East Fighter Su33

Project Structure

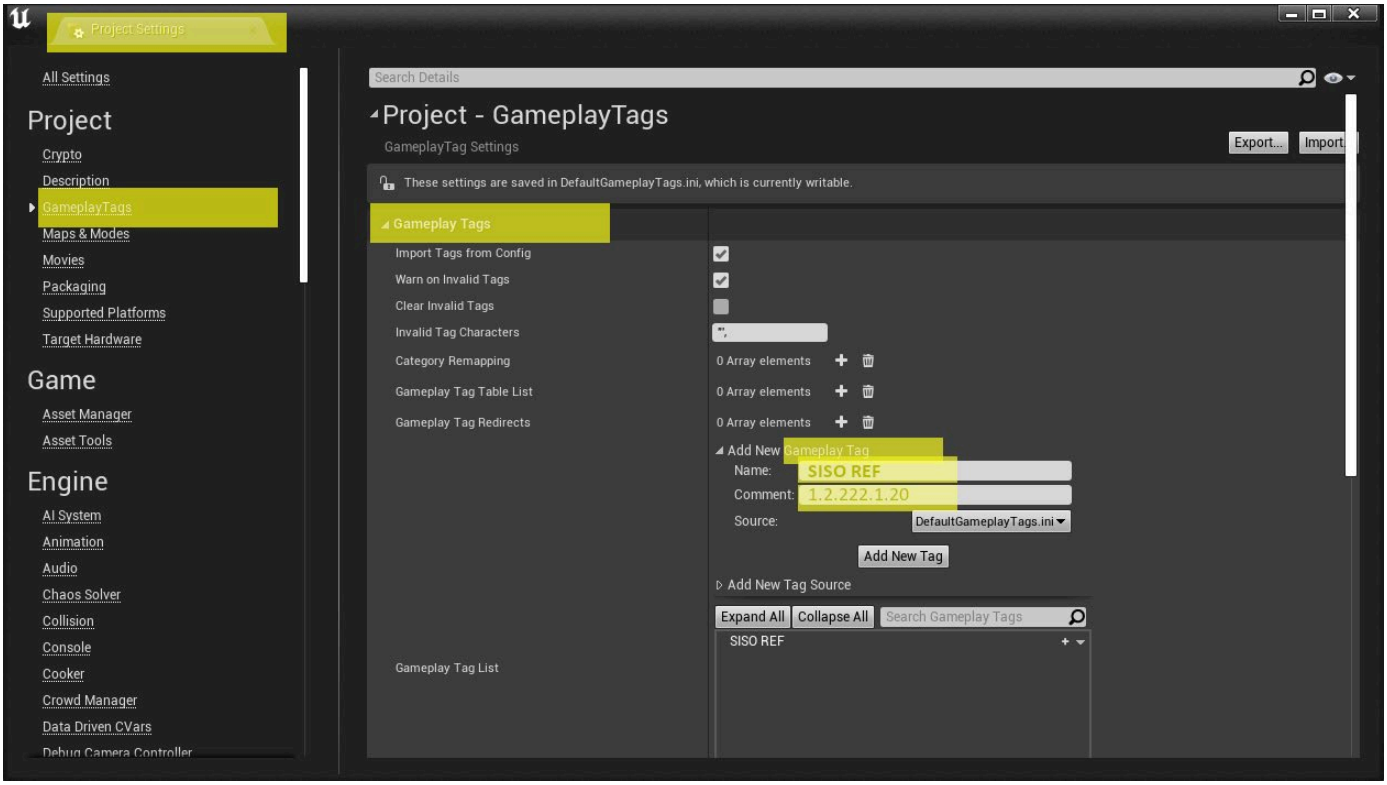
The Vigilante® Fighter Su33 asset pack is rigged and ready to go right from the marketplace! This stand-alone Fighter Su33 is DIS/HLA (RPR FOM) Integration ready for import into your network replicated simulation projects. All inputs and settings have been pre-configured with default settings.

This Asset has been cleaned for import and migration between projects. Instructions explaining how to install the asset along with the setup illustrations are shown below.

SISO Identification

This vehicle is compliant with SISO Entity Reference No.: **1.2.222.1.20**

The SISO REF Number has been added to the Project Settings under **GameplayTags**. The Tag Name is SISO REF, and the Tag Comment includes the SISO Entity Reference No. listed above.



License

Licensed for Use Only with Unreal Engine-based Products.

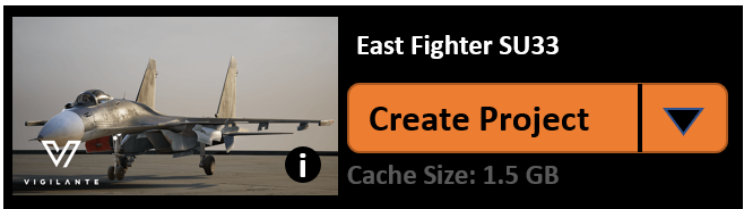
New to Unreal Engine?

Helpful resources for those who may be new to using Unreal Engine:

- https://docs.unrealengine.com/4.27/en-US/Basics/
- https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/
- https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/

Installation

To install the vehicle from the Epic Games launcher, simply click, Create Project:



Once opened, the project is in an “Content” Folder. Subfolders include folders for the vehicles, along with a shared folder. The shared folder consists of assets to be shared by all vehicles, from the master vehicle parent blueprint to common FX shared by all Vigilante vehicles.

Project Folder Structure

Once the .Zip file has been extracted and migrated into your project, the Vigilante folder structure will be as follows:

```
├── Config
├── Content
│   └── VigilanteContent
│       └── Shared
│       └── Vehicles
│           └── East_Fighter_Su33
│               └── Damaged
│                   └── Material
│                       └── Textures
│               └── Anim
│               └── FX
│               └── Level
│               └── Materials
│                   └── Textures
├── [YourProjectName].uproject
```

If you have installed multiple Vigilante assets from the content library, your folder structure might look this:

```
├── Config
├── Content
│   └── VigilanteContent
│       └── Shared
│       └── Vehicles
│           └── East_Fighter_Su33
│           └── West_APC_Stryker
│           └── East_IMV_TigrM
├── [YourProjectName].uproject
```

This folder configuration will let you take advantage of resource sharing across assets from the Vigilante Content Library. (e.g., shared particle effects like dust, smoke, fire, etc.)

Vigilante Customer Support

Vigilante customer support can be reached through our uemarketplace@vigilante.us support team. Please reach out with any questions regarding our product or asset configuration.

Asset Configuration Details

When you hit "Play" in the Unreal Project, a User Interface will appear on the right-side of your screen that allows you to manipulate the options available for configuration.

This asset comes with two separate Actor blueprints. The Blueprint with the "_Showcase" suffix includes a Demo User Interface with sliders and toggles for the various vehicle functions. (e.g., Open / close hatches, fire weapon, etc.). This is designed for you to experiment with how the various components of the asset function and are hooked-up.

To use our **East_Fighter_Su33** vehicle as an example, this project would include the following two Blueprints for you to interact with:

BP_East_Fighter_Su33- Functions for DIS / HLA compliant System Integration.

BP_East_Fighter_Su33_Showcase – Demo / Showcase controls for vehicle review

When you feel like you have a good understanding of the Asset and are ready to Migrate it into your Unreal Project, pick a version of actor blueprint without the "_Showcase" suffix (located in the same folder), and migrate it into your project. This way you will migrate just the essentials needed for your use. (No User Interface with sliders or toggles, etc.)

Vehicle Materials

Material	File Name	Channel
MI_East_Fighter_SU33_Main		
Base Color	T_East_Fighter_Su33_D	RGB
Metallic	T_East_Fighter_Su33_ARM	B
Roughness	T_East_Fighter_Su33_ARM	G
Emissive	T_East_Fighter_Su33_E	RGB
Normal	T_East_Fighter_Su33_N	RGB
Ambient Occlusion	T_East_Fighter_Su33_ARM	R

Material	File Name	Channel
MI_East_Fighter_SU33_Glass		
Base Color	T_East_Fighter_Su33_Glass_D	RGB
Metallic	T_East_Fighter_Su33_Glass_ARM	B
Roughness	T_East_Fighter_Su33_Glass_ARM	G

Normal	T_East_Fighter_Su33_Glass_N	RGB
Ambient Occlusion	T_East_Fighter_Su33_Glass_ARM	R

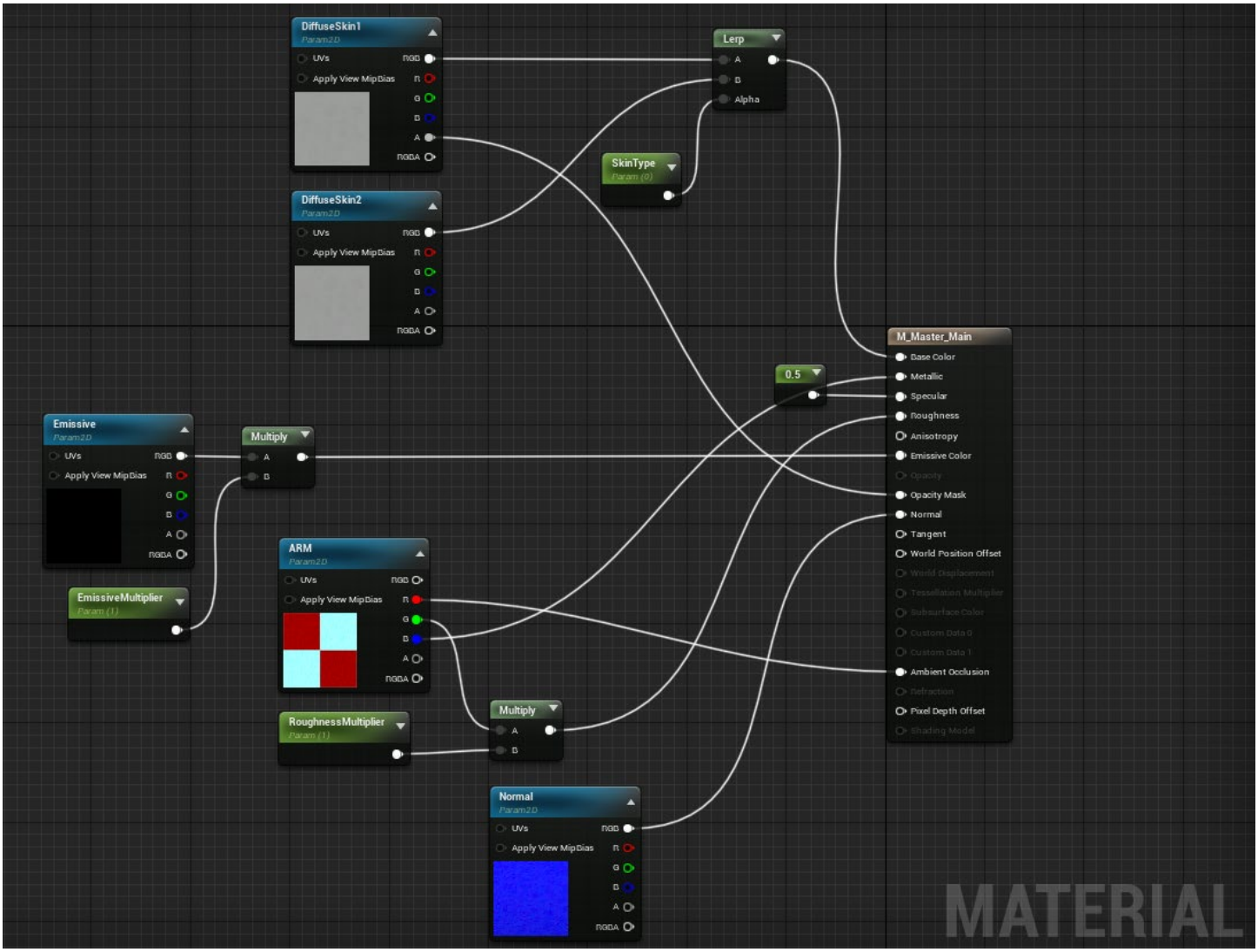
Vehicle Damaged Materials

Material	File Name	Channel
MI_East_Fighter_SU33_Damaged		
Base Color	T_East_Fighter_Su33_Damaged_D	RGB
Metallic	T_East_Fighter_Su33_Damaged_ARM	B
Roughness	T_East_Fighter_Su33_Damaged_ARM	G
Normal	T_East_Fighter_Su33_Damaged_N	RGB
Ambient Occlusion	T_East_Fighter_Su33_Damaged_ARM	R

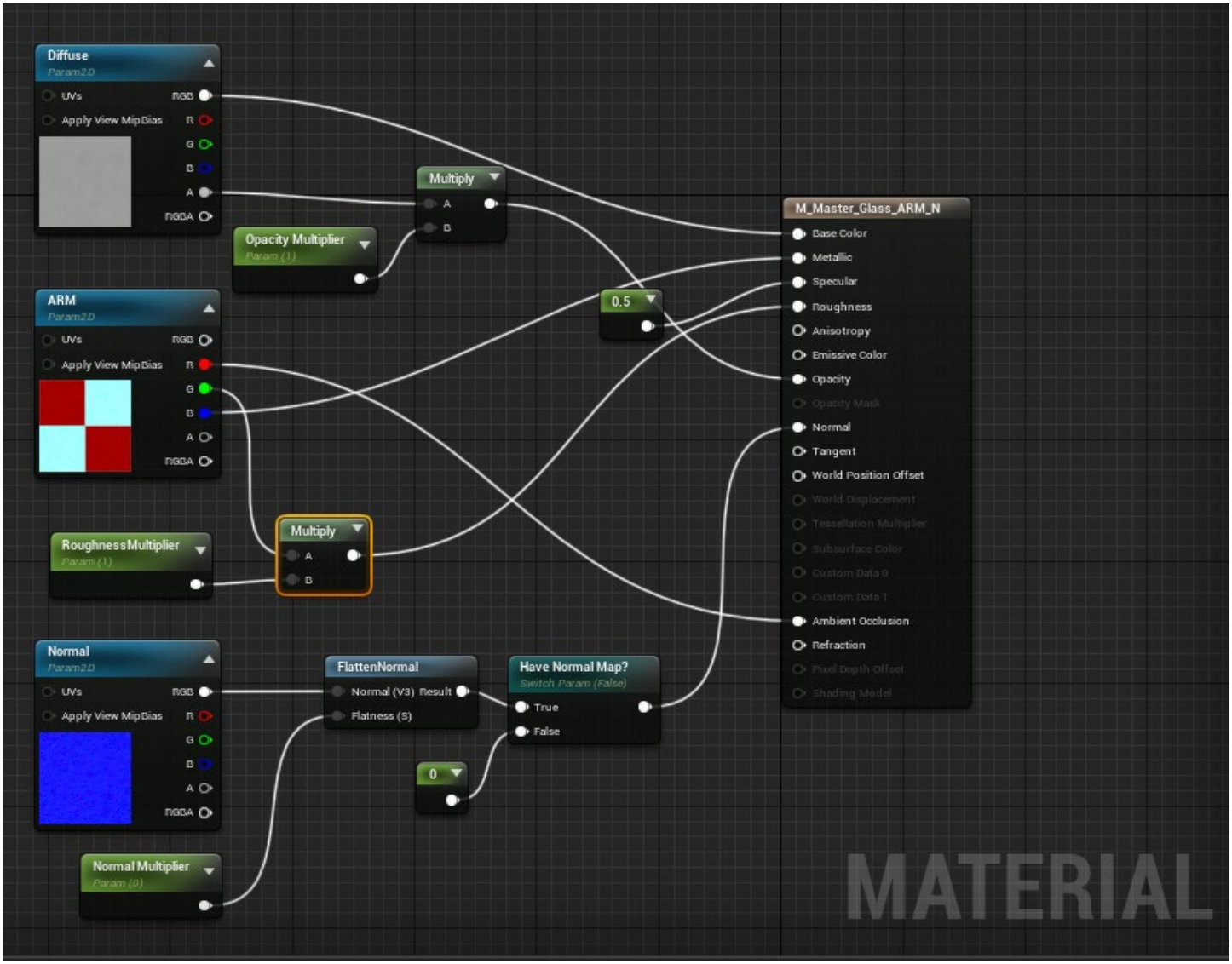
Vehicle Body Material Graph

The material and textures used with this Vigilante Content Library Asset use Material Channels as diagramed below. Please note the vehicle uses Material Instances for a cleaner and more convenient asset management. The vehicle base color texture (skin) is toggled using the **SkinType** parameter.

Texture emissivity is controlled using the **EmissiveMultiplier** parameter. The ARM texture (which stands for Ambient / Roughness / Metallic) RGB channels are mapped to the associated channels of the Vigilante Asset Material as shown below. (Red to Ambient, Green to Roughness, and Blue to Metallic)

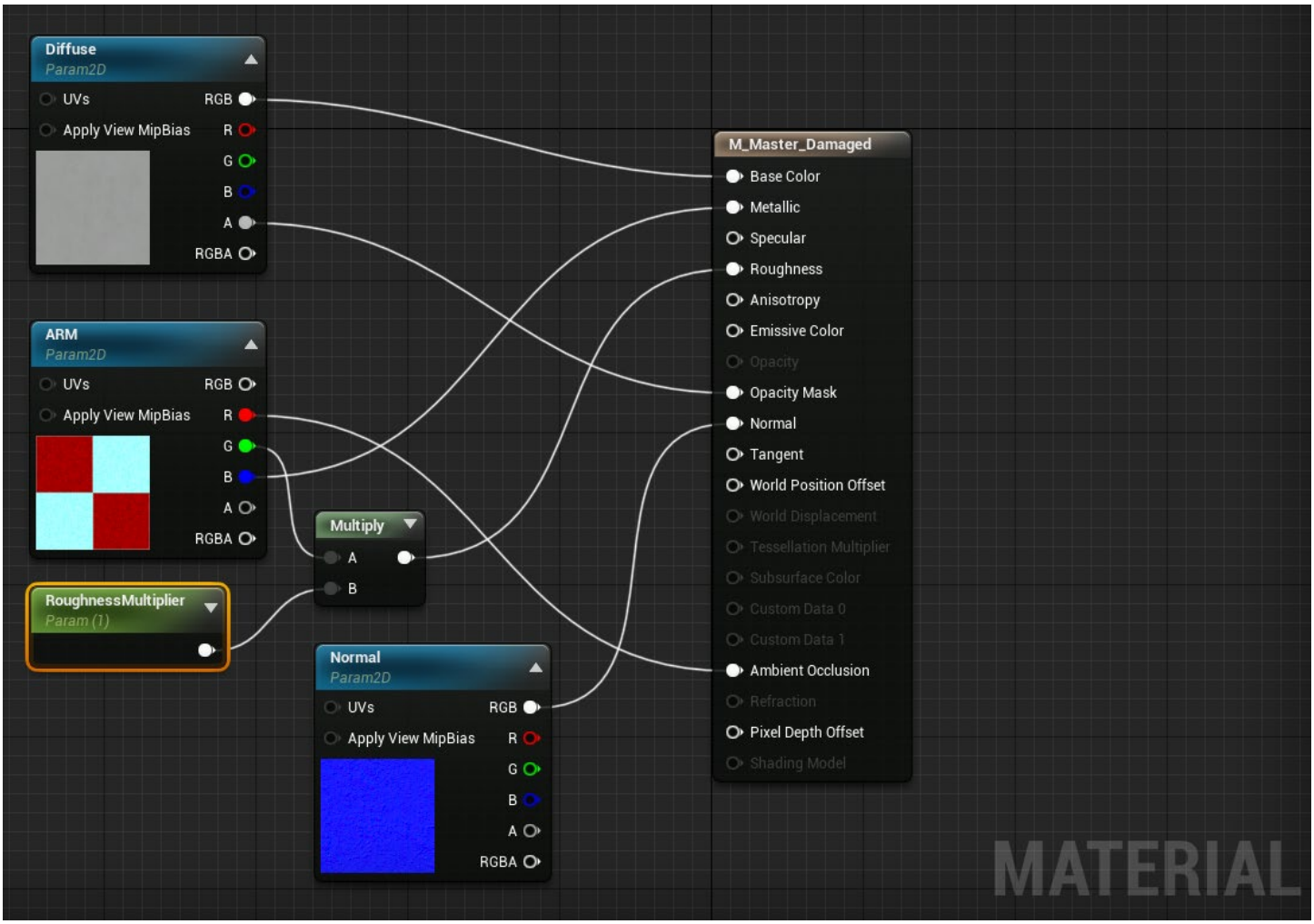


Vehicle Glass Material Graph



© Copyright Vigilante, LLC 2021 East Fighter Su33

Vehicle Damaged Body Material Graph

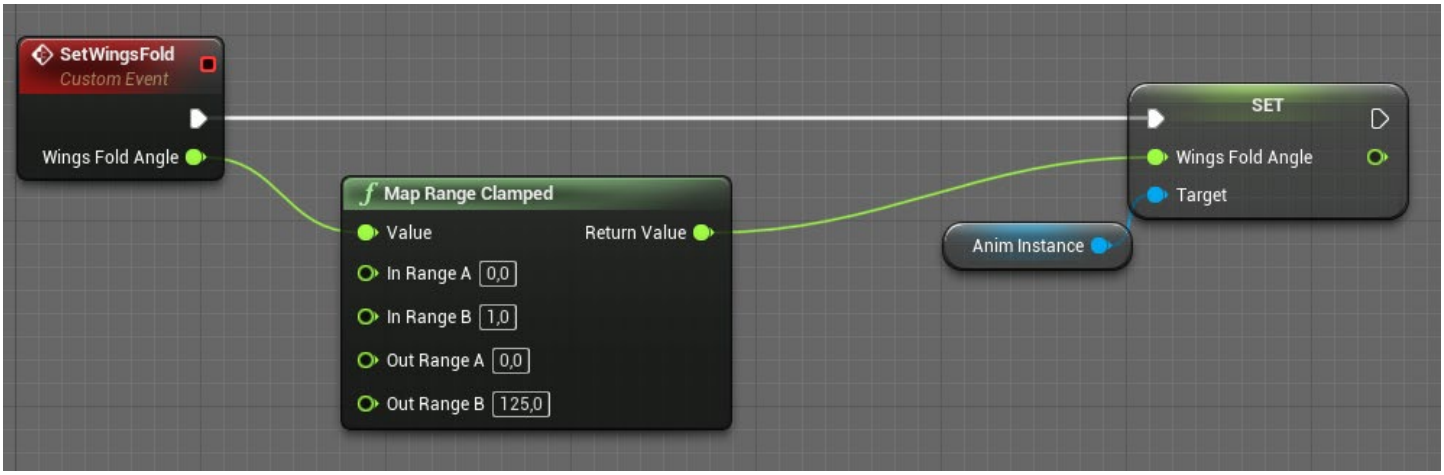


The Vigilante asset has a ‘damaged’ state. This state is triggered through the UI controls. Setting this to TRUE will cause both the mesh visibility to toggle to the damaged state and will begin the DamagedSmoke FX playing.

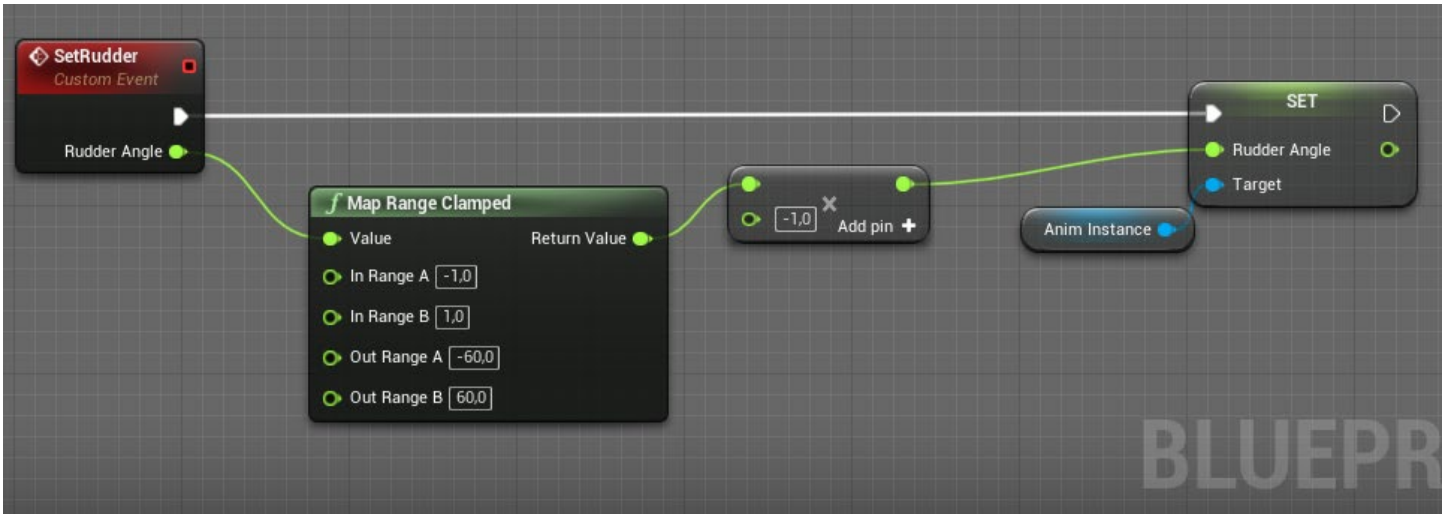
Blueprint functionality

All the logic is about applying changes either to the active status of the Vehicle parts (FX or damaged state), modifying the materials parameters, or modifying the Animation Blueprint parameters.

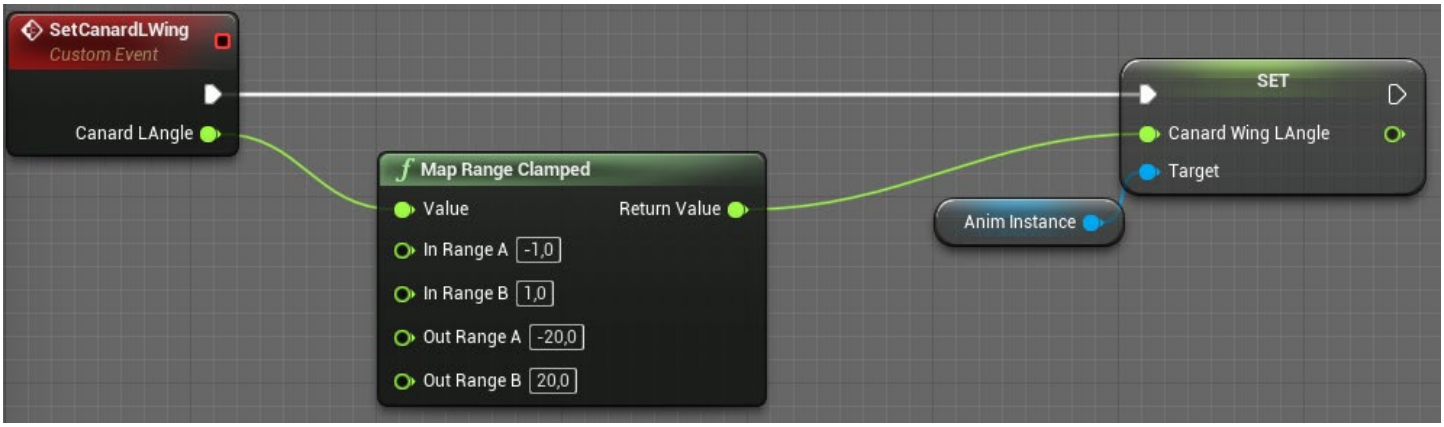
Control Wings Fold



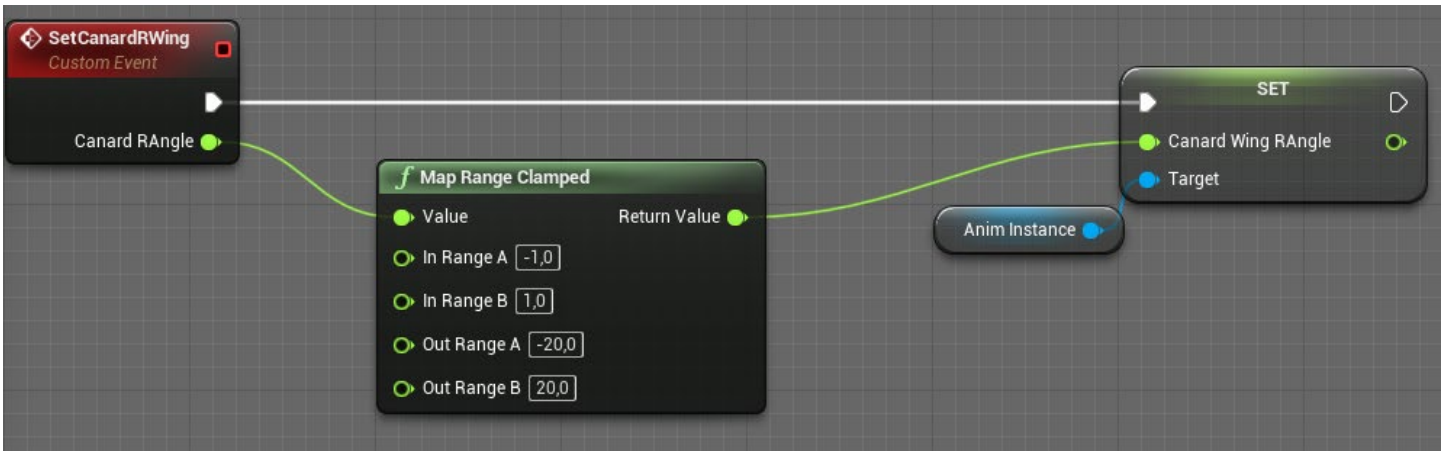
Control Rudder



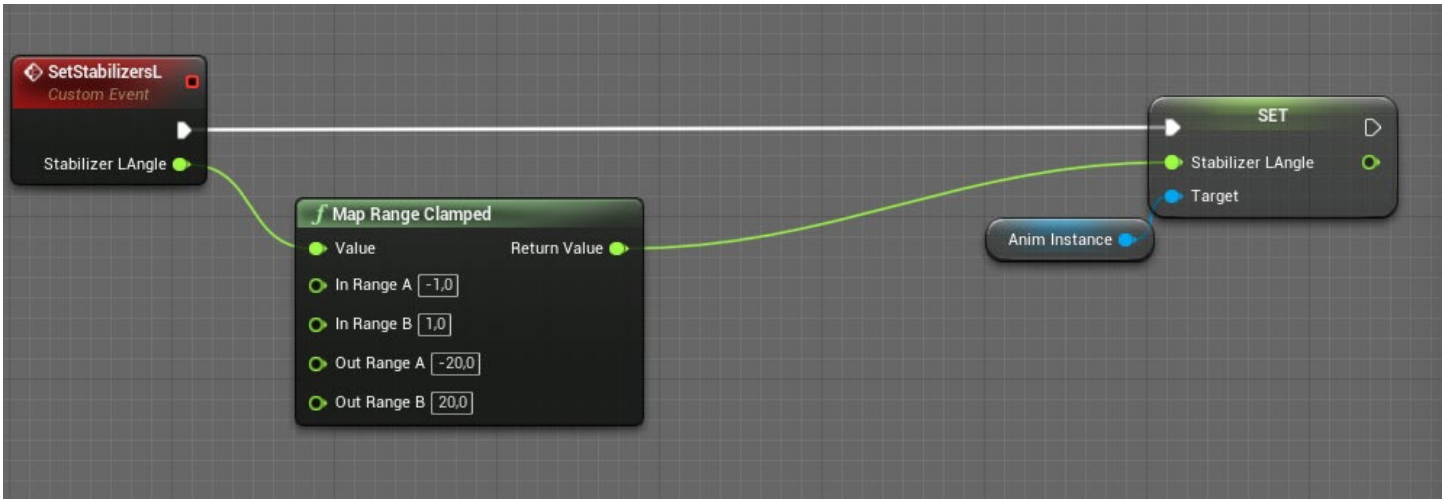
Control Canard Wings

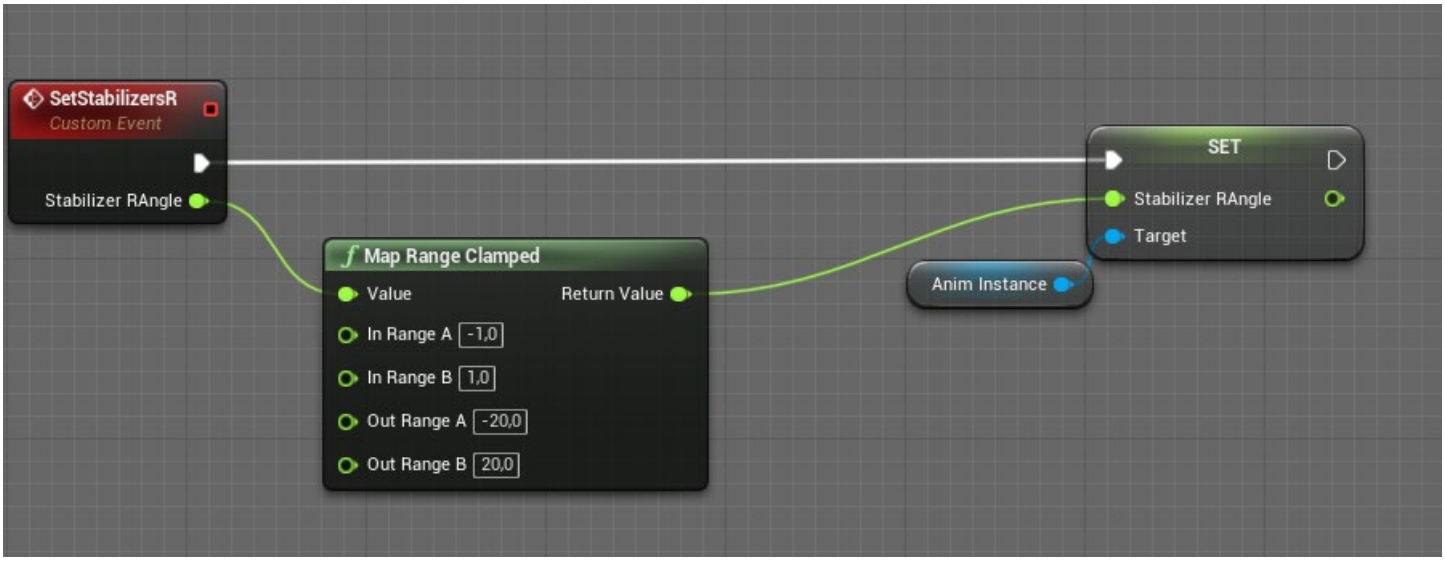


© Copyright Vigilante, LLC 2021 East Fighter Su33

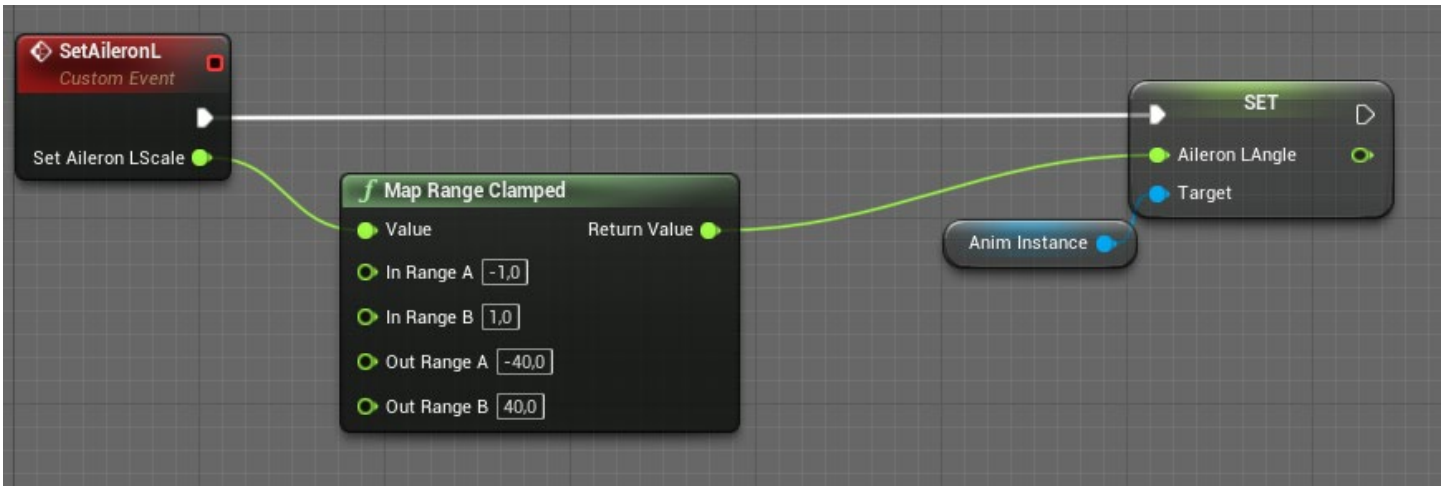


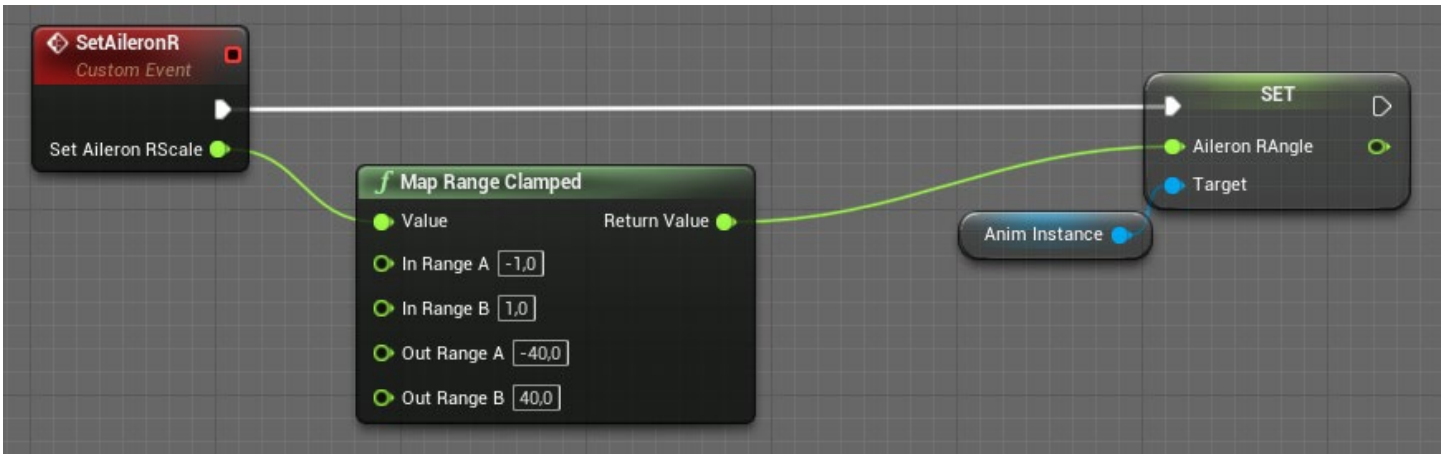
Control Stabilizers



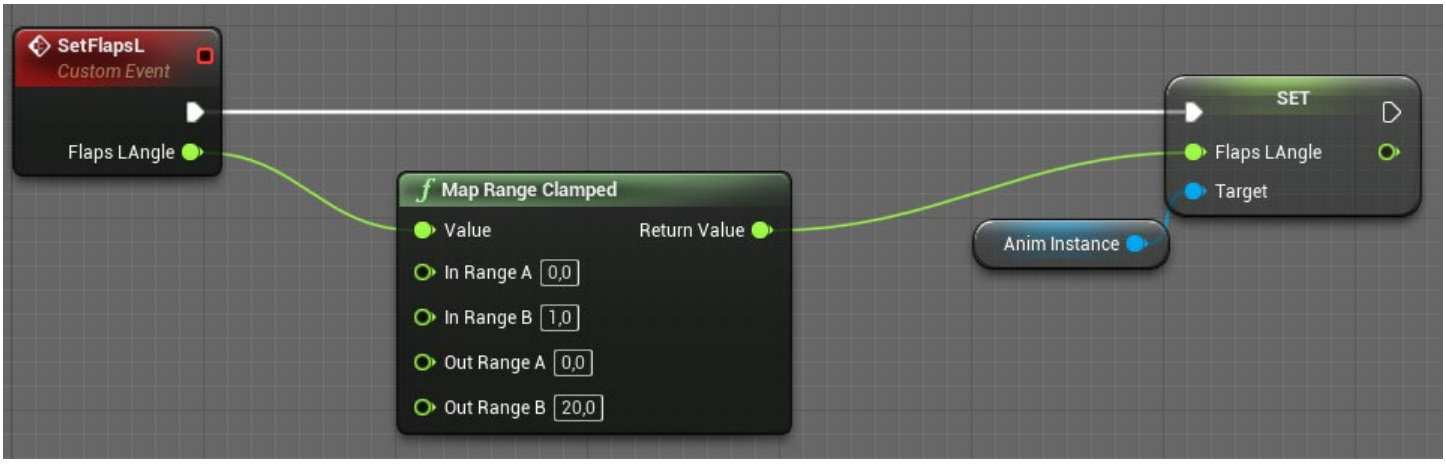


Control Ailerons

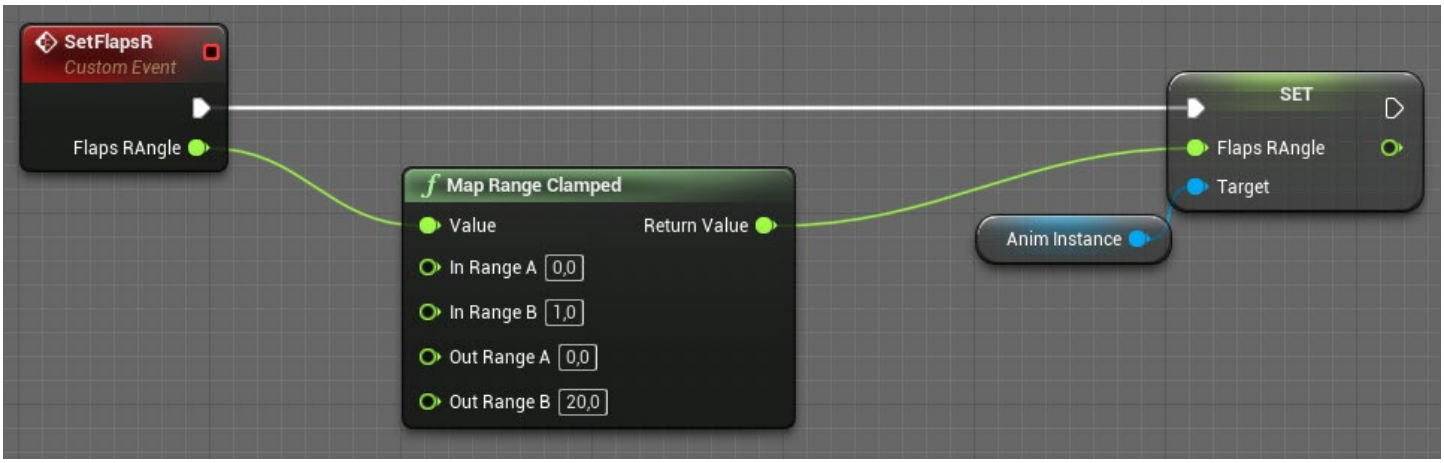




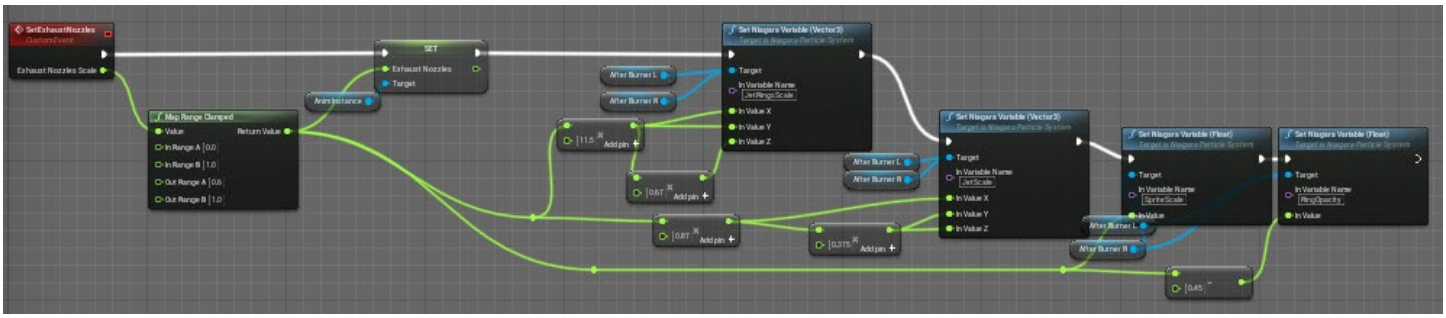
Control Flaps



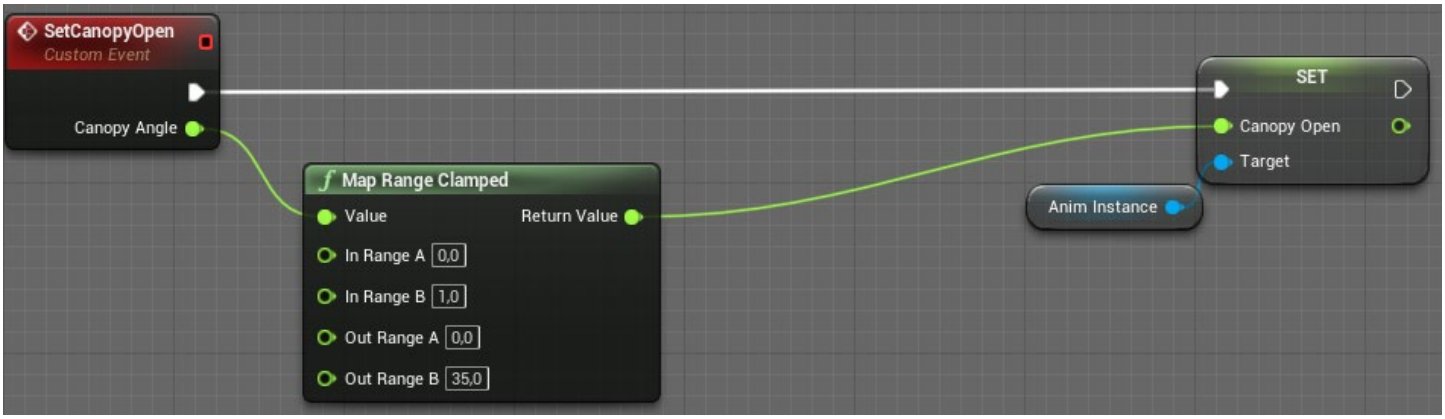
© Copyright Vigilante, LLC 2021 East Fighter Su33



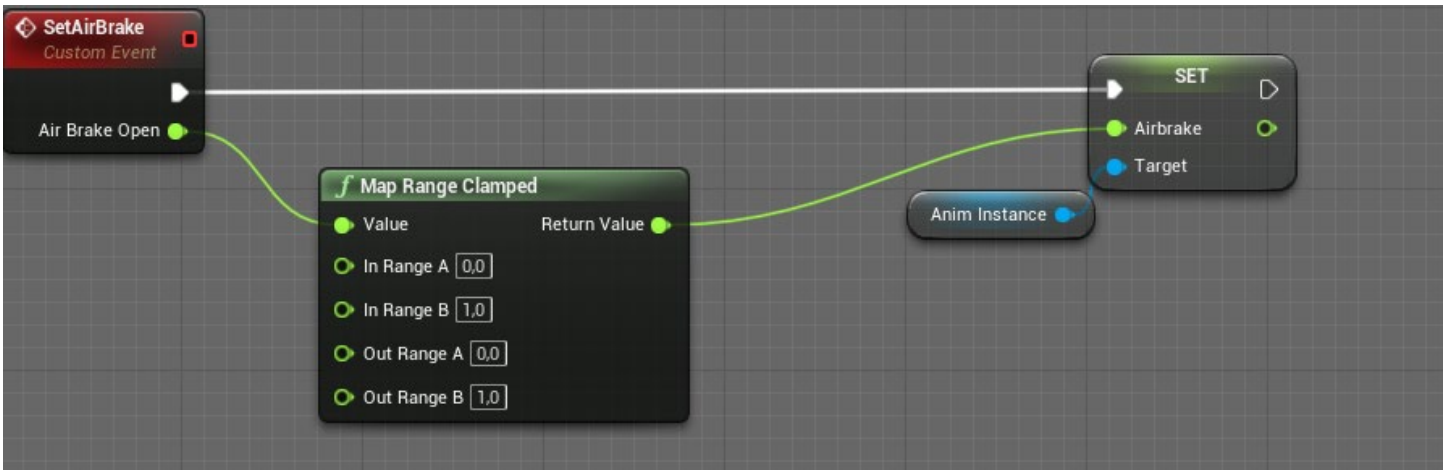
Control Exhaust Nozzles



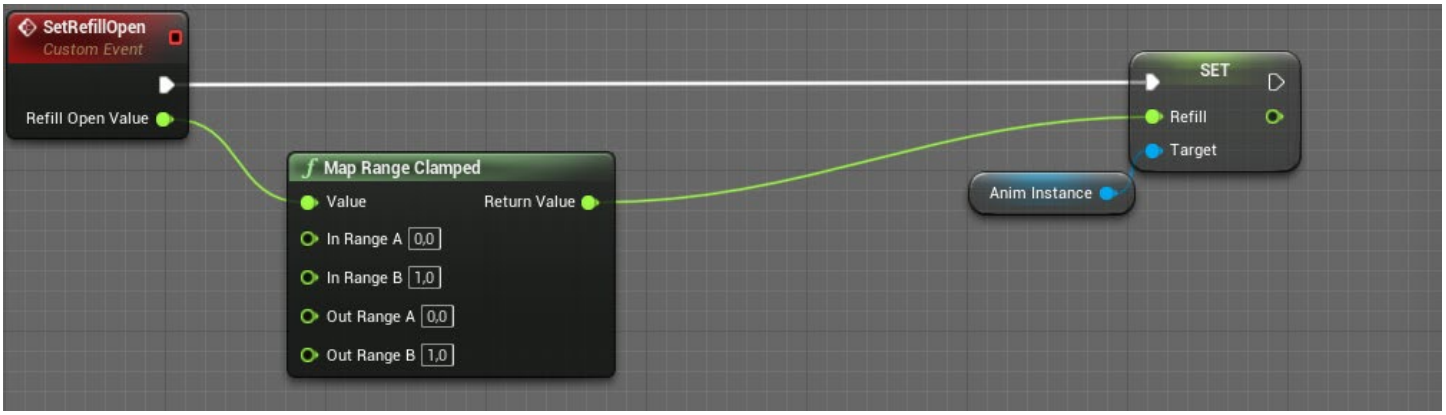
Control Canopy



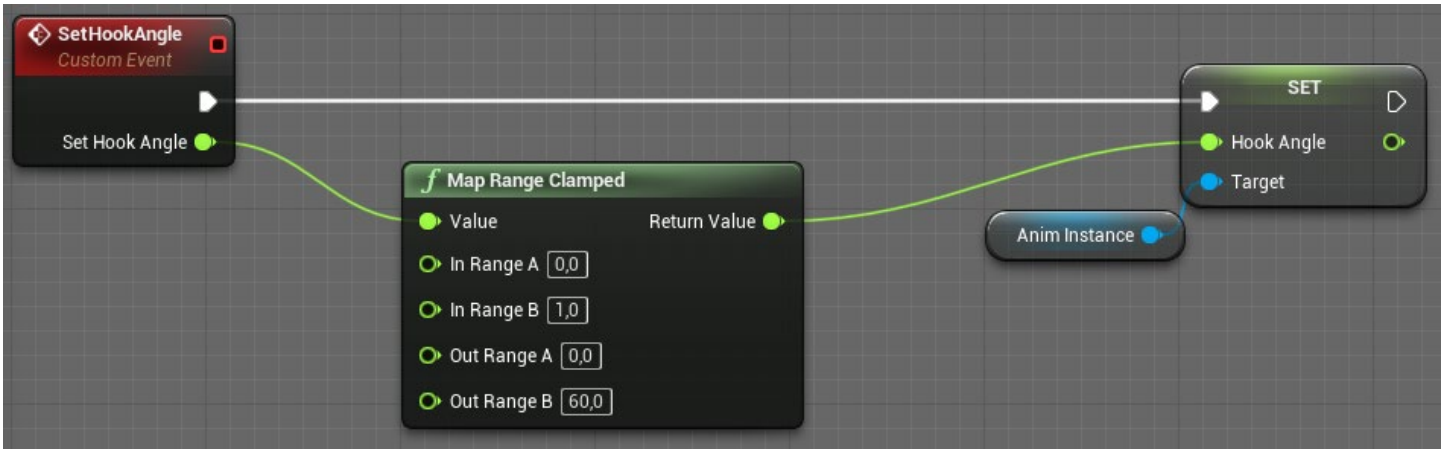
Control Air Brake



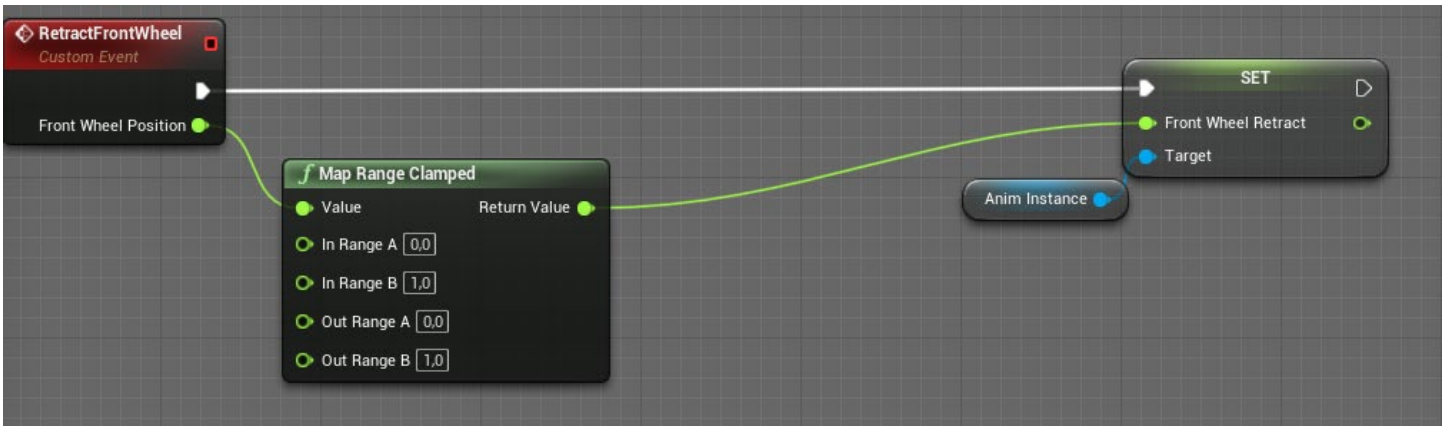
Control Refill



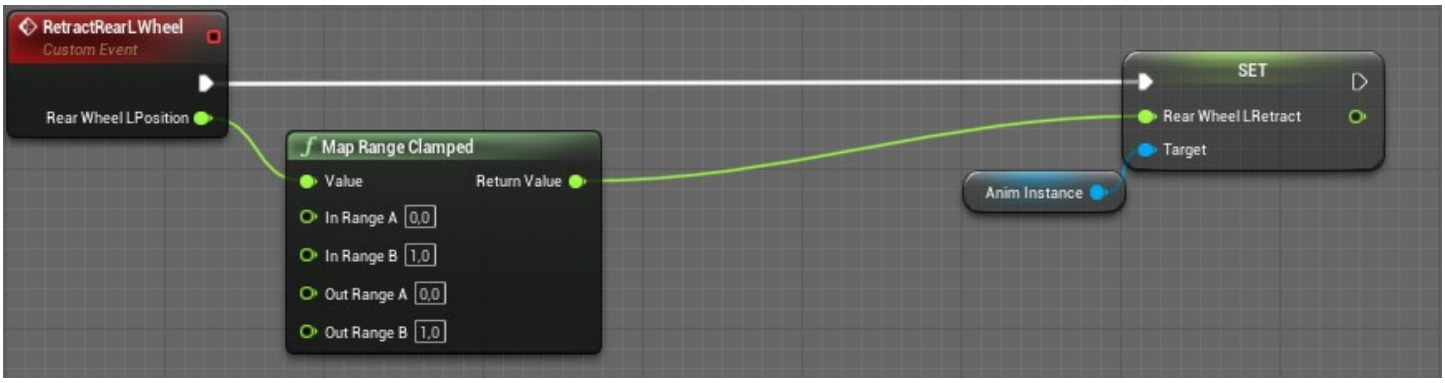
Control Hook

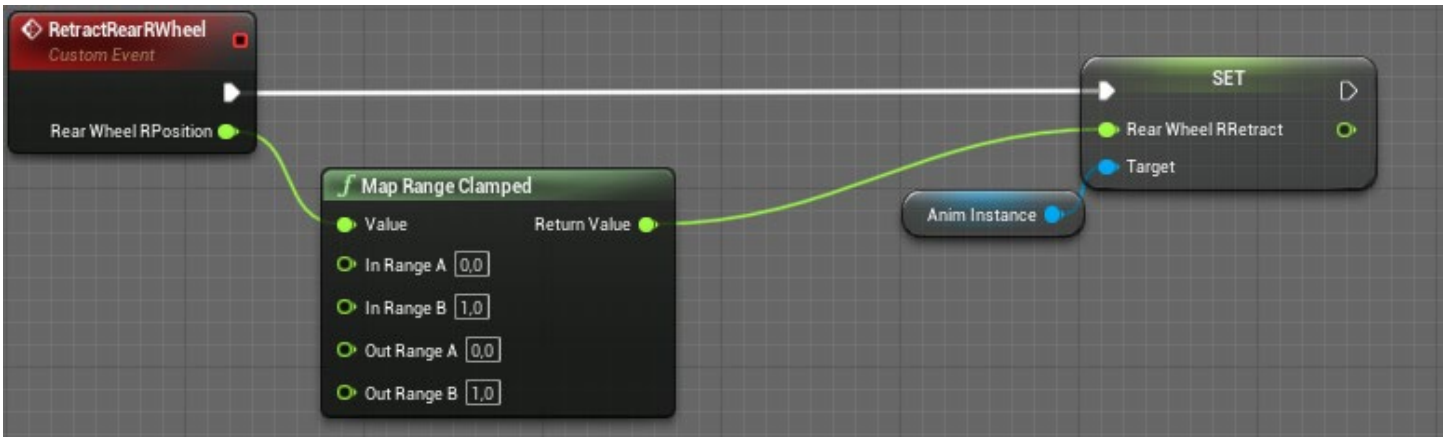


Control Front Wheel Retract



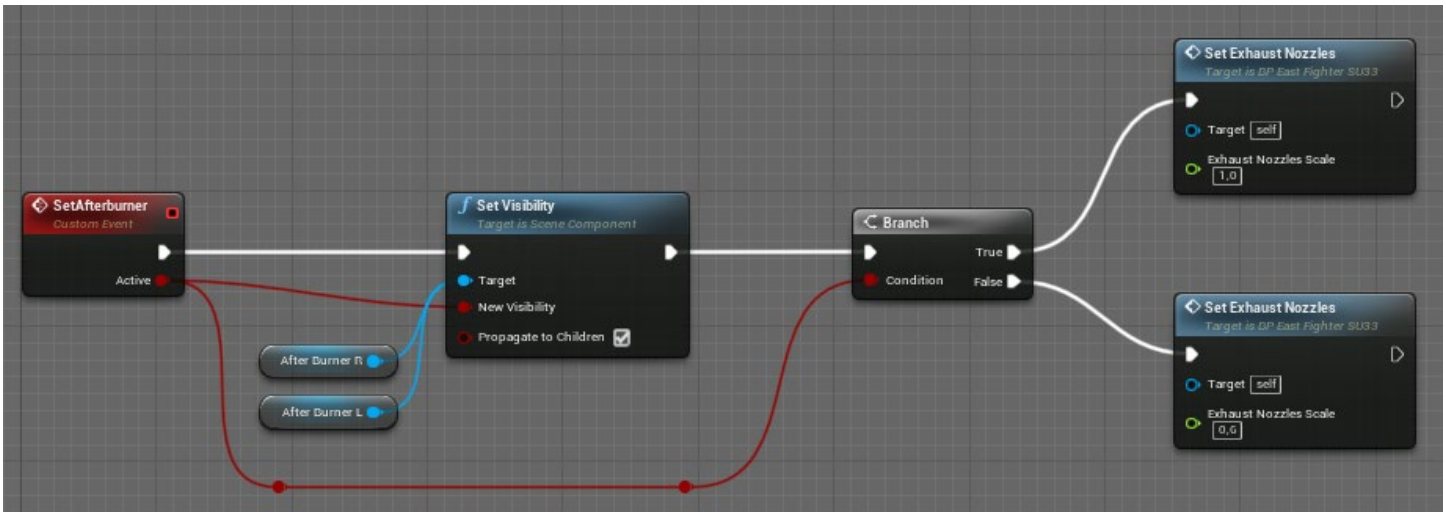
Control Rear Wheels Retract



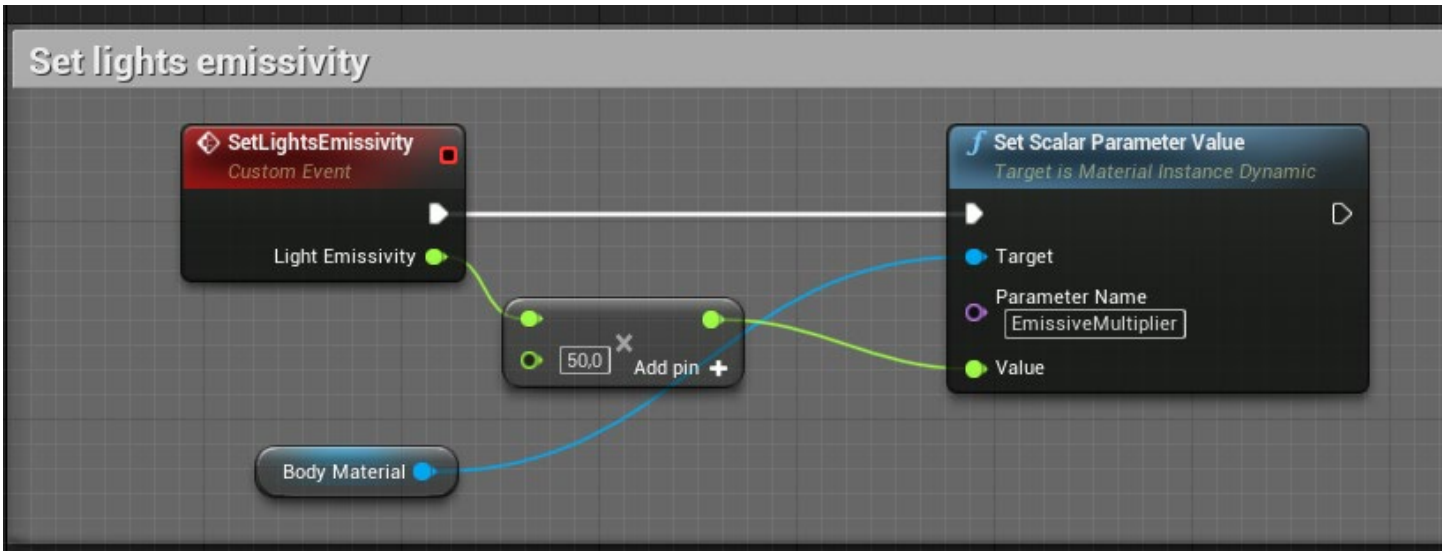


© Copyright Vigilante, LLC 2021 East Fighter Su33

Control Afterburner

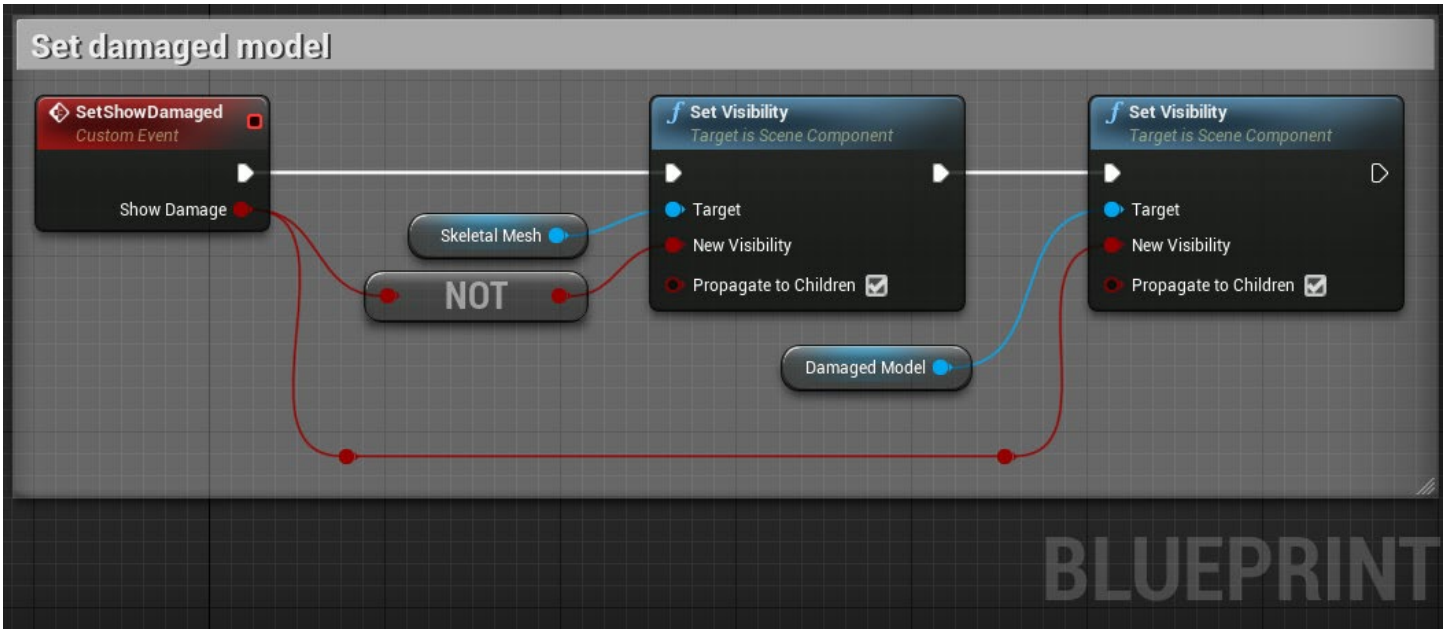


Control vehicle lights



Control damaged state visibility

The Vigilante asset has a ‘damaged’ state called ‘Damaged’. This state is triggered through the **SetShowDamaged** parameter. Setting this to TRUE will cause mesh visibility to toggle to the damaged state.


Set damaged model
SetShowDamaged
Custom Event
Show Damage
Skeletal Mesh
NOT
Set Visibility
Target is Scene Component
Target
New Visibility
Propagate to Children
Damaged Model
Set Visibility
Target is Scene Component
Target
New Visibility
Propagate to Children
BLUEPRINT