Table of Contents

Factions Extension Documentation

Add factions and relations in your game using C++ or Blueprints

This plugin is for Unreal Engine 4 and has support for versions **4.20**, **4.19** and **4.18**.
You can download this **Test project** to see and test the API

Introduction

What are "Factions"

When we say Faction we refer to a **group of entities or actors** that share something in common.

Almost every single type of game uses factions in different ways. For example, in Shooter games there will be enemies and friends, in RTS games every player will be a faction by itself, and in Open-World games you will have factions fighting each other while you run around.

This plugin fulfills the needs of this feature in UE4 with a very flexible tool that will make the implementation, editing and design of your own factions a 5 minutes thing.

"What" can have a faction?

Everything!... well not everything, but pretty much. **All actors of any type can have a faction.**

Implementation

Blueprints

*Check the **Test Project** for an in-engine example*

Adding the interface

For an actor to have a faction, we need to add a FactionAgentInterface. This will allow the system to set and get the current interface.

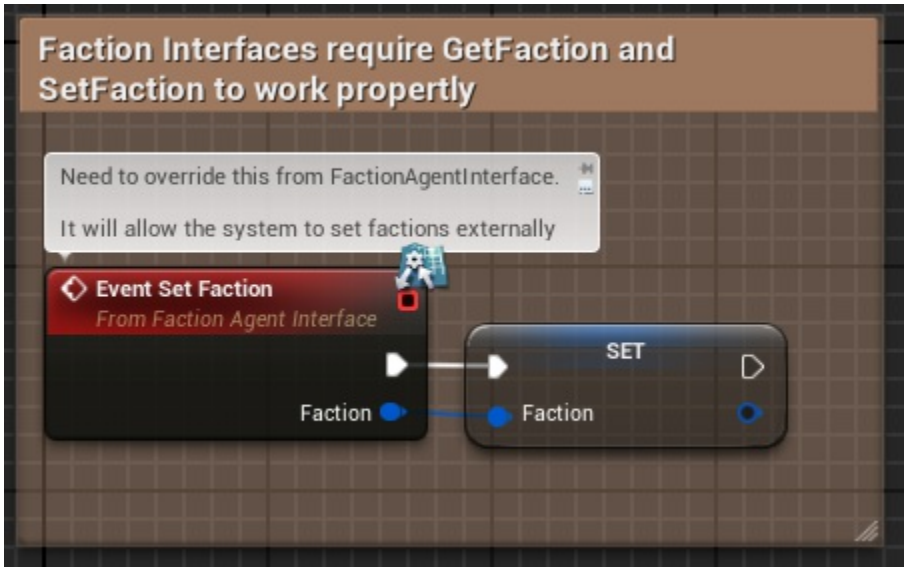
To add an interface we have to go into **Class Settings**

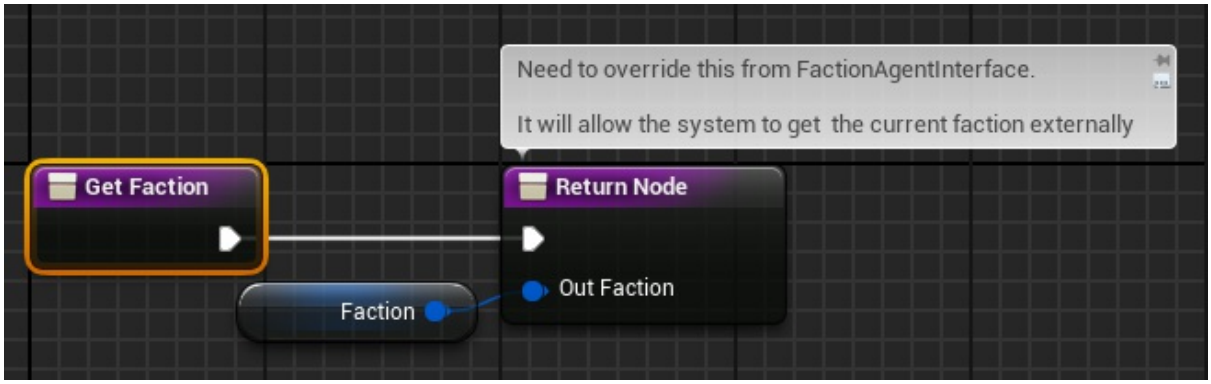
From here we click on **Add**, inside Interfaces and search for FactionAgentInterface

Getting and Setting the Faction

Factions require your actor to define how to get and set a faction. That can be done overriding **GetFaction** and **SetFaction**.

A common implementation would be to have a **variable** (of type *"Faction"*), and then return it with GetFaction and set it with SetFaction:





Factions in Controllers

Sometimes you may want a controller to share a faction with its controlled pawn or character.

This process is exactly the same as before, except that instead of getting and setting a variable, we will get and set the faction from our pawn.

C++

Adding the interface

Add **IFactionAgentInterface** to any Actor:

```
UCLASS()
class GAME_API AUnit : public APawn, public IFactionAgentInterface
{
    GENERATED_BODY()
```

Getting and Setting the Faction

In order for the system to read our faction, we need to provide a getter and a setter. The returned variable depends on what your code needs.

```
/** Retrieve faction identifier in form of Faction */
virtual FFaction GetFaction() const override { return Faction; }

/** Assigns faction */
virtual void SetFaction(const FFaction& InFaction) override;
```

A common case is to have a Faction variable exposed to the editor *(like in the example)*. You could also set dependent factions by getting another actor's Faction *(e.g Inside a controller)*.

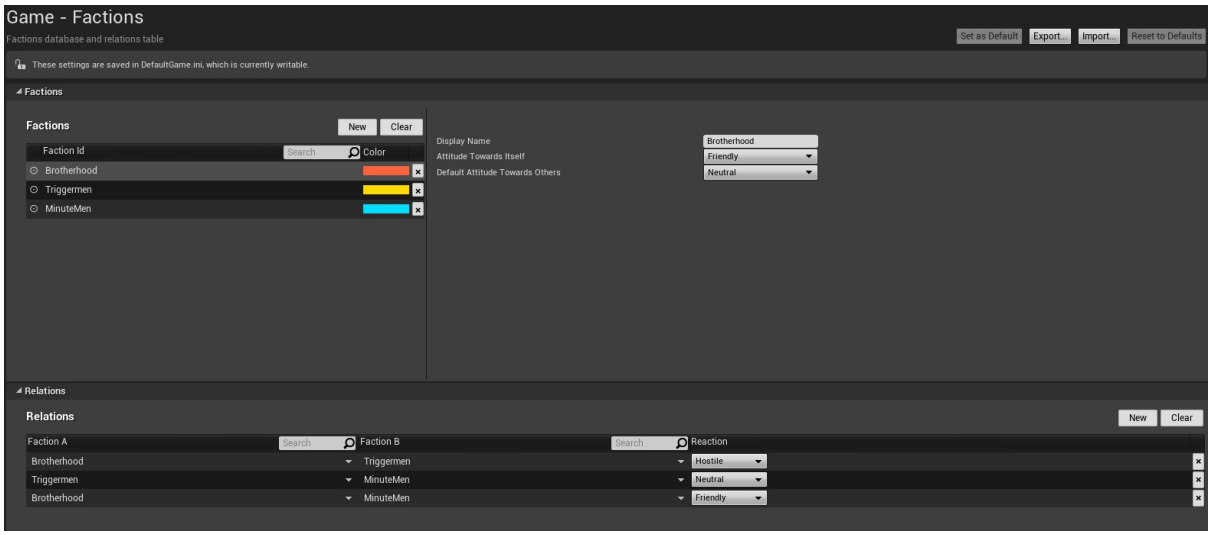
Using Factions without Actors

Non-actor objects can also have factions, but they can't be called though normal API functions for simplicity. This means that, for example, a component could have a faction, but only Actors can be checked for attitude.

This is useful for modularizing Actor logic inside components, distributing faction logic.

Usage

All settings can be edited from **Project Settings -> Game -> Factions** tab.

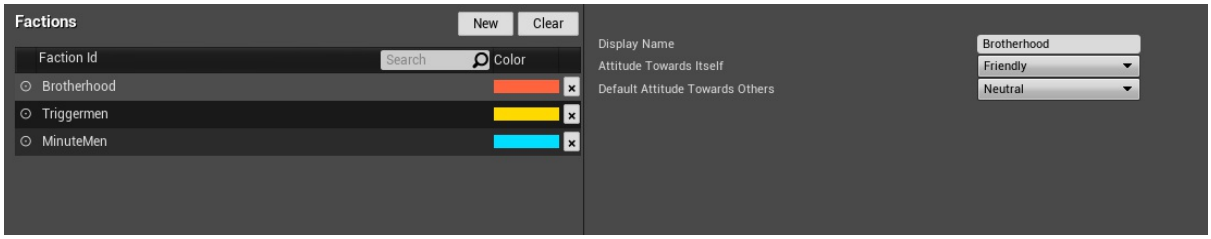


Factions

Factions settings are defined by right and left areas.

At the left you have a list of all factions, with a little search box in case you have more than you can deal with.

At the right you will find all the properties of a selected faction (which was selected by just clicking on it on the list).



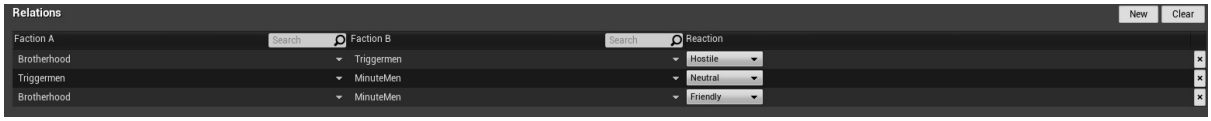
You can also rename factions by double clicking on them.

All Faction information can be read on runtime too:

Relations

Relations define how will two factions interact to each other.

In this example, Brotherhood and Triggermen are enemies:



By default there are three possible attitudes: **Hostile**, **Neutral** and **Friendly**

Installation

Manually

This are the general steps for installing the plugin into your project:

1. Download the last release from here
Make sure you download the same version than your project

2. Extract the folder “FactionsExtension” into the **Plugins folder** of your existing project (e.g "MyProject/Plugins")

2. Done! You can now open the project

From Marketplace

Install from the launcher: AVAILABLE HERE