by René Hecker

Mai 2023

Unreal Engine 5

Gameplay Ability System

Beginner's Guide

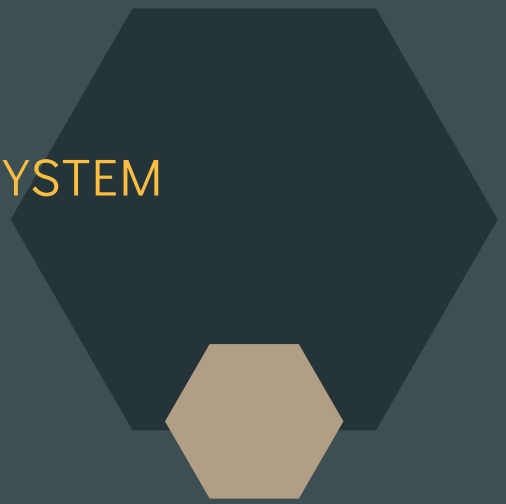
Game Engineer

at School For Games Berlin

github.com/TheBitFossil

GAMEPLAY ABILITY SYSTEM

Preface



Preface

The purpose of this guide is to help others getting started with the Gameplay Ability System. Focussing on visualization, step by step instructions and showcasing a "Minimum Viable Prototype".

I recommend having knowledge about Character Locomotion, World Settings, Game Mode, Player State, Replication and C++ before starting with this guide.

Please check the attribution of sources for a deeper explanation about the GAS.

Preface

Summary

1 **Introduction**
What Is The Unreal Engine Gameplay Ability System ?

2 **Components**
Gameplay Tags, Attributes, Abilities & Co.

3 **Initializing The Project**
First steps to a functional template

4 **Starting with GAS**
How to set up a minimal GAS project

5 **Default Attributes**
Our first Ability System check

6 **First Interaction**
Creating a Gameplay Effect and changing Attributes

7 **Start Creating Abilities**
We are ready to implement the first Ability

8 **Personal Note**
Is it worth investing your time to learn the GAS ?

9 **Sources**
Useful resources to start learning more about the GAS

10 **Appendix**
Extras, Information, Erklärung


GAMEPLAY ABILITY SYSTEM
Introduction

Introduction

The Gameplay Ability System (GAS) is a flexible, multiplayer ready framework to create gameplay mechanics, skills and abilities.

It can be used for a wide variety of genres including Action-RPG's, RTS, MOBA's.

1. Introduction

Should you use the GAS ?

1. Introduction

What is the Unreal Engine Gameplay Ability System ?

Modular

Each component contains its own logic, allows easy extension of the system.

Abilities & Skills

Interactions, Jumping, Aiming, Dashing, magical Spells and more.

Cooldown & Costs

Locking Abilities by time (cooldown) or values (Mana, Stamina, Health).

Reuseability

Skills can 100% be reused for the next Project.

Dynamic

Changing Attributes or creating new Gameplay Effects during runtime.

Effects

Handles visual and sound effects with its own logic.

Complex

High learning curve, takes time to set up and debug a basic template project.

Powerful

Knowledge of C++ is necessary to unleash its full potential.

1. Introduction

What to do when your stuck?

Question

Where or how do I start?

From the beginning the whole System seems to be a bit daunting when facing a huge code base and a lot of new components for the first time.

Answer

Watch the Unreal Engine talks about the GAS. Learn every component and what it does. Then read the source code and read it a second or third time.

Question

How does this System work ?

There will be a moment when you do not comprehend what this specific thing does and where you have to write code.

Answer

If you are new to a system then do your best to visualize each of the parts that you do understand and research about the ones you don't. This will help to get a better picture of the overall structure.

1. Introduction

What to do when your stuck?

Question

How to stay motivated ?

"It will never work, its just too big and the code does not compile and breaks a lot."

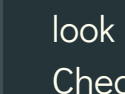
Answer

Take a break. Write down what you have already achieved and be proud of every little step. This is a complex system and it may take some time to get used to. Others have mastered it, so can you.

Question

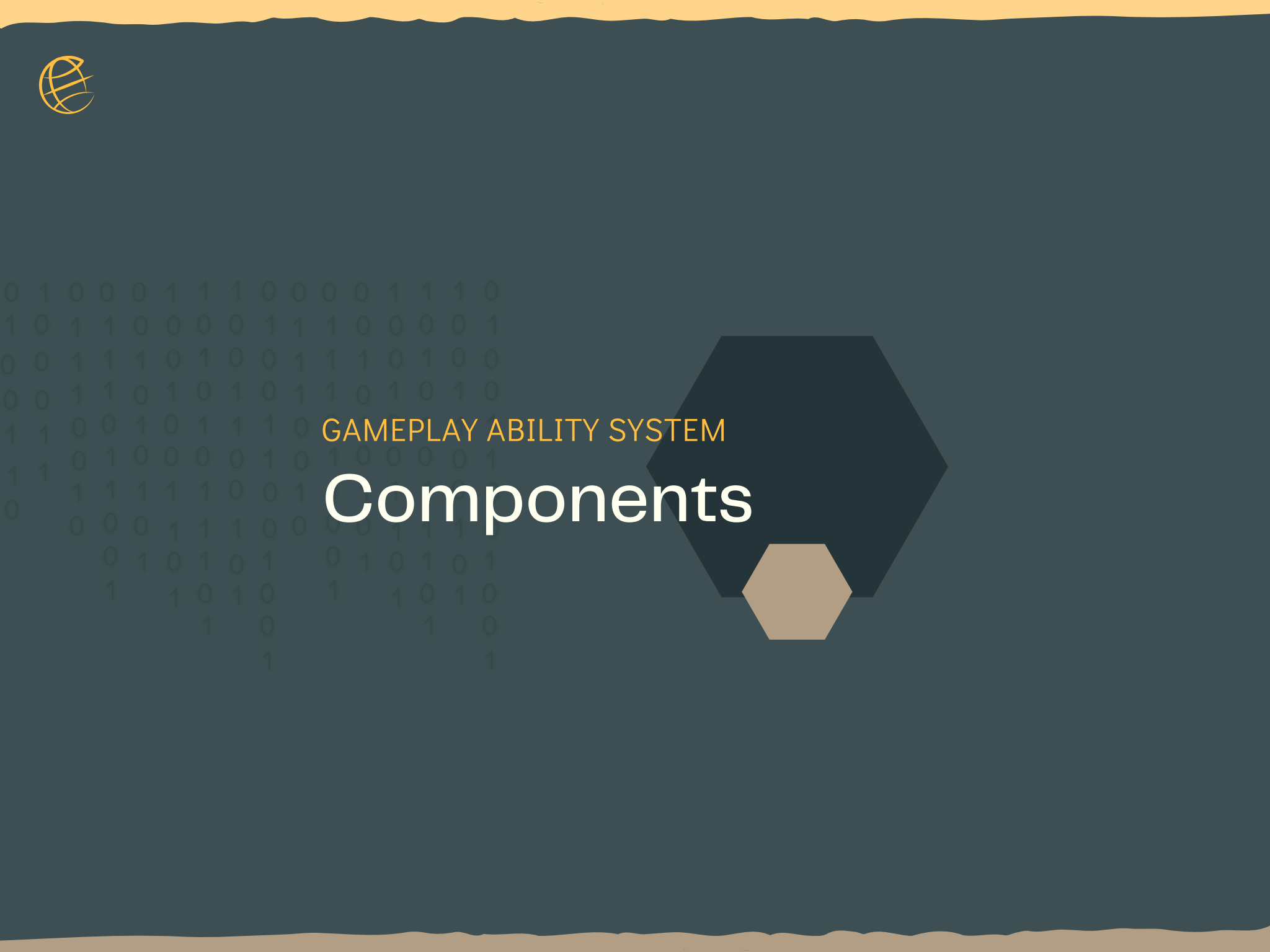
I don't find a solution ?

You took a break, are calm, focused and researched about this specific component. Even read the API and source code twice. Still no luck ?!



Answer

If you really don't find a solution, then look for others who are in the same boat. Check on forums, watch tech-talks, ask questions and start again. There will always be a solution.


GAMEPLAY ABILITY SYSTEM
Components

Components

At a higher level, the backend of the Gameplay Ability System consists of multiple components, each with its own specific logic and responsibilities, working collaboratively to enable the system's functionality.

2. Components

Gameplay Tags, Attributes, Abilities & Co.

GAS

These components work together and are responsible for different aspects of the Gameplay Ability System.

Gameplay Tags (If)

- Conditions
- Tag Container
- Hierarchical

Gameplay Attributes (What)

- Values, Data
- AttributeSets
- Base & Current Value

Gameplay Effects (How)

- Changes Attributes & Tags
- Duration Policy
- Spec, Context & Handle

Gameplay Abilities (Why)

- Actions & Skills
- Costs & Cool-down
- Passive
- Self-Contained
- Active
- Predictable

Gameplay Tasks (When)

- Asynchronous
- Resposive
- Root motion

Gameplay Cues (Visuals)

- Non-Gameplay Related
- Effects
- Replicated

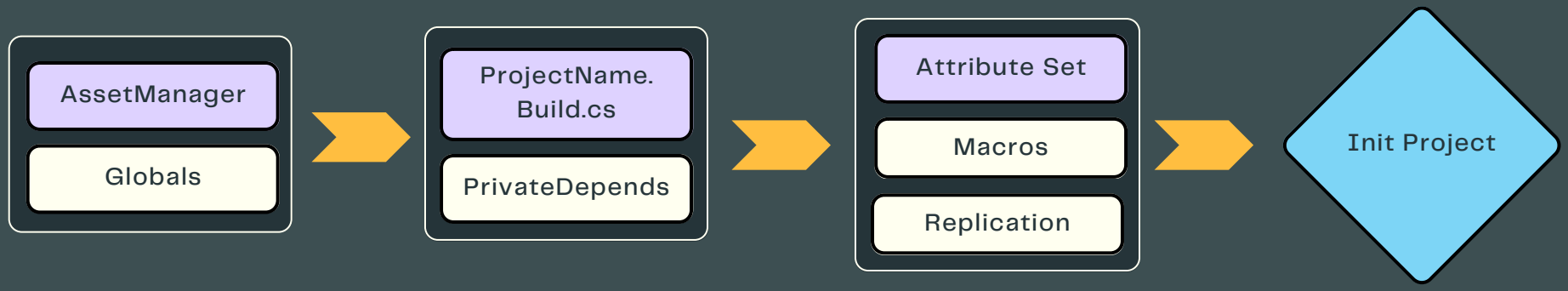

GAMEPLAY ABILITY SYSTEM
Initializing

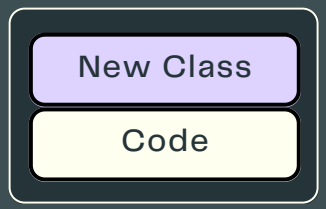
Initializing The Project

Before we can start working on the project template,
its necessary to initialize a few properties and helpers.

3. Initializing The Project

Classes and responsiblity





For More Information

Please check the source code of the template project and read the "How To Checklist".

github.com/thebitfossil

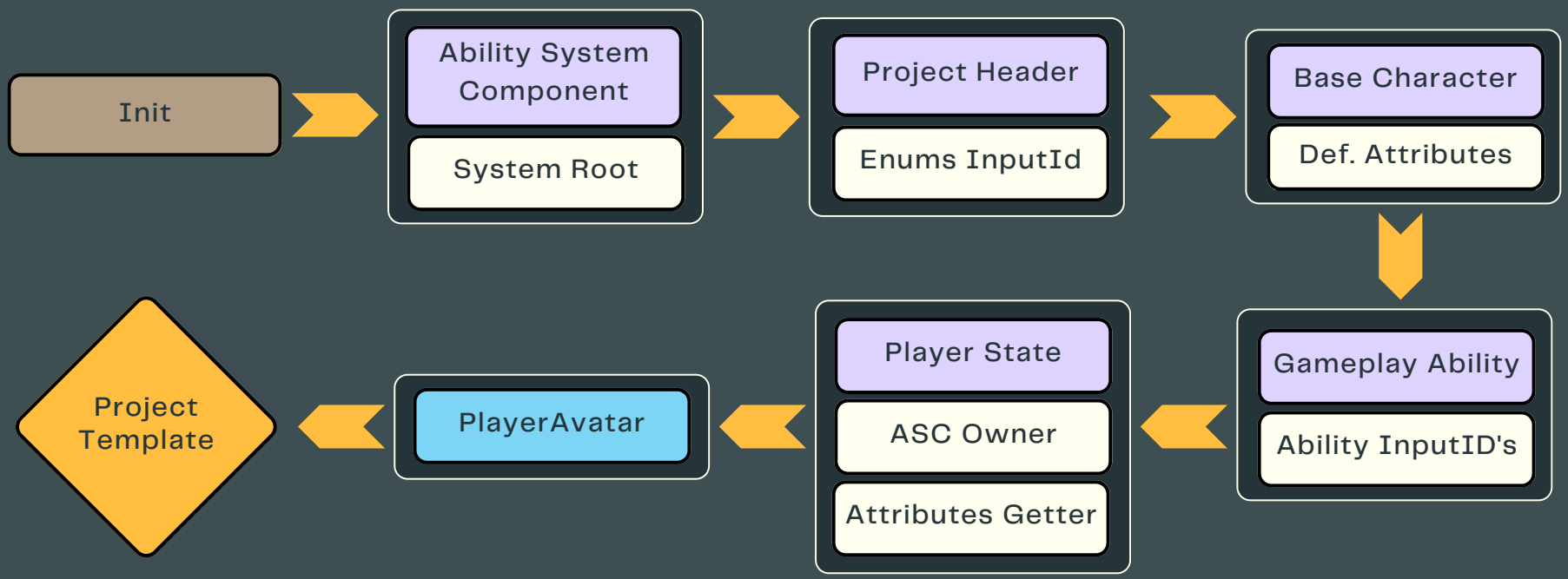

GAMEPLAY ABILITY SYSTEM
Starting with GAS

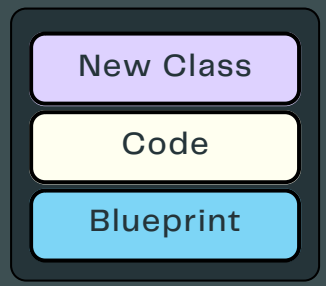
Starting with GAS

Lets start putting the base system together. The better you know each component the easier it gets to create a working template project.

4. Starting with GAS

Basic Classes and Responsibilities

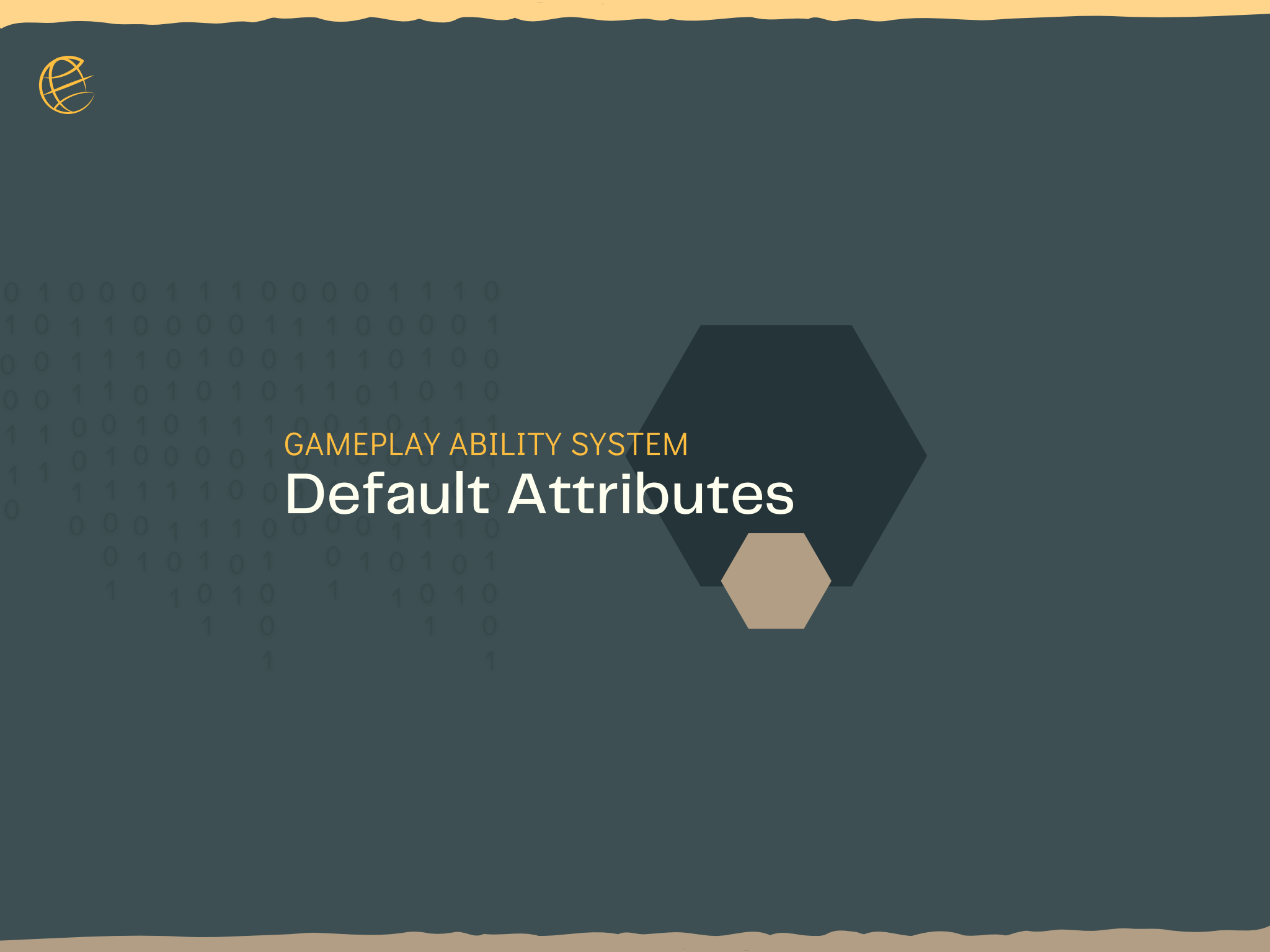




For More Information

Please check the source code of the template project and read the "How To Checklist".

github.com/thebitfossil


GAMEPLAY ABILITY SYSTEM
Default Attributes

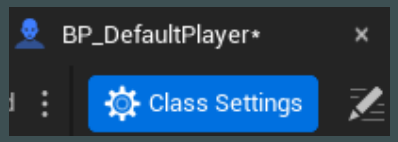
Ability System Check

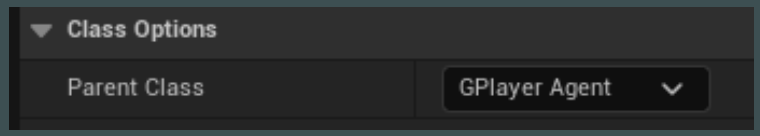
For the first time we are ready to put the pieces together.

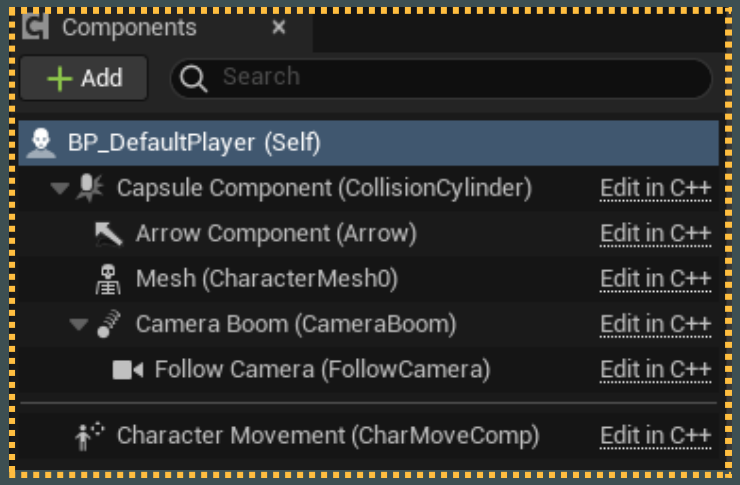
Carefully check every Class and Blueprint:
Game Mode, Default Pawn, Player State, Player Controller

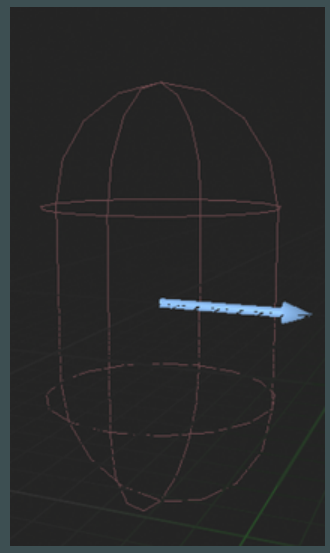
5. Default Attributes

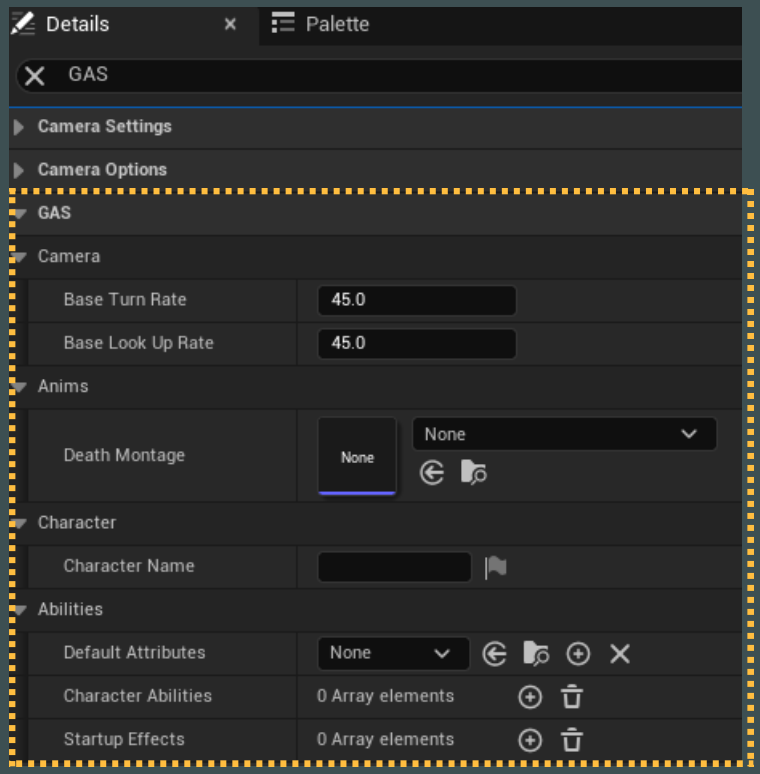
Player- and GAS-Settings











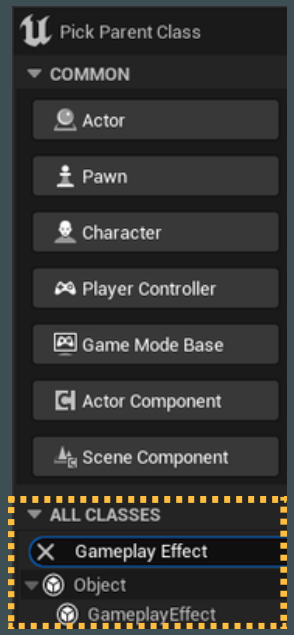
Default Player Blueprint

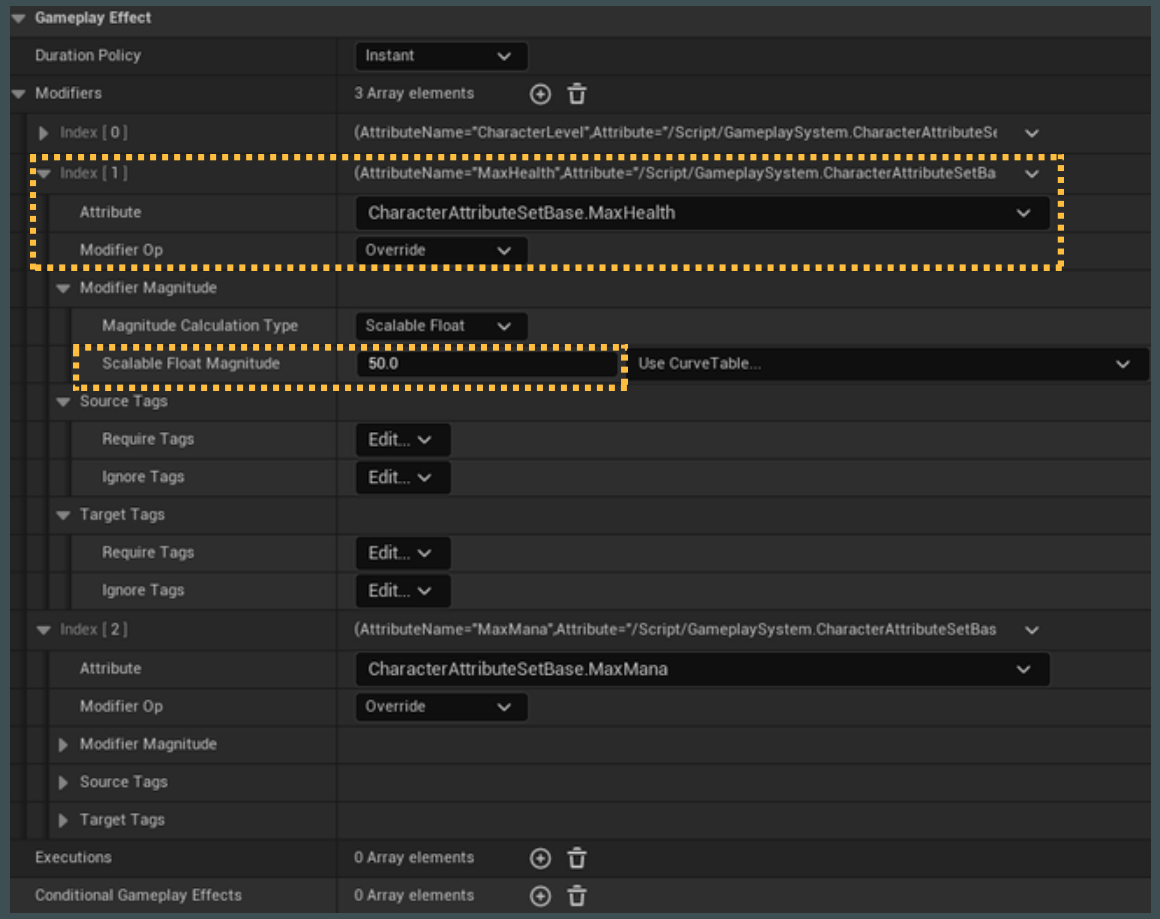
After creating a new BP from our C++ Player Class.

These are the default settings and the player should not be able to move.

5. Default Attributes

An instant duration Effect





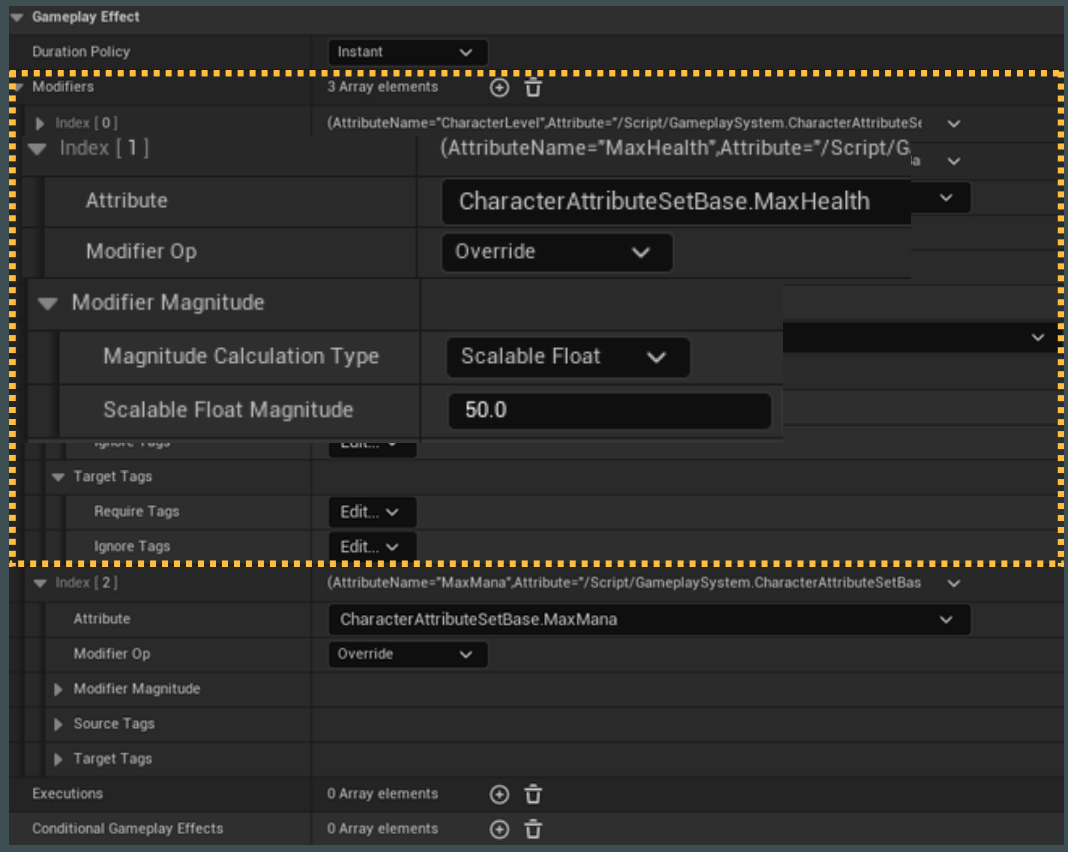
First Gameplay Effect

To check if the System is working, create a new Gameplay Effect.

It will contain the default values for the AttributeSet.

5. Default Attributes

Attributes & Values for Health, Mana, Stamina, etc.



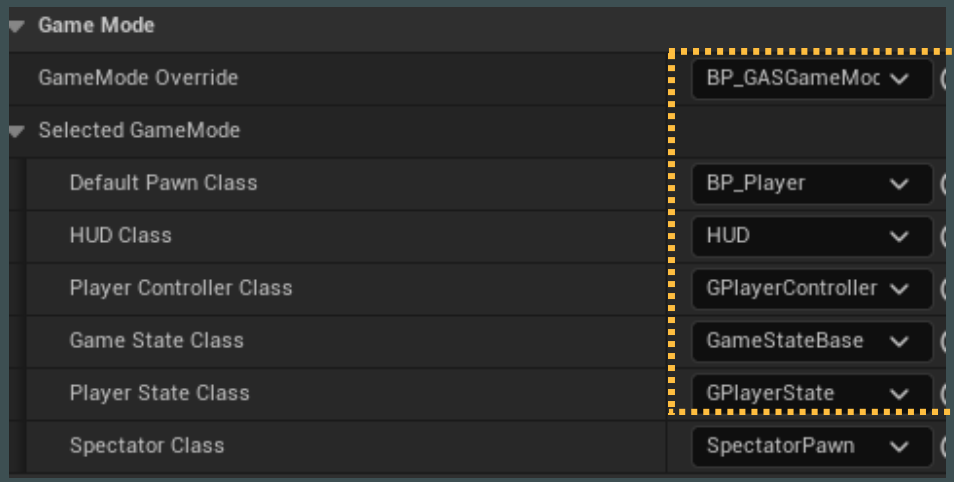
First Gameplay Effect Settings

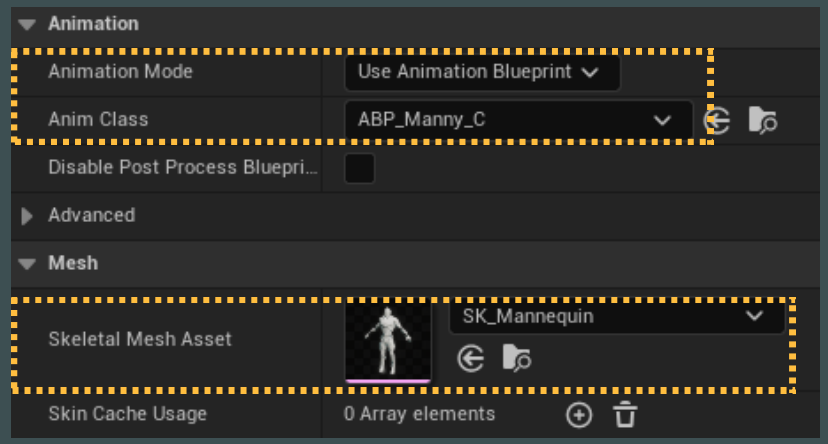
Add Modifiers and choose from the Attributes drop down.

Override the Float Magnitude with your values.

5. Default Attributes

General Settings & Animations





World Settings & Game Mode

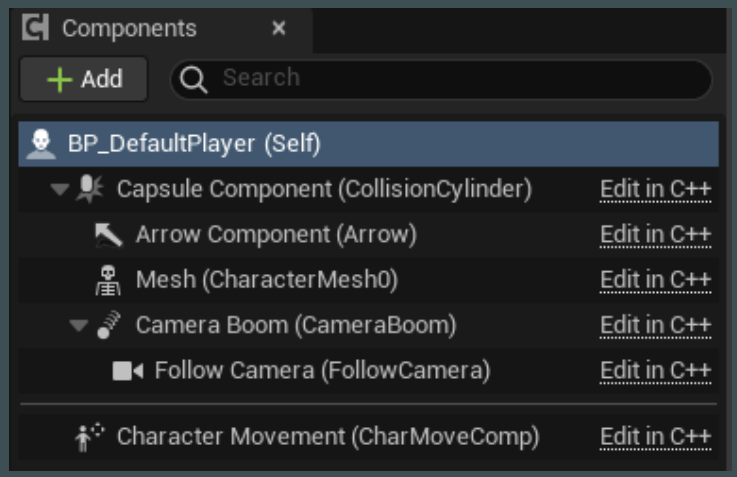
Think about overriding the default values.

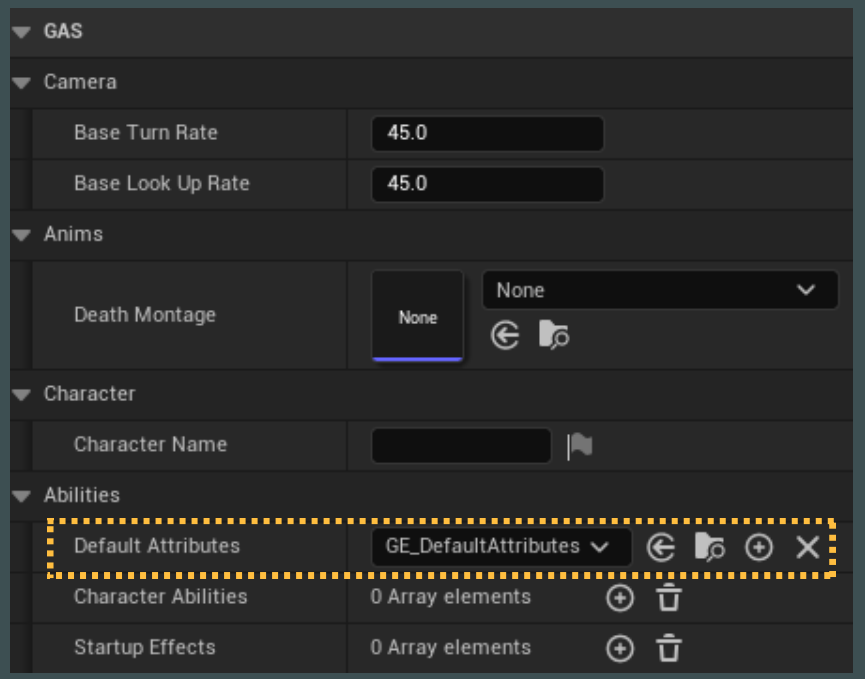
Animation & Mesh

This would also be a good time to choose your Mesh and Animation Blueprint.

5. Default Attributes

Back to the Player Blueprint





Default Player Blueprint

Add the default Attributes,

You should have Health and the Player can finally move.

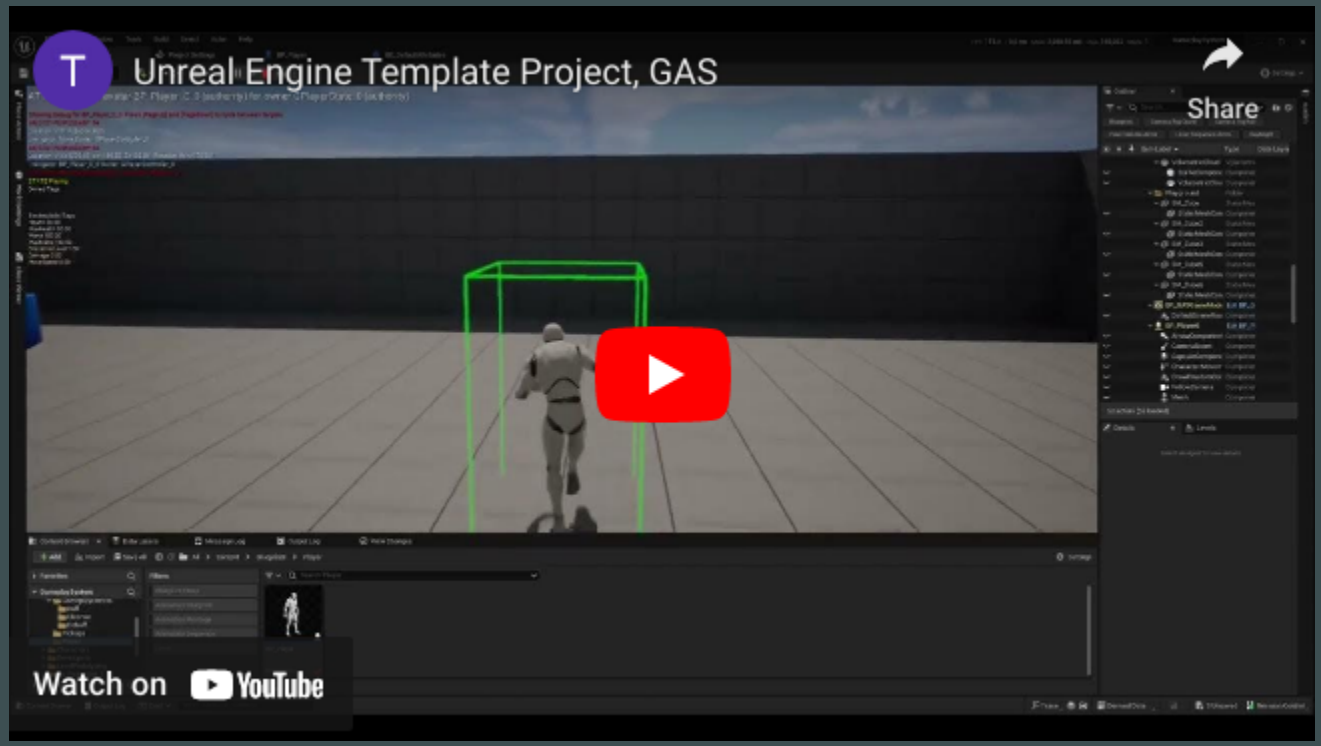
Debugging

Open the console and type:

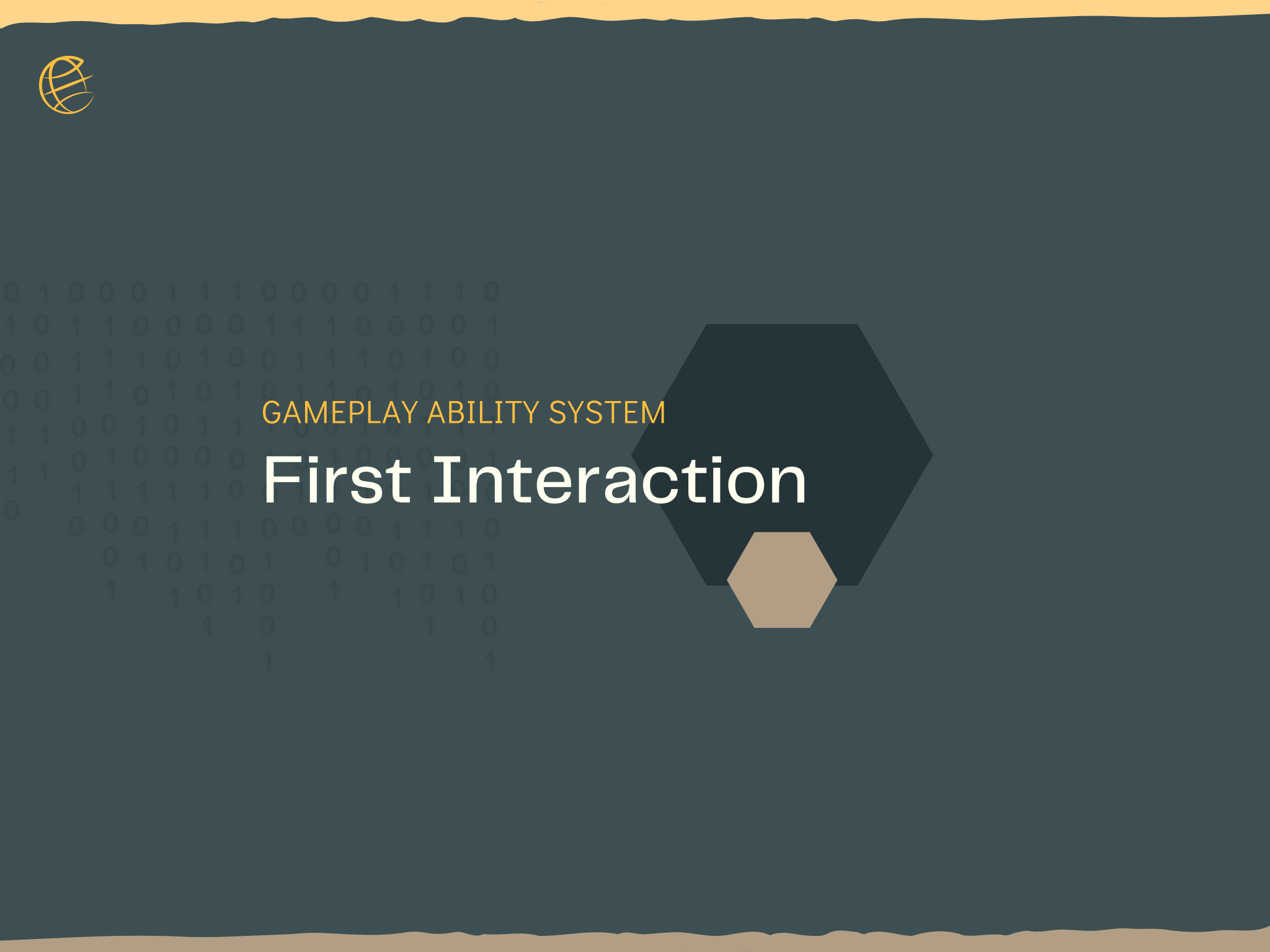
showdebug abilitysystemcomponent

5. Default Attributes

Prototype playable state



[Link to Video]

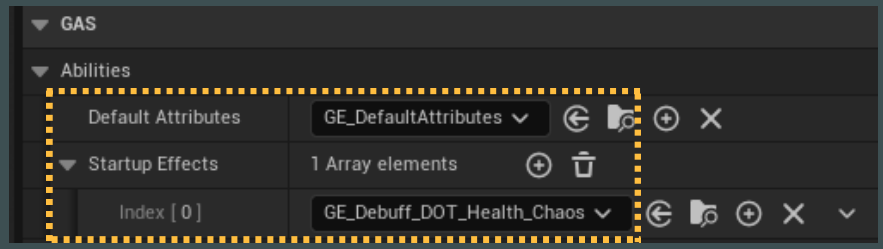

GAMEPLAY ABILITY SYSTEM
First Interaction

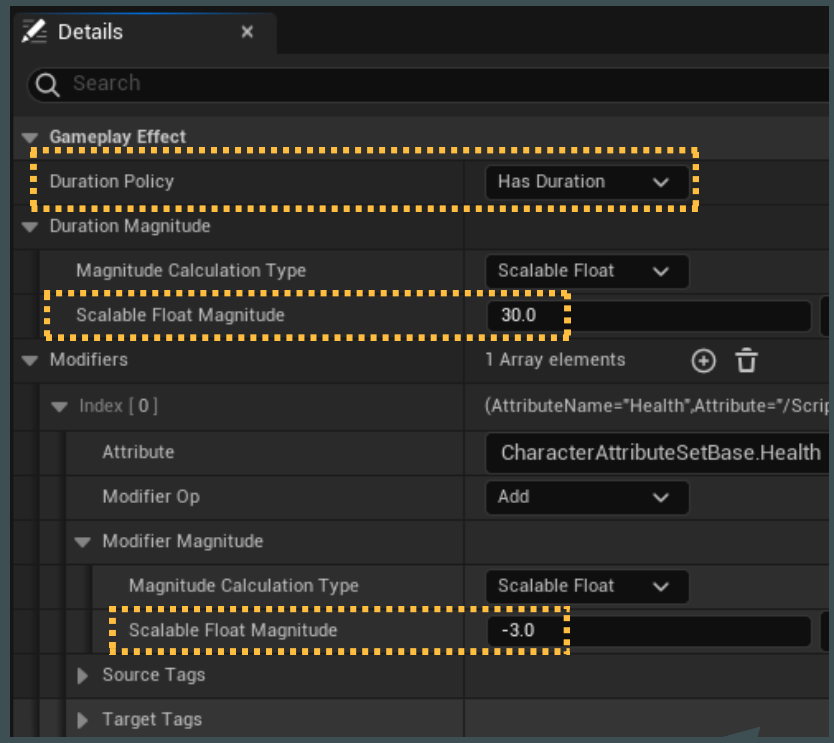
Attributes and Effects

The current Health Attribute allows us to move. An easy check if a new Ability works, would be to damage the player so he stops moving.

6. First Interaction

Second Gameplay Effect:
Damage Over Time





Player Avatar

Create a new Gameplay Effect class and add this as the Startup Effect.

Our goal is to take Damage as soon as we hit Play.

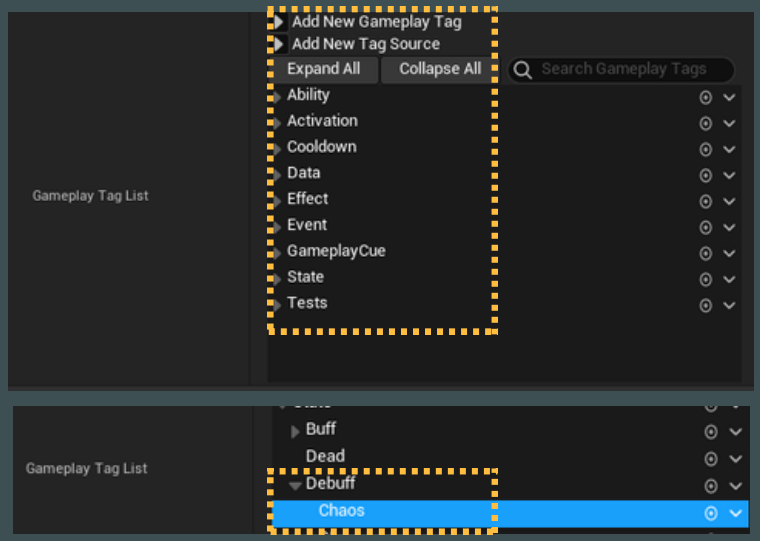
Gameplay Effect & Tick

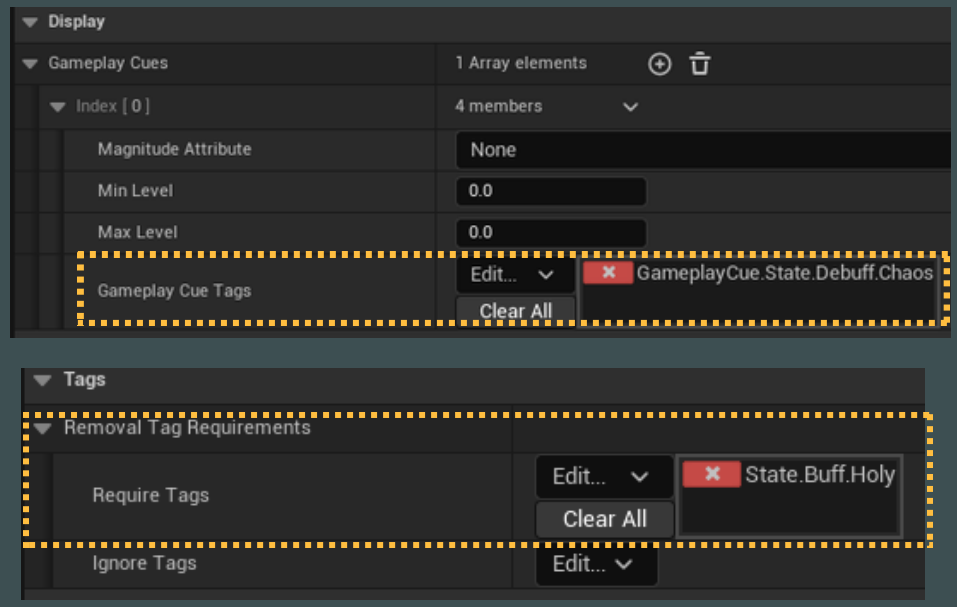
Has a Duration of 30 sec. and adds -3 to the Health Attribute per tick.

To change the the tick/sec: Search for the Period settings.

6. First Interaction

Organizing and Handling Effects with Tags





Gameplay Tags

Organize your States and Abilities with these handy Tags, applied by the Effect.

The application of Abilities depends on active Gameplay Tags.

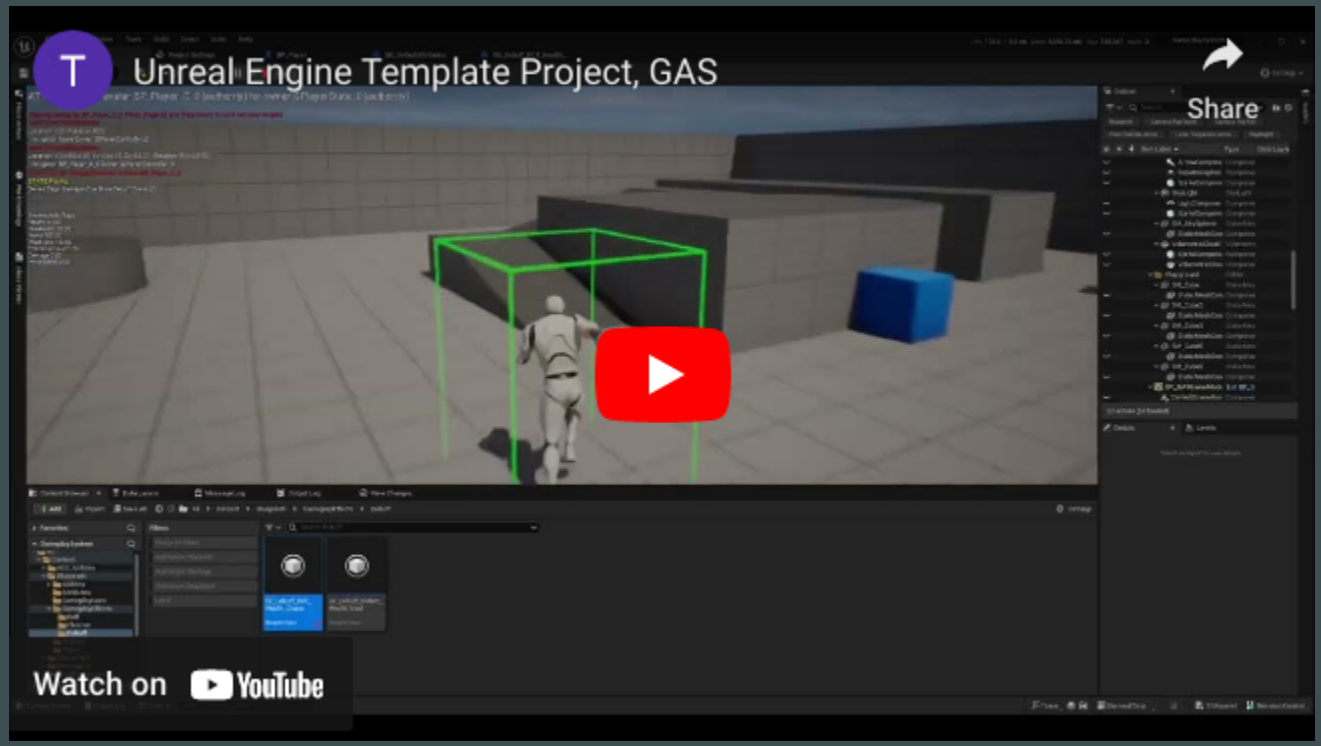
Gameplay Cue & Removal

Cues are used to show a visual feedback when the effect is active.

Effects can be removed from the owner by applying other Gameplay Tags.

6. First Interaction

Prototype: Player takes Damage Over Time



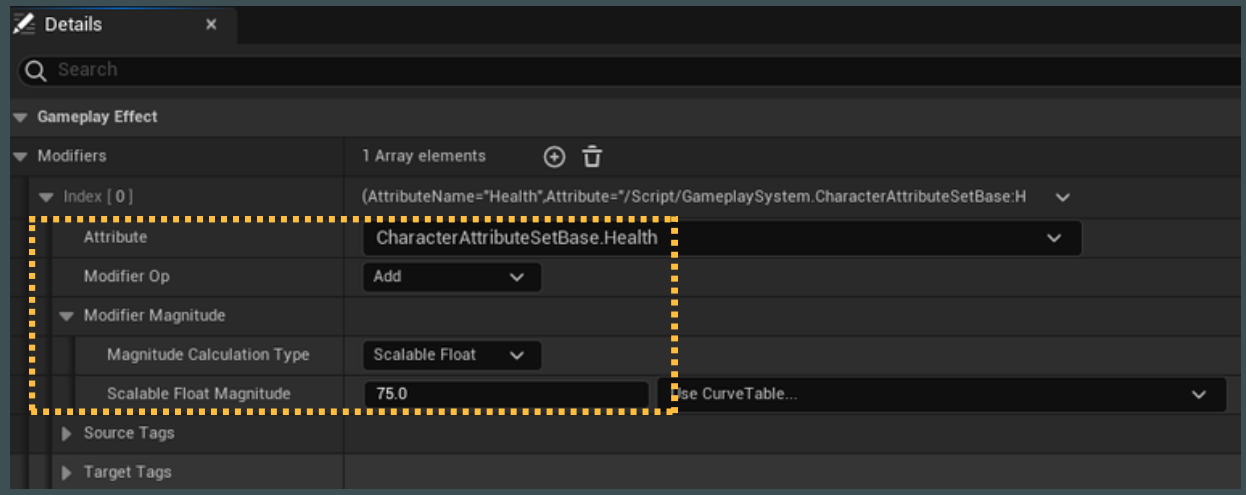
[Link to Video]

6. First Interaction

Step 1: Creating a Cleanse Potion

Gameplay Effect

Create a new Blueprint class, we use this to modify our Health Attribute with instant 75 points.



Step 1 Step 2 Step 3 Step 4 Step 5

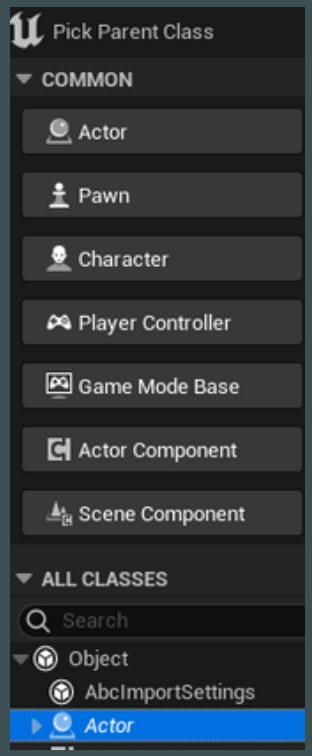
Gameplay Effect

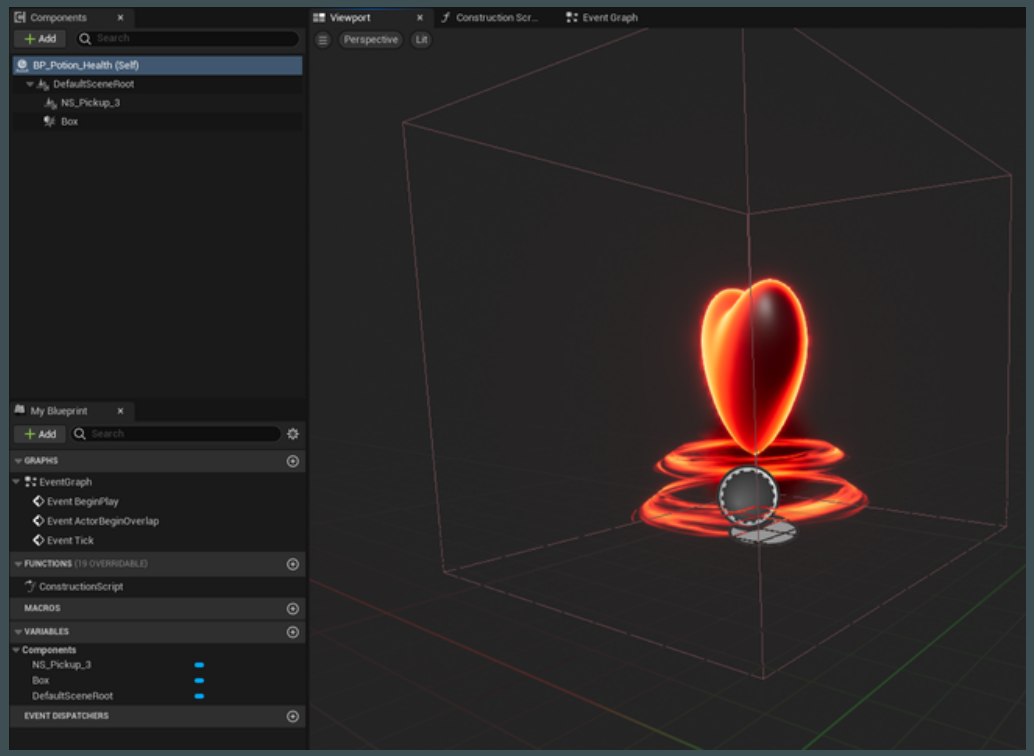
Health is an Attribute.

Gameplay Effects change our Attributes.

6. First Interaction

Step 2: Potion Actor





Actor Blueprint

Any Object that is visible will do. Placeholder that activates a Gameplay Effect on overlap.

Step 1 — Step 2 — Step 3 — Step 4 — Step 5

Gameplay Effect

Health is an Attribute.

Gameplay Effects change our Attributes.

Actor Blueprint

Overlapping Actors.

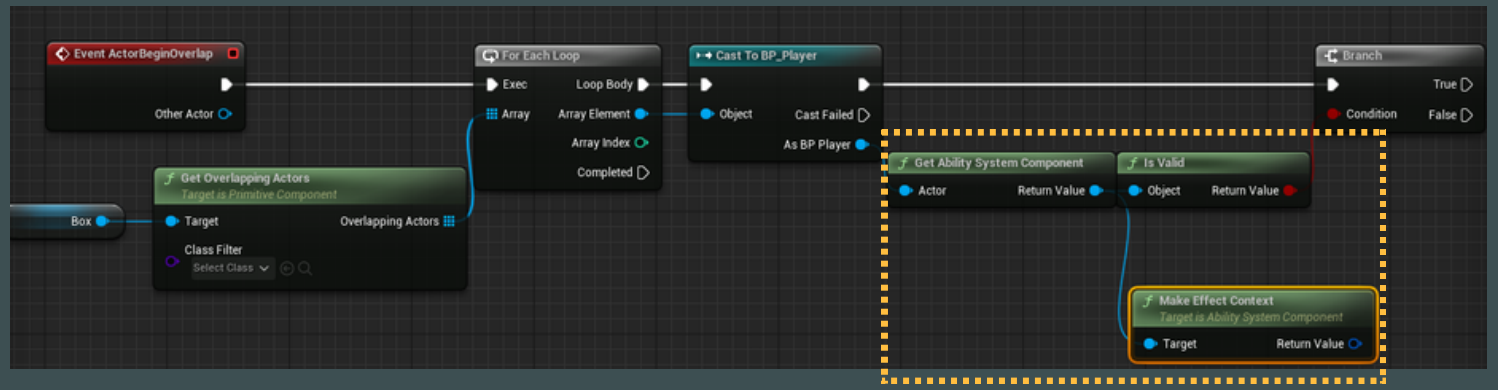
A simple pickup with a box collider attached.

6. First Interaction

Step 3: OnOverlap reaction

Casting to our Player

Getting the Effect Context from the Ability System Component.



Step 1 — Step 2 — Step 3 — Step 4 — Step 5

Gameplay Effect	Actor Blueprint	Visual Scripting
Health is an Attribute.	Overlapping Actors.	Logic behind Overlap.
Gameplay Effects change our Attributes.	A simple pickup with a box collider attached.	Context carries information about the Gameplay Effect.

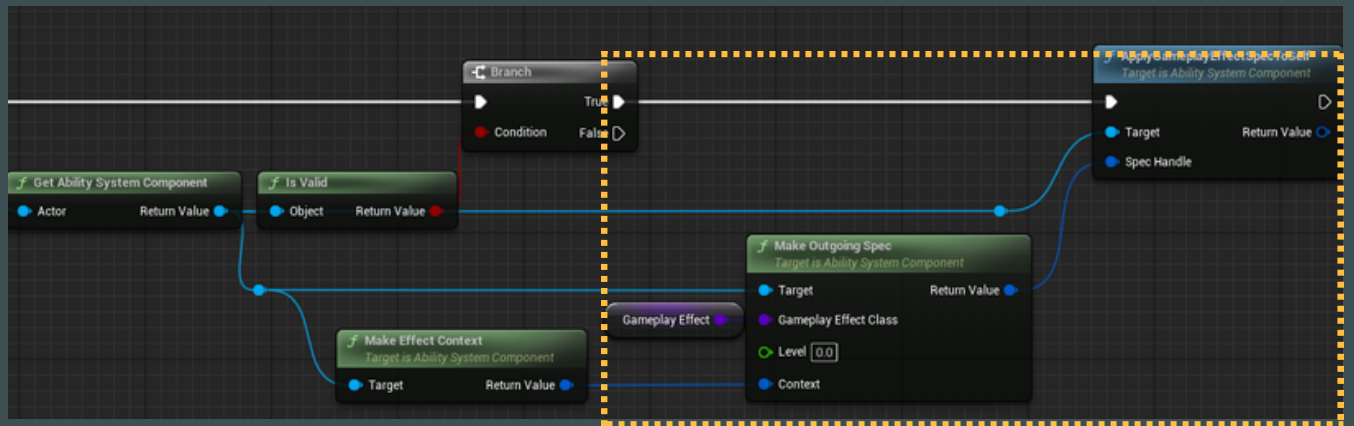
6. First Interaction

Step 4: Effect application

Gameplay Effect Specs

Effect Context holds specifications.

Of the Effect that will be applied to the AbilitySystemComponent.



Step 1 | Step 2 | Step 3 | Step 4 | Step 5

Gameplay Effect

Health is an Attribute.

Gameplay Effects change our Attributes.

Actor Blueprint

Overlapping Actors.

A simple pickup with a box collider attached.

Visual Scripting

Logic behind Overlap.

Context carries information about the Gameplay Effect.

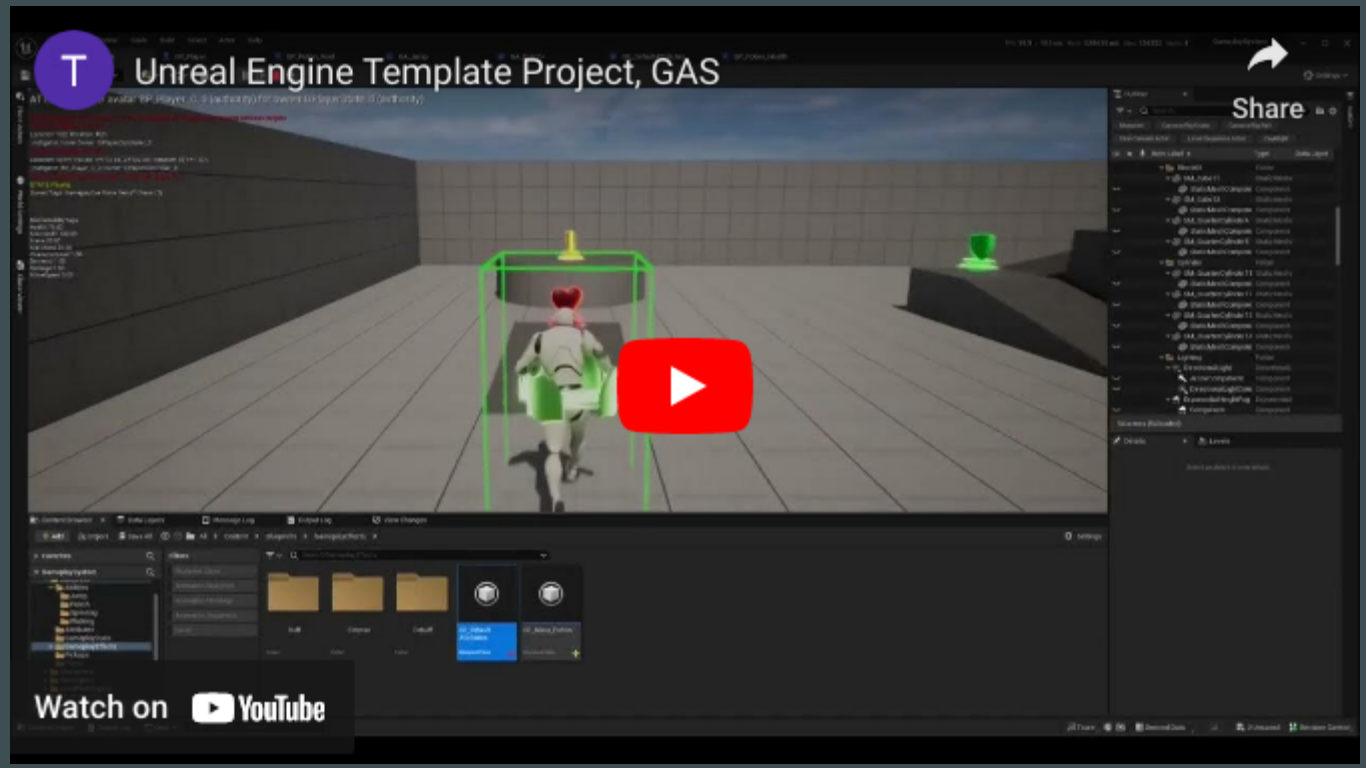
Applying the Effect

Finalizing the Overlap.

Specs carry specific data related to this Gameplay Effect.

6. First Interaction

Step 5: Item Interaction



[Video Link]

Step 1	Step 2	Step 3	Step 4	Step 5
Gameplay Effect	**Actor Blueprint**	**Visual Scripting**	**Applying the Effect**	**Placing the Object**
Health is an Attribute.	Overlapping Actors.	Logic behind Overlap.	Finalizing the Overlap.	Ready to be placed .
Gameplay Effects change our Attributes.	A simple pickup with a box collider attached.	Context carries information about the Gameplay Effect.	Specs carry specific data related to this Gameplay Effect.	Overlapping will heal the Player by the specific amount.

6. First Interaction

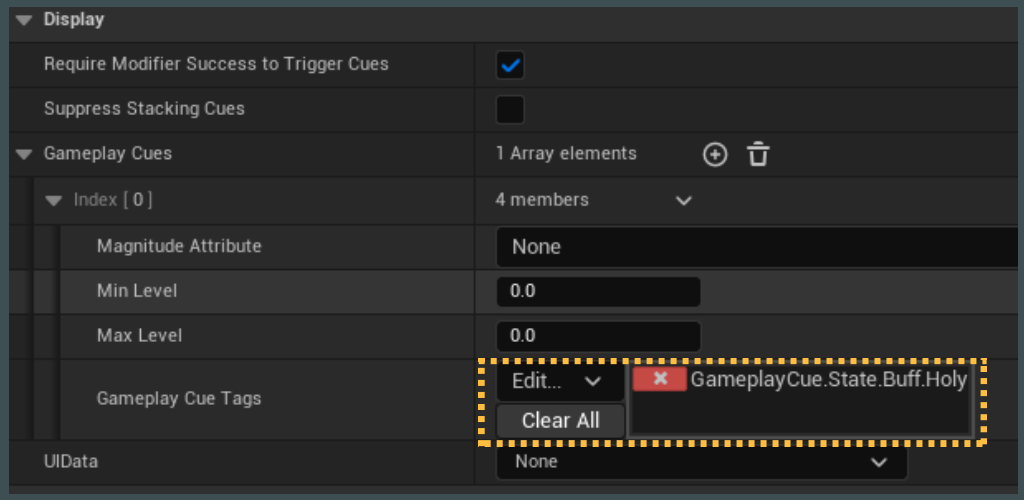
Enhancing visuals via Gameplay Cues

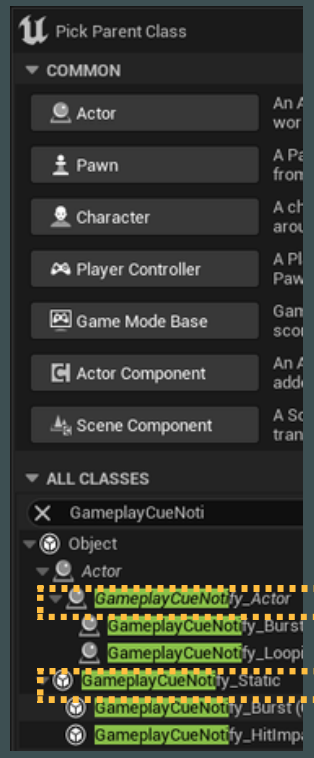
Gameplay Cues

These steps are optional and not necessary for a basic GAS template project.

Gameplay Cues

Notify Actor or Static?





Gameplay Effect

Add the Gameplay Tag for the Cue that should be linked to this Effect.

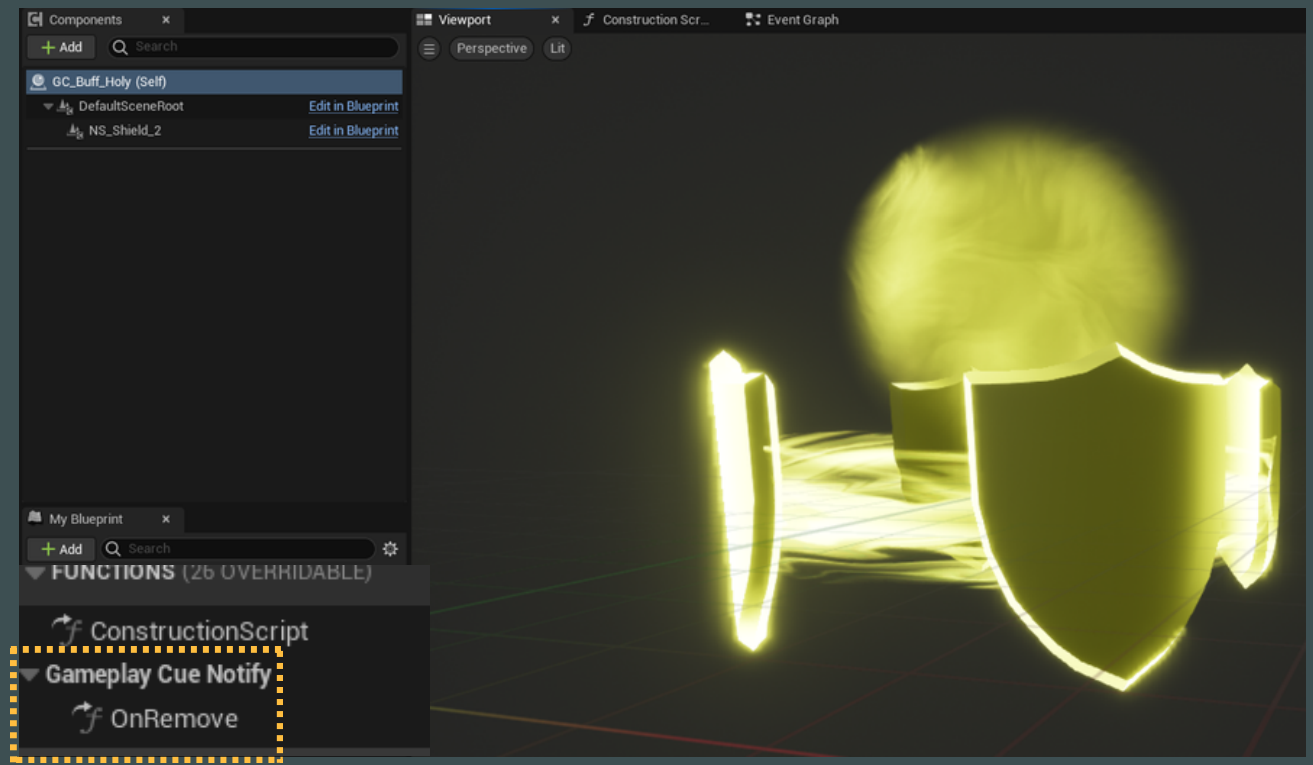
Gameplay Cues

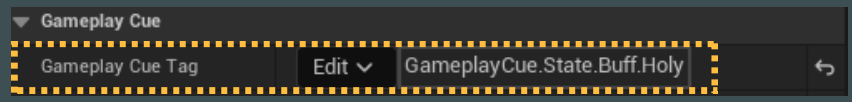
Notify is for an updating VFX.
Static is a one time activation VFX.

We choose Notify first.

Gameplay Cues

Linking Gameplay Cue and Effects





Gameplay Cues

Override the OnRemove function.

This will be called if one Effect cancels the other.

Gameplay Cue Tag

This is how the ASC knows which Cue to play.

GAMEPLAY ABILITY SYSTEM

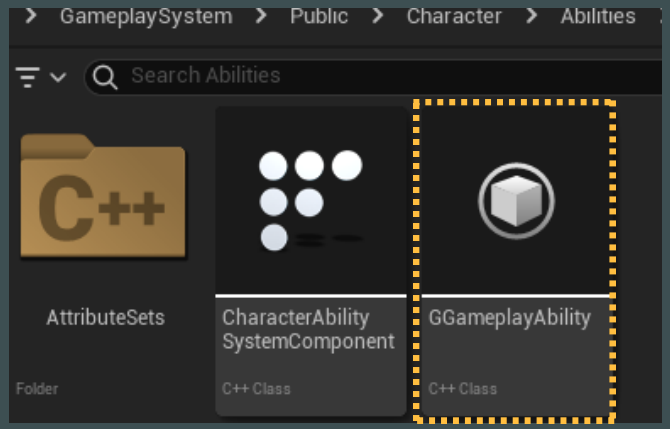
Creating Abilities

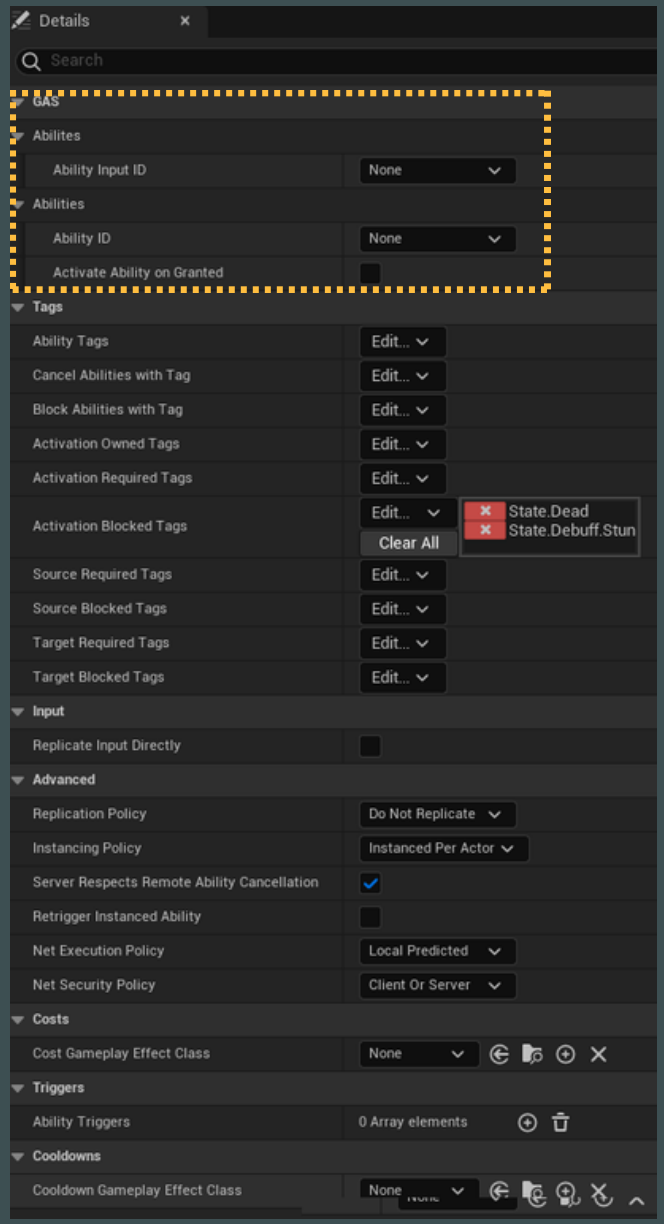
Creating Abilities

A good start for an Ability is the "Jump" functionality.
This allows us to focus on the Ability set up without needing a complex Blueprint logic.

7. Start Creating Abilities

Step 1: Creating a new Ability





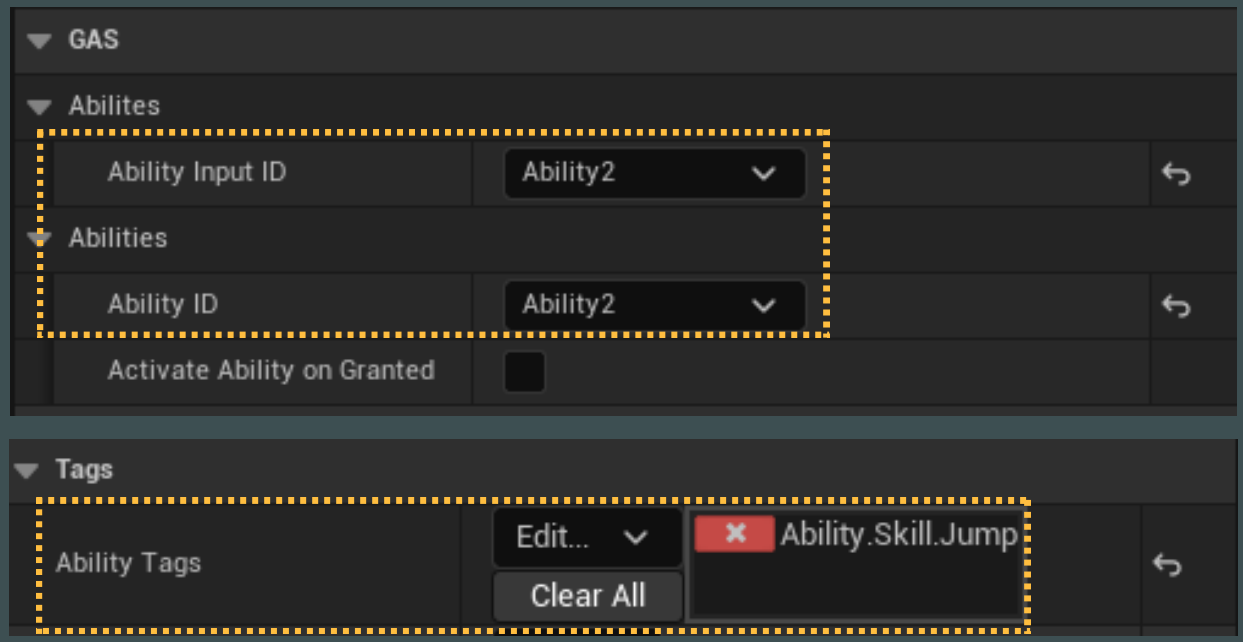
Gameplay Ability

Create a Blueprint class based on our own C++ Gameplay Ability.

Then you should have two fields for the InputID attached with the Ability Settings.

7. Start Creating Abilities

Step 2: Binding Input to Abilities



Ability Activation

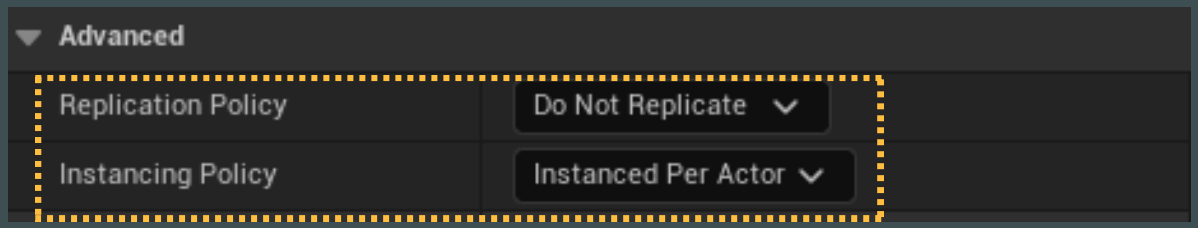
To activate an Ability you have to match the Ability Input ID with the Project Settings->Input.

Gameplay Tags

A Tag will be applied during an active Ability to know which one we are actively using.

7. Start Creating Abilities

Step 3: Replication



Multiplayer

Fill Replication and Instancing.

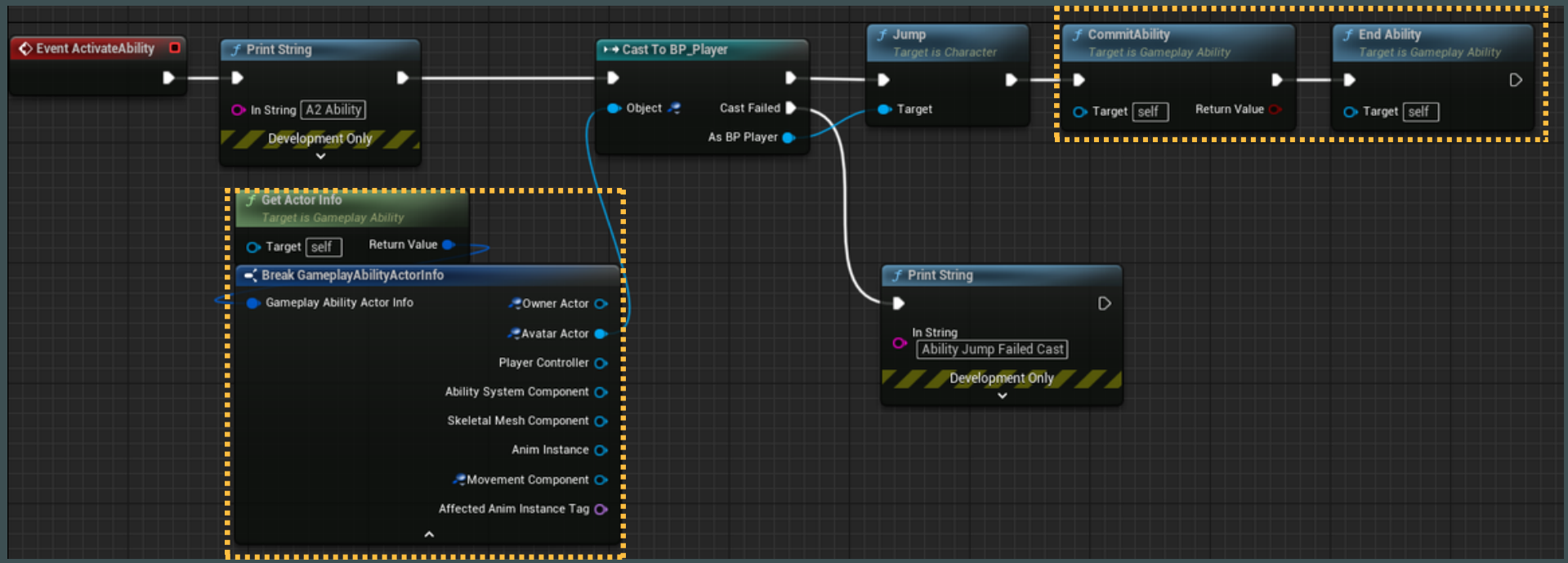
This makes a significant difference when going for a Multiplayer game.

Gameplay Tags

Check Network Compendium if you need help with the Network and Unreal Engine.

7. Start Creating Abilities

Step 4: Three mandatory Nodes



GetActorInfo

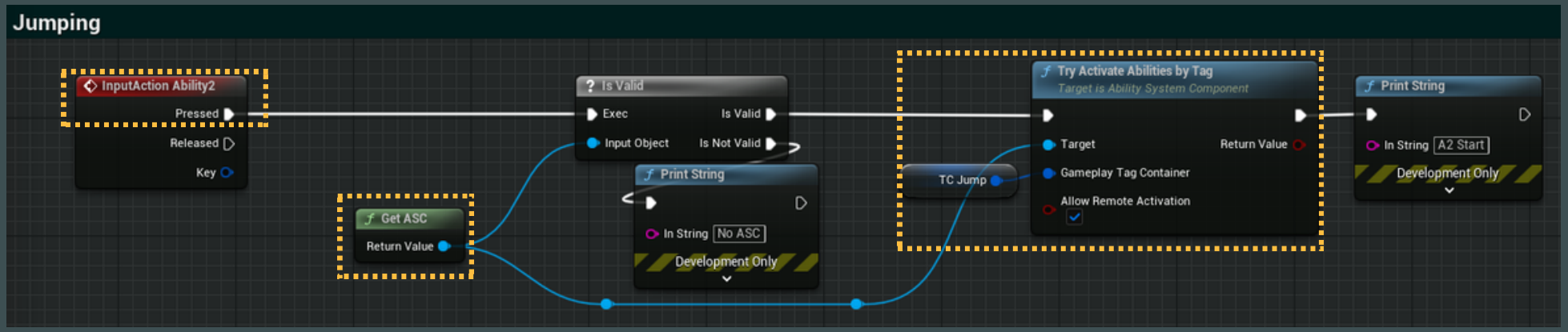
This struct will give us a lot of useful data about which we need to handle Abilities.

Commit, End Ability

Committing is useful when adding Costs for the Ability.
Always check that you End the Ability.

7. Start Creating Abilities

Step 5: Player calling the Ability



Input, Activation

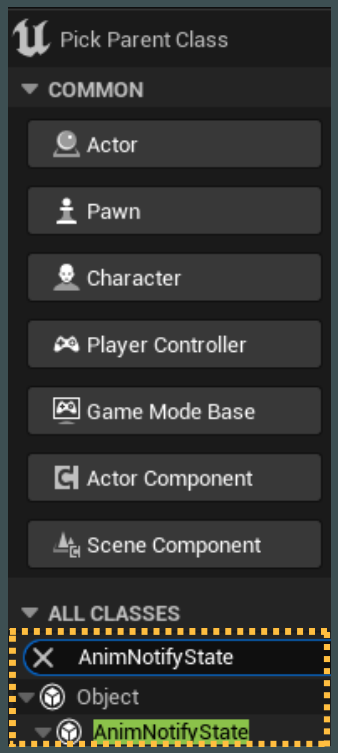
We get the ASC from the owner character.
And then using the Gameplay Tags to call the choosen Ability.

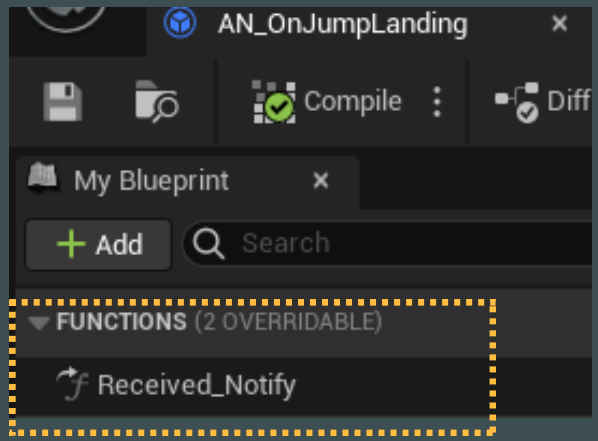
Gameplay Tag Container

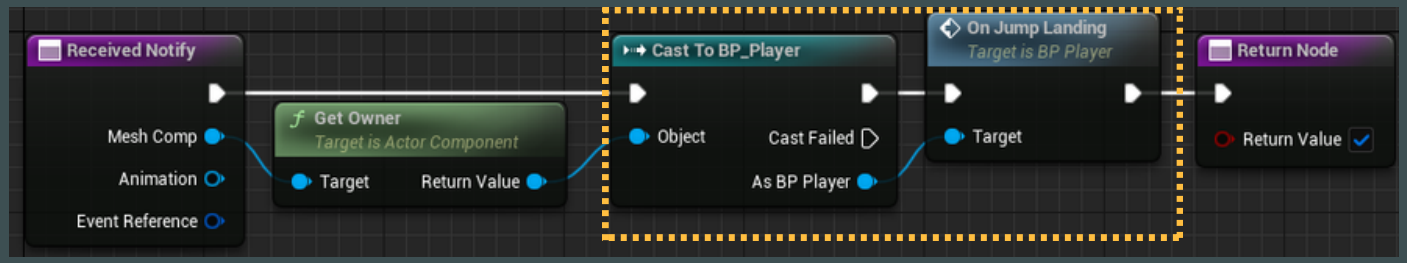
Holding references to all our Gameplay Tags.
There are different ways to activate an Ability, we use Tags for now.

7. Start Creating Abilities

Step 6: Animation Notify Event







Animation Notify

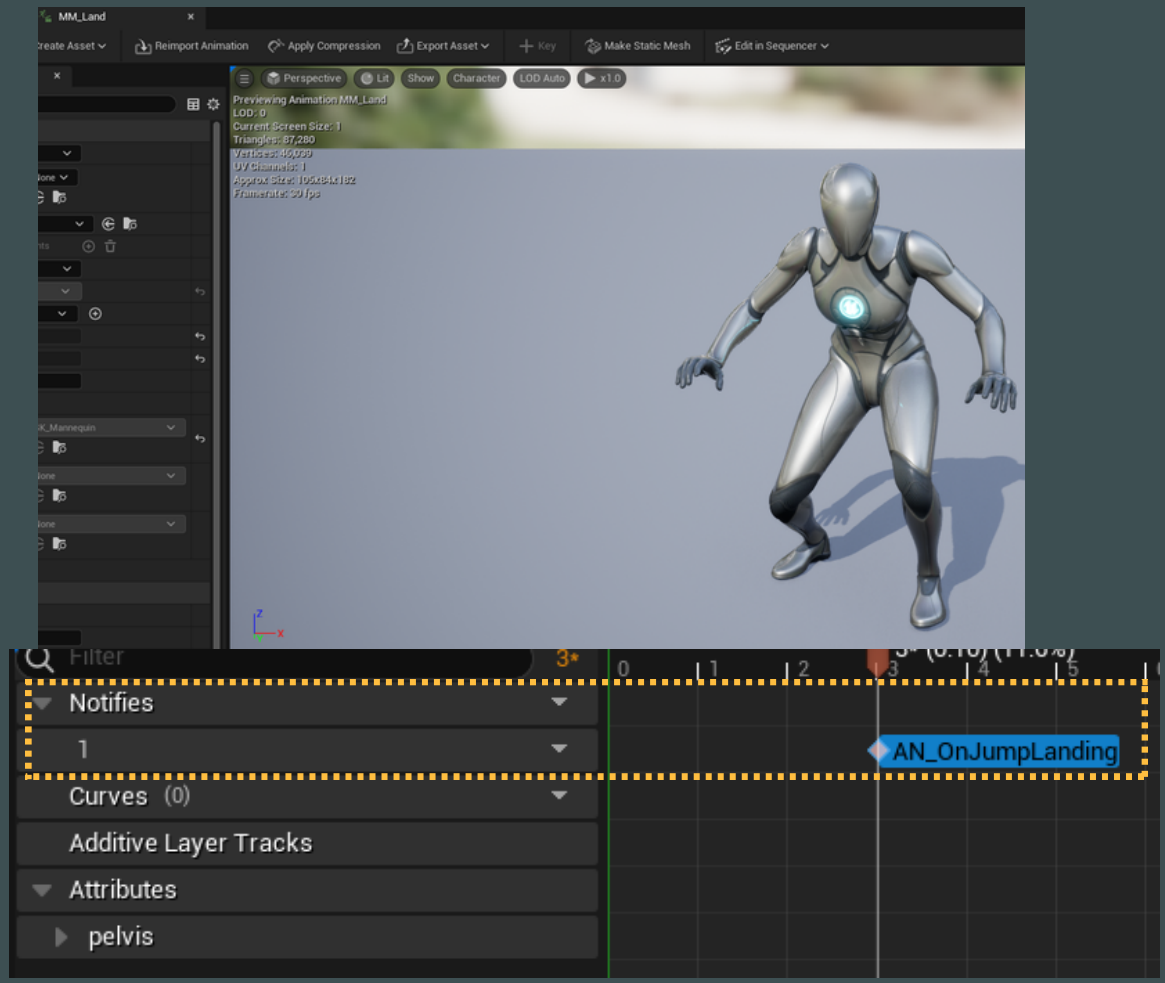
We create a new Blueprint class and override the Received_Notify function.

Event Callback to Player

This will call an Event inside the Player which is called from the Animation Blueprint.

7. Start Creating Abilities

Step 7: Adding the Notify Event

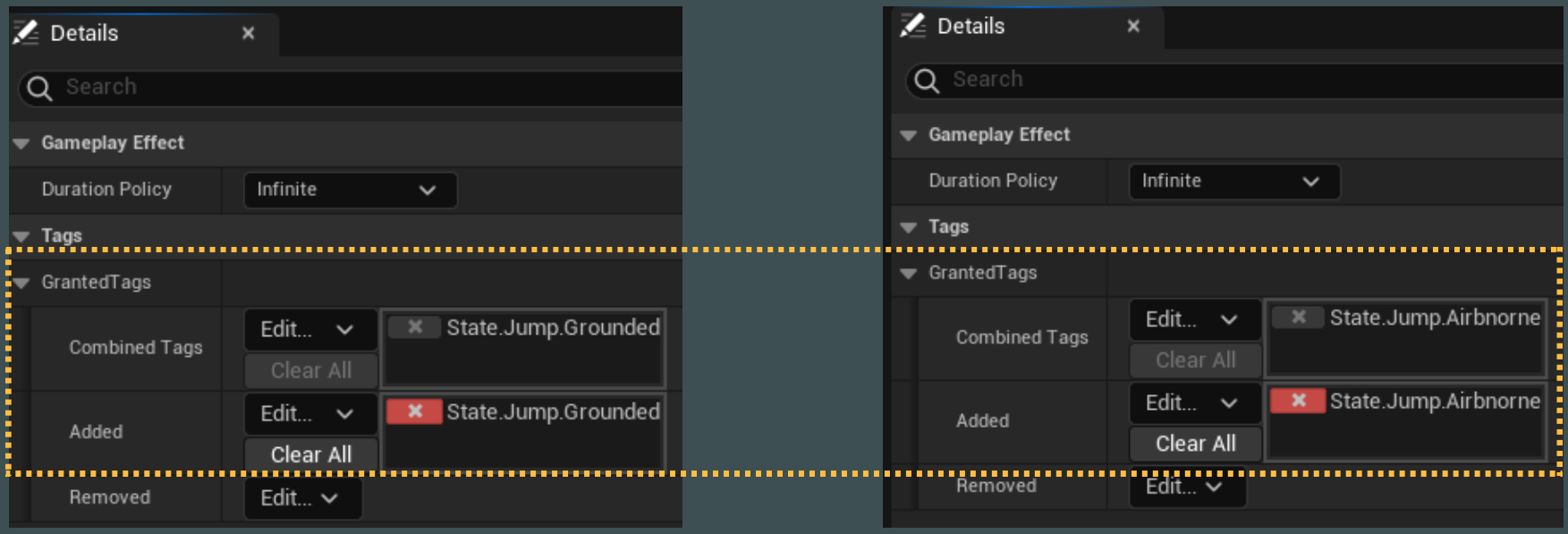


Animation Notify

Inside the Animation Blueprint we add our own Notify.

7. Start Creating Abilities

Step 8: Simple Effects for States

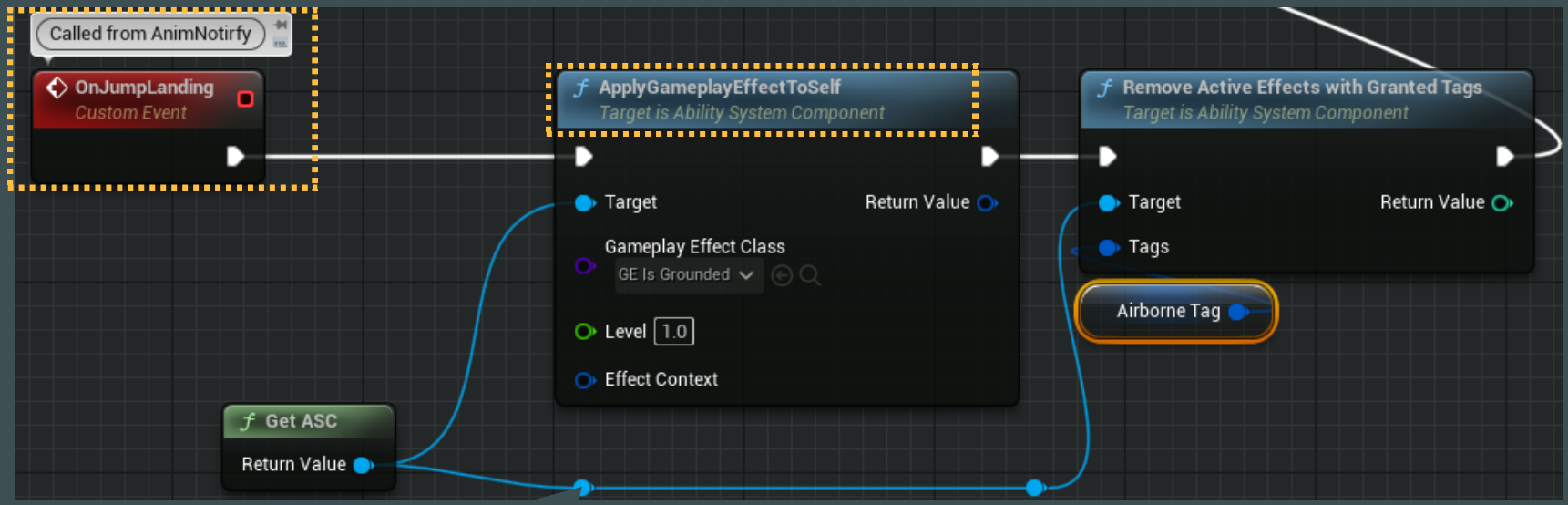


Granting Tags

Just by granting Tags you can handle Abilities and Player actions.

7. Start Creating Abilities

Step 9: Adding & Removing

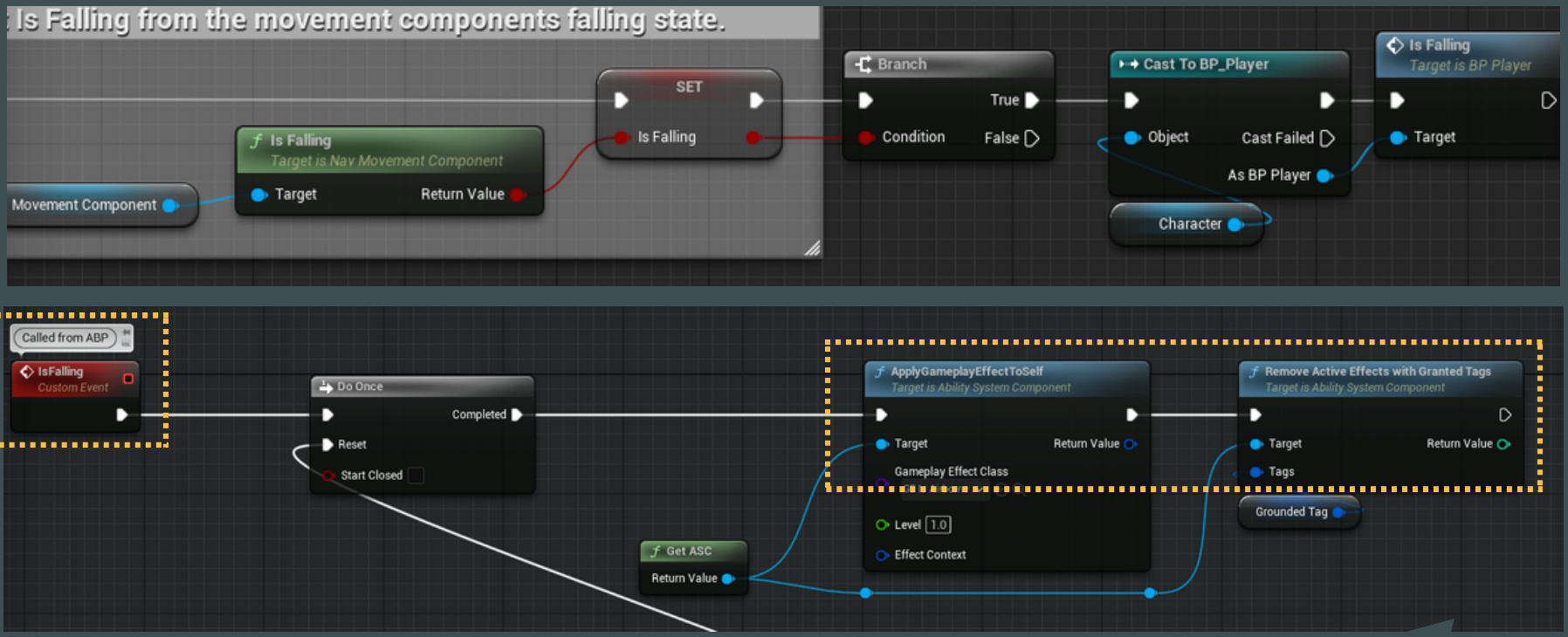


Gameplay Tags

Creating two simple GE for "IsGrounded" and "IsAirborne" is an easy way to allow Jumping

7. Start Creating Abilities

Step 10: Adding & Removing



Animation Blueprint

Call the Event from everywhere, even from the ABP.
This makes the GAS fit seamless into the existing Framework.

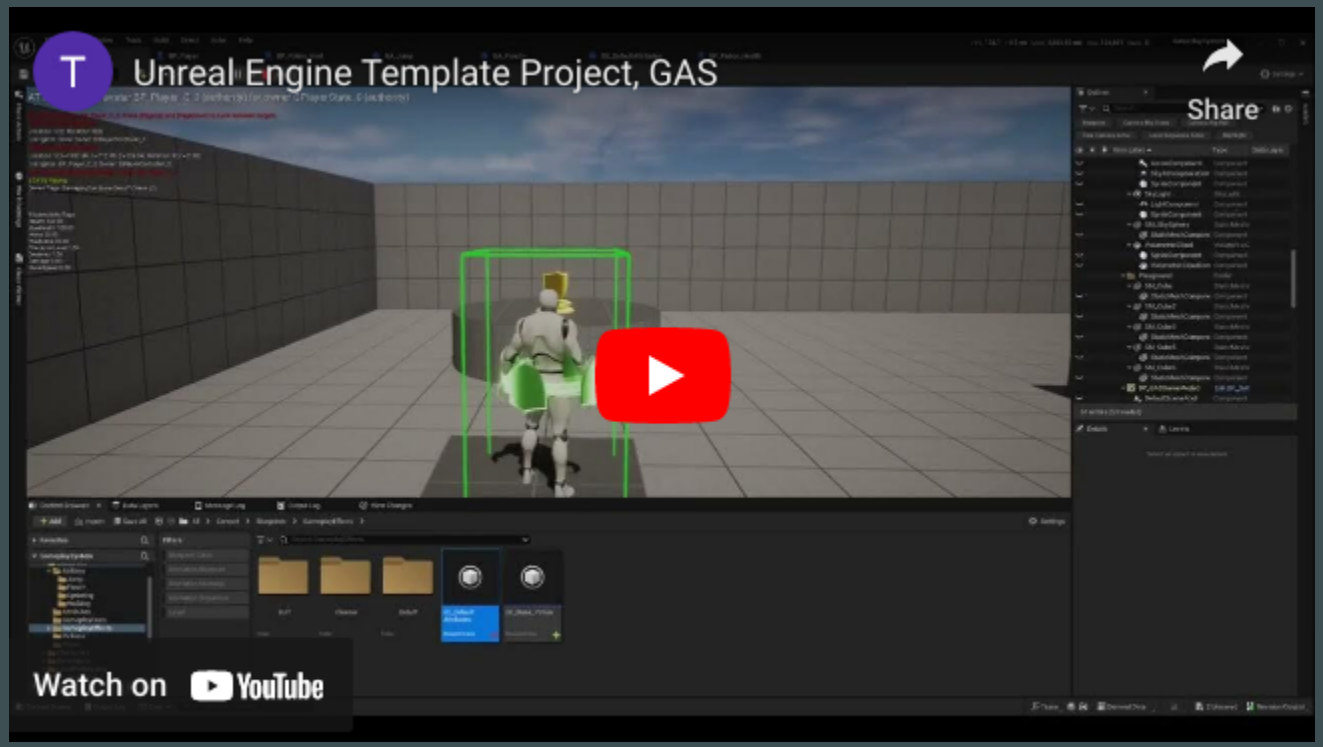
Adding and Removing

Use the Gameplay Tag Containers.

Just make sure that you add and remove the right Gameplay Tags.

7. Start Creating Abilities

Step 11: Testing our Jump ability



[Video Link]

GAMEPLAY ABILITY SYSTEM

Personal Note

8. Personal Note
Last but not least

Personal Note

I hope this guide has helped you in getting an overview of what the GAS is and how to start working with it.

Please consider checking all the attribution of sources, there you will find more information and advanced topics for this system.

8. Personal Note
No code has been harmed during the process

My Learnings

References: Really takes time to read a lot of code and get up to speed.

Good exercise: reading code and daily routine to come back to a problem and solving it.

Beginning of the week: Unsure if I can solve the current task, but documenting and managing my working times helped me stay motivated and keep at it.

Focus: Shifted from a personal goal "Gameplay Prototype" to a community goal "Beginners Guide". This helps everyone who is interested in getting started with the GAS.

Finding solutions: Slowly reading the source code again and again. If there is a time when you still don't comprehend what, why or how. Feel free to ask questions or continue exploring related topics to compare and reconsider your initial approach.

8. Personal Note

My learnings and thoughts

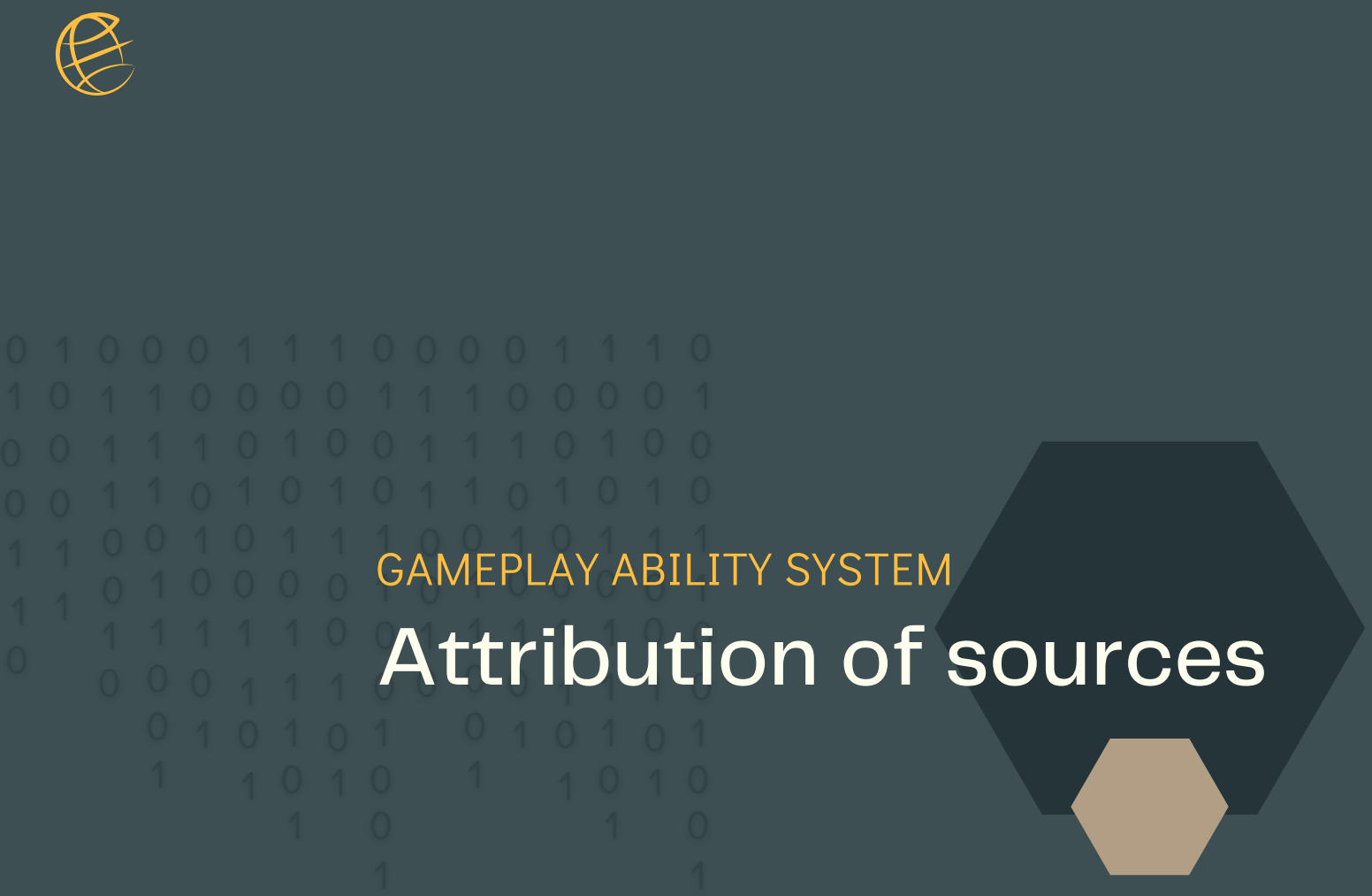
Is it worth investing your time to learn the GAS ?

TL;DR: The more it boils down to: dynamic changes, interactions, skills, progression, multiplayer and modularity it will be more beneficial in the long run to start with GAS.

Consider before using the GAS: Starting from scratch, means a lot of new stuff to learn and taking time to get something useful. Plan accordingly about implementing this system and if your production schedule will have benefits.

Good reasons for starting: Your game relies heavy on the usage of skills that are reacting to one and other and change different stats of the Player at the same time and you need Multiplayer support. Also the skills and abilities are 100% reusable for your next games and iterated very fast to make new ones.

Rethinking if: If your game is a small project like a jump and run, with just a few simple mechanics. You might lack the basic understanding of the Unreal Engine and C++ and have to prototype quickly.



GAMEPLAY ABILITY SYSTEM

Attribution of sources

Resources

Useful Links to help you learn more about the GAS

GASDocumentation
Github.com/tranek

A Guided Tour of GAS
Unreal Engine Talk

Exploring GAS
Unreal Engine Talk

ARPG Unreal v4.27
Demo Project

Using the GAS
Unreal Engine Talk

GAS Getting Started
Video Tutorial

Patong51
GAS Content Repo

Gameplay Abilities API
Unreal Engine Docs

Balancing C++ and BP
Unreal Engine Docs

Multiplayer Network Compendium
cedric-neuenkirchen

GAMEPLAY ABILITY SYSTEM

Appendix

Appendix

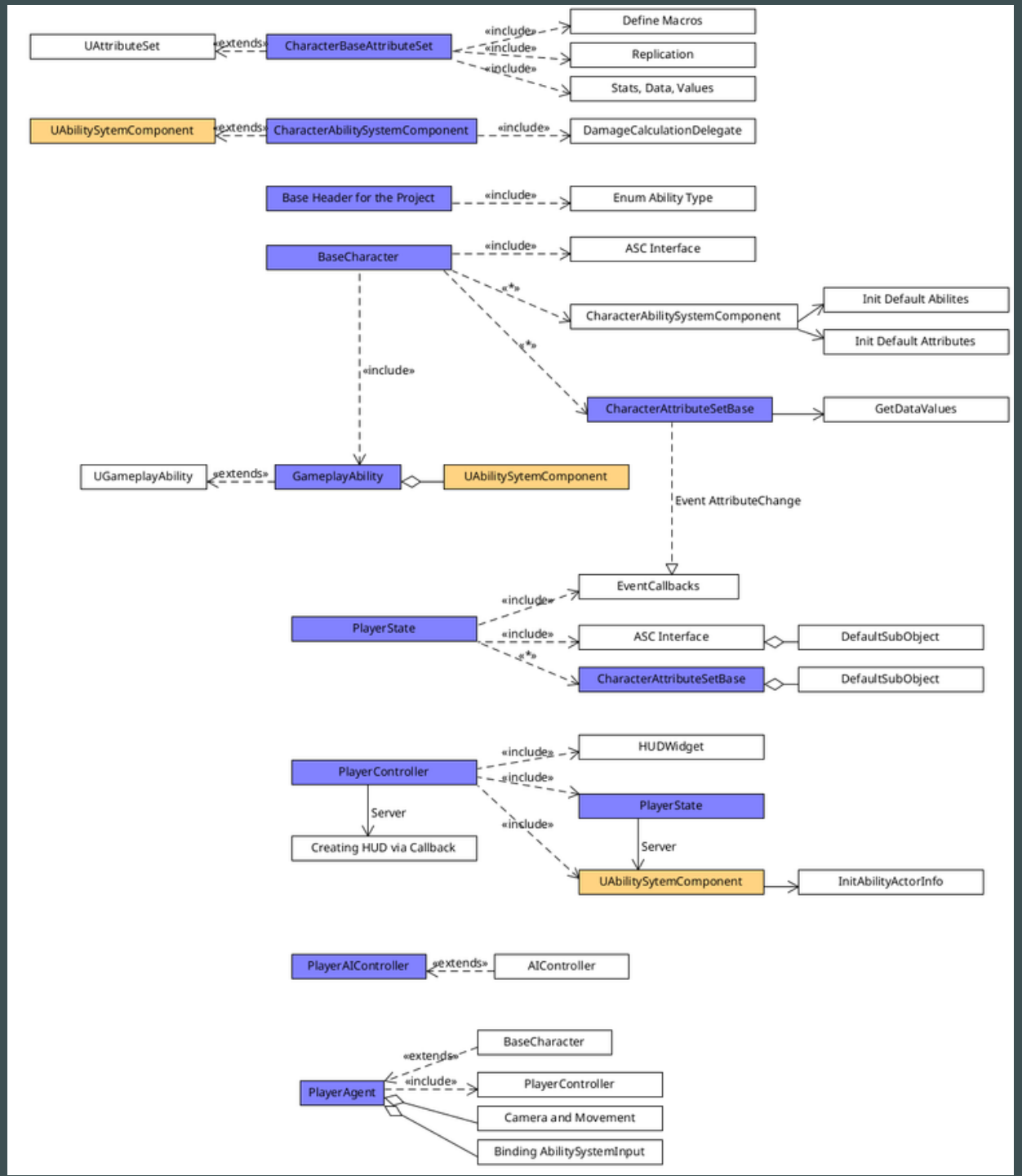
Some additional information about this guide and the GAS template project.

10. Appendix

Broader picture

High-Level Class Diagram

My visual representation of the GAS and its component relationship.



10. Appendix

Step by Step to Prototype

Step 1

Initialization

Foundation.

Important helper functions and dependencies.

Step 2

Ability System Component

Brain of the System.

Every Actor with Abilities must attach an ASC.

Step 3

Project Header

Adding Input-ID's for our Abilities.

If you make a new active Ability, you add its identifier here.

Step 4

Gameplay Ability

Abilities inherit from this class.

The Blueprint has a field to assign a corresponding InputID.

Step 5

Base Character

Initializes, Removes and sets Abilities.

References the ASC, AttributeSetBase and has an Interface.

Step 6

ASC Interface

Main purpose is to get the ASC.

If you add an ASC then you also have to add this Interface!

Step 7

Player State

Also implements the ASC Interface.

Opting for a Multiplayer Game? The ASC should be attached here.

Step 8

Player Avatar

A Blueprint class, our final Player.

Defining our Movement and binding Inputs for the ASC.

Step 9

Default Attributes

Health, Mana, Stamina etc.

Initialize once to enable changing them at runtime.

Step 10

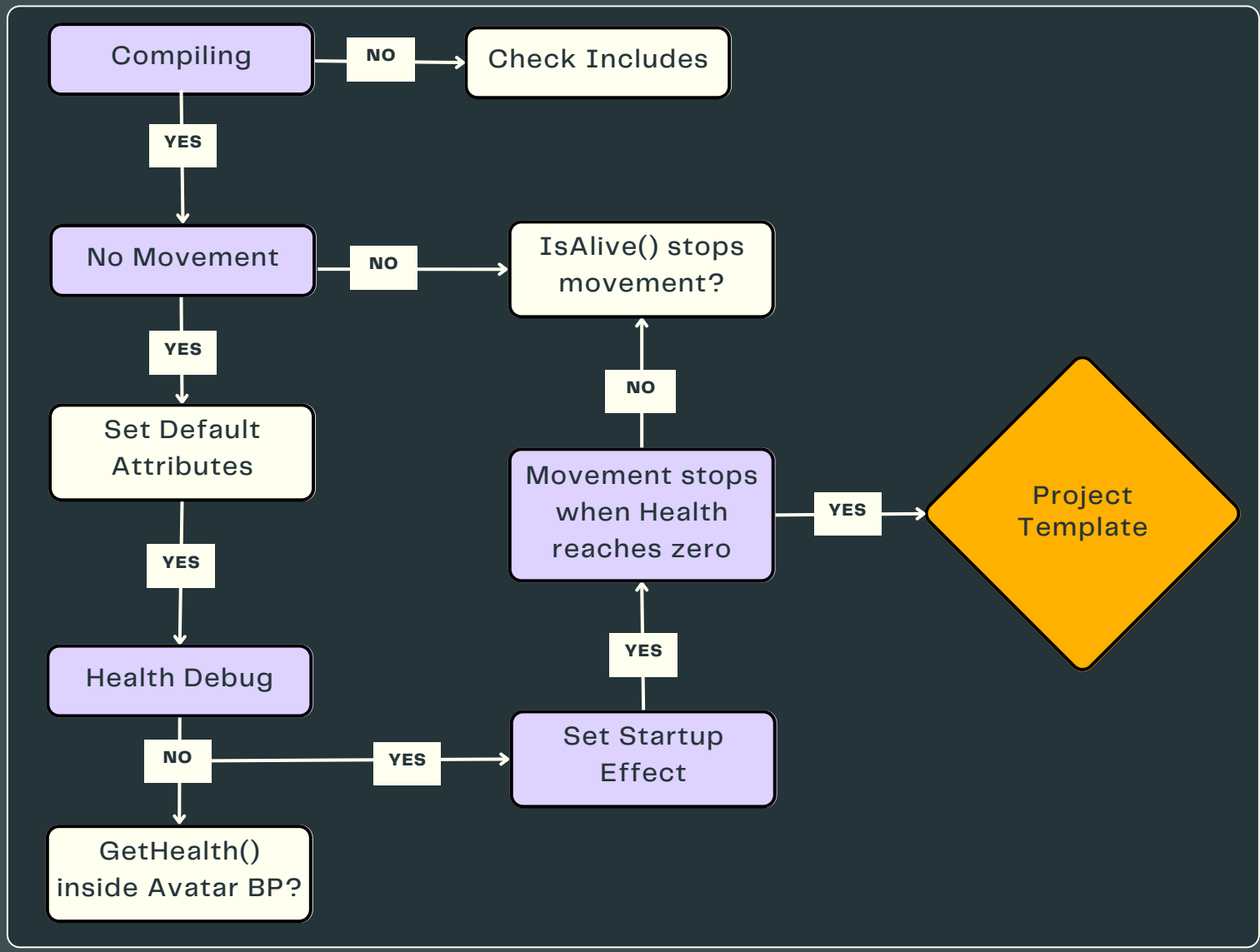
Interaction

Using our primary Attributes.

Creating simple actions between Abilities.

10. Appendix

Are you stuck? Test this out!



10. Appendix
Helpers & Tools

Software

Every end is a new beginning

I hope my guide has given you a better understanding about GAS and broken down its components even further. It should have made it more accessible and easier to get started in creating your games.

This guide was made by TheBitFossil aka René Hecker and is available here.

You have reached the end of this guide

by TheBitFossil aka René Hecker

Mai 2023

S4G School For Games, Berlin