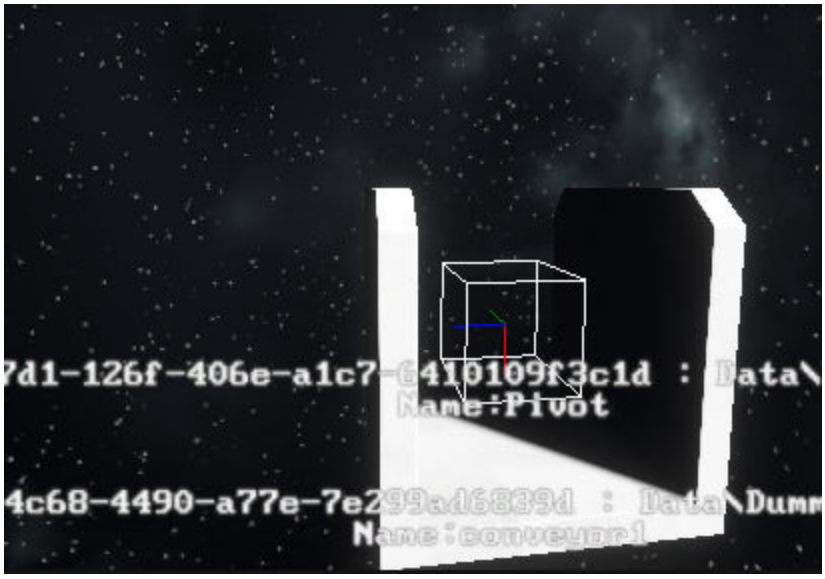

## Barrel

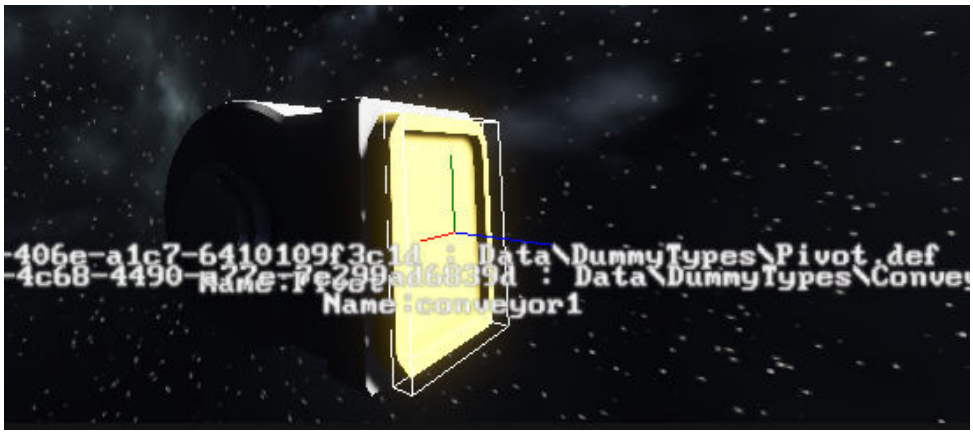
This is used to define the location and direction projectiles should start from and travel.




## Pivot

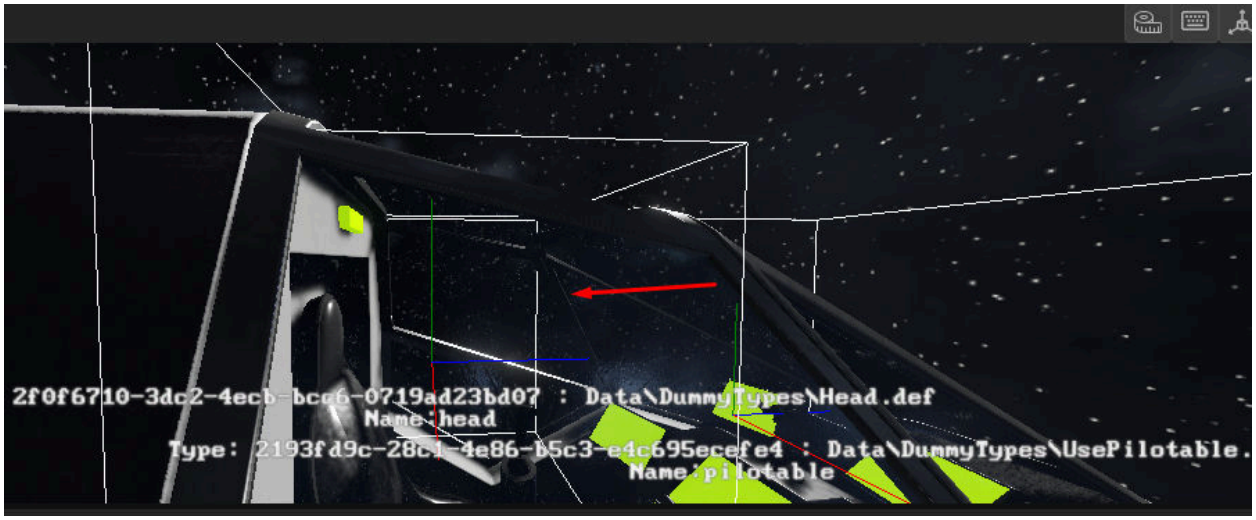
This is used to define the starting point and direction of travel for mechanical block top parts. A pivot has to be placed on both the base and the head for it to work.




## Head

This is used to define the 1st person camera position, direction and orientation when entering a seat.




## Pumps

Specific dummy types used with pump blocks to either collect or discharge water voxels.

### Pumping

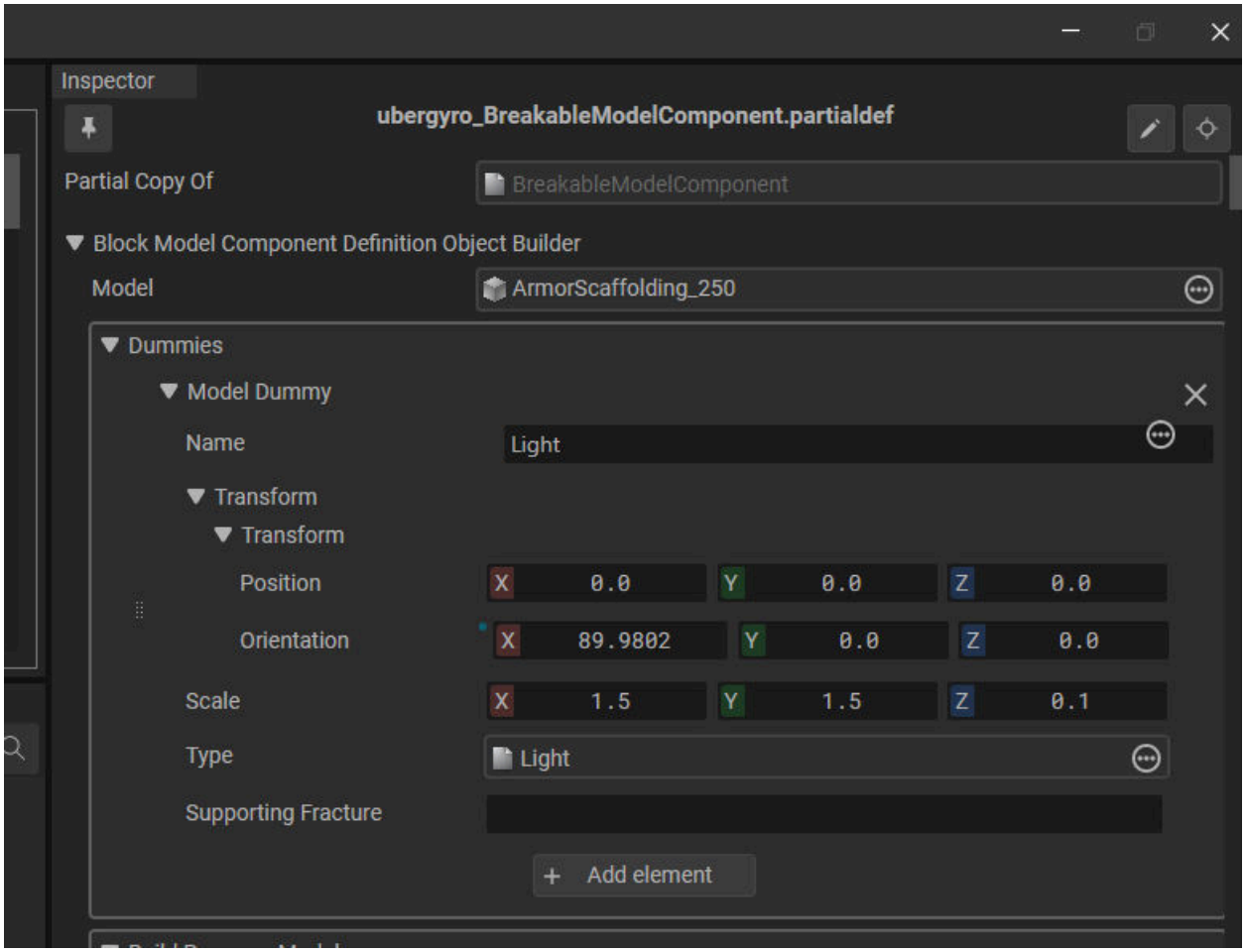
This defines the area on a block where water voxels will be collected and converted into non-tangible resource within the grid system.


### Dumping

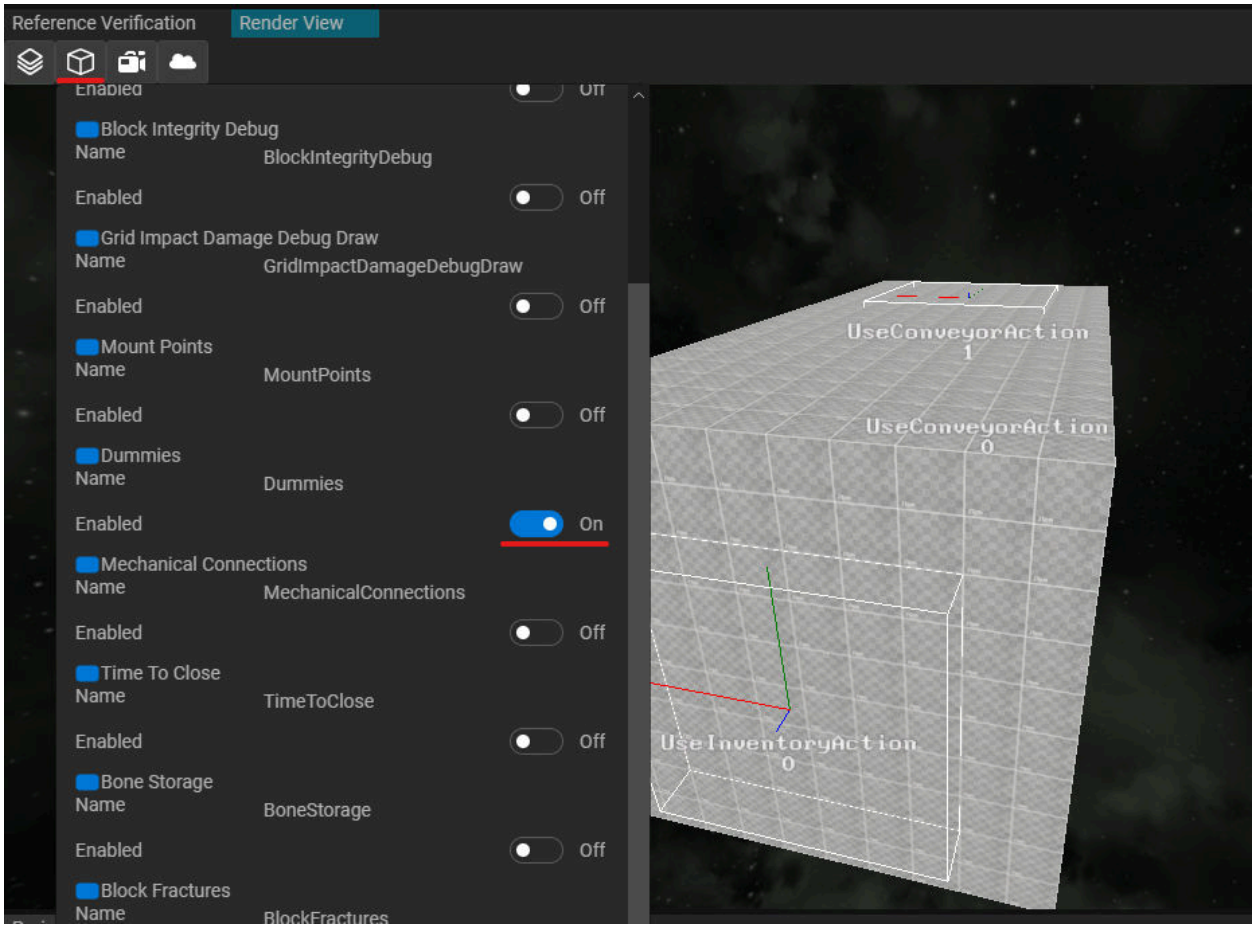
This defines the area on a block where a water resource will be discharged from the grid system and converted into physical water voxels.


## Setting up Dummies

They are specified in a list on the Block Model. You have to specify the dummy name and its position, rotation, and scale in local coordinates.  (**Note: Z+ on dummies are the front not Z-**)




You can visualize the dummies in the Editor by turning on the Dummies debug draw:




If you need multiple dummies that occupy the same area, you can set their priority. This is done by the order they are in the list. (higher up the list, higher the priority) If the dummy is used for player interaction, you also need to select the dummy with the associated action.