> **NOTE**
>
> Join the Astra RPG Discord server!
> There is now a dedicated **Discord server** for Astra RPG Framework and its extensions. Join to **receive notifications** about new extension releases and important updates, **ask for new features**, **report bugs**, **share ideas**, and **showcase your Astra creations** with other developers.
>
> **Join the Discord Server: https://discord.gg/nJVRMkGrZg**

Introduction

Astra RPG Health extends the base framework, [](Astra RPG Framework), by adding functionality for managing health and calculating damage for entities. The package is designed with the same design philosophy as the base asset: a Scriptable Object-based architecture to encourage flexibility, modularity, and testability. If you're already familiar with the base package, you'll feel right at home with its features.

You can define your own damage types, designate defensive statistics used to reduce each damage type, configure the damage calculation pipeline with the desired steps, configure lifesteal for certain damage types, define strategies to execute upon entity death, and much more.

Astra RPG Health Vocabulary

Damage Type

Damage types represent the different categories of damage that can be inflicted on entities. Common examples include "Physical", "Magical", "Bleeding", "Drowning", etc. You can create custom damage types to suit your game's needs.

Damage Source

Damage sources represent the origin of the damage inflicted. This is highly specific to your game's context. For example, one game might have damage sources such as "Entity", "Environment", "Potion", etc., while another might want to define more specific damage sources like "Attack", "Spell", "Equipment", "Trap", "Environment", "Damage Over Time", etc. The main difference between damage source and damage type is that damage types categorize damage based on its nature, while damage sources identify where the damage originates. The distinction can be subtle, but it's important for game logic and mechanics. We'll return to these concepts later, where we'll see the practical differences between the two.

Heal Source

Heal sources represent the origin of healing. Similar to damage sources, heal sources are specific to your game's context. Common examples include "Potion", "Spell", "Lifesteal", "Ability", "Environment", etc. In

some games, a classic Heal Source definition might include "Self" and "Ally", as these enable mechanics for increasing healing provided/received based on the caster and target.

Barrier

The concept of temporary HP can take various names across different games, though the underlying mechanic remains the same: provide an amount of extra and ephemeral hit points that are deducted instead of health when damage is taken. In Astra, these temporary hit points are called "Barrier".

Raw and Net Damage

Raw Damage refers to the damage that a certain attack or skill intends to inflict. This damage does not account for resistances, critical hits, modifiers, etc. Net Damage is the result of processing Raw Damage while accounting for damage modifiers, resistances, barriers, critical hits, etc.

Damage Modifiers

Damage modifiers are components that can alter calculated damage in various ways. They can be used to implement mechanics such as damage reduction, damage increase, resistances, vulnerabilities, and more. Damage modifiers are generally utilized by the damage calculation pipeline (which we'll see shortly).

How is Astra RPG Health organized and how does it work?

Astra RPG Health Config

The `AstraRPGHealthConfig` is a `ScriptableObject` that serves as the central configuration point for the Astra RPG Health package. It has several properties that allow you to define how the health and damage systems should behave in your game.

With Astra RPG Framework, no configuration was needed. However, Astra RPG Health needs to be configured to work around the actual instances of the base framework's components defined for your game. For example, if you defined a certain statistic for general damage reduction in your game with Astra RPG Framework, you need to inform Astra RPG Health about it so that it can use it when calculating damage. The needed configuration is kept minimal, and convention-over-configuration is applied where possible to reduce the amount of setup required.
Configuration will be deeply discussed later in Package Configuration.

EntityHealth

`EntityHealth` is a brand new `MonoBehaviour` that you can add to your entities to provide them with health management capabilities. Worth mentioning, it exposes the most important method of the package: `TakeDamage()`, which allows you to inflict damage on the entity.

Damage Type

`DamageType` is a `ScriptableObject` that represents a specific type of damage. Each damage type can be optionally configured with a defensive `Stat` that will be used to reduce incoming damage of that type. If a defensive stat is assigned, also a piercing stat can be assigned to ignore a portion of the defense when calculating damage.
Both for the defensive and piercing stat, you can select a `DamageReductionFormula` and a `DefenseReductionFormula` respectively, to define how the stats will affect damage reduction and defense piercing.

Damage Source

`DamageSource`, derived from `ScriptableObject`, represents the origin of the damage inflicted. They don't have any specific properties, but they can be assigned to other objects of the package to create specific behaviors based on the damage source. We will see for what and how in the Workflows section.

Damage Reduction Functions

The package comes with three built-in `DamageReductionFunction`s that you can use to define how defensive stats reduce incoming damage:

- `FlatDamageReductionFn`: Reduces damage by a flat amount based on the defense stat value.
- `PercentageDamageReductionFn`: Reduces damage by a percentage based on the defense stat value.
- `LogarithmicDamageReductionFn`: Reduces damage using a logarithmic scale based on the defense stat value.

In case of need, custom damage reduction functions can be created by extending the `DamageReductionFn` class.

Defense Piercing Functions

As for damage reduction functions, the package comes with three built-in `DefensePiercingFunction`s that you can use to define how piercing stats ignore a portion of the defense stat:

- `FlatDefensePiercingFn`: Ignores a flat amount of the defense stat based on the piercing stat value.
- `PercentageDefensePiercingFn`: Ignores a percentage of the defense stat based on the piercing stat value.
- `LogarithmicDefensePiercingFn`: Ignores a portion of the defense stat using a logarithmic scale based on the piercing stat value.

Also in this case, custom defense piercing functions can be created by extending the `DefensePiercingFn` class.

Heal Source

`HealSource`, derived from `ScriptableObject`, represents the origin of healing. Similar to `DamageSource`, they don't have any specific properties, but they can be assigned to other objects of the package to create specific behaviors based on the heal source. We will see for what and how in the Workflows section.

Damage Calculation Strategy

The damage calculation pipeline is the component of the framework responsible for processing raw damage and producing net damage. The pipeline will use a given `DamageCalculationStrategy` to determine the sequence of steps to apply when calculating damage.
Therefore, a `DamageCalculationStrategy` is a `ScriptableObject` that defines a sequence of `DamageStep`s to be executed in order when calculating damage.

Death Strategy

A `DeathStrategy` is a `ScriptableObject` that, by relying on the strategy programming pattern, allows you to define different behaviors upon entity death.
When an entity's health reaches zero, a `DeathStrategy` is used to determine what actions should be taken.
Strategies can be game-oriented, such as "Respawn", "Game Over", or "Drop Loot", or they can serve more technical purposes, like "Disable GameObject", "Play Death Animation", or "Return to Pool", etc.

Astra RPG Health includes several predefined technical death strategies:

- `DisableOnDeathStrategy`: Disables the entity's GameObject upon death. Useful if the entity can be resurrected or reused later.
- `DestroyOnDeathStrategy`: Destroys the entity's GameObject upon death. Useful for entities that should not be reused. This is simpler and more immediate than using pooling when pooling is not needed.
- `DoNothingOnDeathStrategy`: Performs no action upon death. Useful for testing or debugging.
- `MultipleOnDeathStrategy`: Sometimes, you need to combine multiple strategies to achieve the desired behavior. For example, for enemies, you might want to drop loot, play a death animation, and destroy the GameObject (or return it to a pool if available). For this purpose, the framework includes a strategy that allows you to combine multiple death strategies in sequence.

You can also create custom death strategies by extending the `OnDeathStrategy` class. This allows you to implement death behaviors specific to your game's architecture and combine them with the predefined strategies in a `Multiple Death Strategy` if needed to achieve the desired result.

Resurrection Strategy

Similar to `DeathStrategy`, a `ResurrectionStrategy` is a `ScriptableObject` that allows you to define different behaviors upon entity resurrection.
When an entity is resurrected, a `ResurrectionStrategy` is used to determine what actions should be taken.
Astra RPG Health includes some predefined resurrection strategies:

- `EnableOnResurrectionStrategy`: Enables the GameObject of the entity upon resurrection. Useful if the entity's GameObject was disabled upon death.
- `DoNothingOnResurrectionStrategy`: Performs no action upon resurrection. Useful for testing or debugging.
- `MultipleOnResurrectionStrategy`: Similar to the multiple death strategy, this strategy allows you to combine multiple resurrection strategies in sequence.

Also in this case, you can create custom resurrection strategies by extending the `OnResurrectionStrategy` class to implement resurrection behaviors specific to your game's architecture.

Lifesteal Configuration

`LifestealConfig`, deriving from ScriptableObject, allows you to define how lifesteal mechanics work for specific damage types. This allows to bind a statistic to each damage type you want to have lifesteal for. That statistic will be used to calculate the amount of health to restore to the attacker when they deal damage of that type.
The configuration allows also to configure the damage pipeline timing of the lifesteal effect. For example, you might want lifesteal to occur before or after damage reduction is applied. We will see this in detail later in the Workflows section.

Health Scaling Component

Astra RPG Health provides a brand new `HealthScalingComponent` that you can use in your `ScalingFormula`s to have skills or abilities scale based on either the attacker or the target's health. You can choose to scale upon one or more among Maximum HP, Current HP, and Missing HP.

More Game Events

Astra RPG Health comes with many new Game Events that you can use to react to health&damage-related events in your game. Some of the most important ones are:

- `PreDamageGameEvent`: Triggered before damage is applied to an entity. Useful for modifying or canceling damage. Use this for implementing custom passives or effects that need to react before damage is taken.

- `DamageResolutionGameEvent`: Triggered when an entity takes damage. Can be used to react to damage being applied.
- `EntityDiedGameEvent`: Triggered when an entity dies.
- `EntityHealedGameEvent`: Triggered when an entity is healed.
- `EntityResurrectedGameEvent`: Triggered when an entity is resurrected.

And many more events. We will discuss them in detail later in the Workflows section.

Package Configuration

Astra RPG Framework needed no configuration. The health package, however, needs to be configured to have the system work around the specific instances of your game. For example, if you defined a "Super Duper All-damage resistance" `Stat` in your game, you need to tell Astra RPG Health to use it when calculating damage.

The package uses a flexible configuration system that balances convenience with explicit control. This page explains how to set up and configure the health system for your game.

Configuration Overview

The Astra RPG Health system uses a **two-tier configuration architecture**:

1. **Global Settings** (`AstraRpgHealthGlobalSettings`) - A lightweight pointer stored in `Resources` that references your active configuration
2. **Gameplay Configuration** (`AstraRpgHealthConfig`) - The actual configuration containing all gameplay parameters

This separation allows you to:

- Switch between different configuration profiles easily (e.g., for testing, different game modes)
- Keep configuration data separate from the loading mechanism
- Support convention-based fallbacks for quick prototyping

Global Settings

Automatic Setup

The package automatically creates the Global Settings asset on first import or when the editor loads. You don't need to do anything manually.

What happens automatically:

1. The `AstraRpgHealthGlobalSettings.asset` is created in `Assets/Resources/`
2. If a default configuration exists (e.g., from imported samples), it's automatically assigned
3. The system is immediately ready to use

> 📁 **Default Location:** `Assets/Resources/AstraRpgHealthGlobalSettings.asset`

Project Settings

You can manage the health system configuration through Unity's Project Settings window:

1. Open **Edit → Project Settings**
2. Navigate to **Astra RPG Health**
3. Assign your desired **Active Config Profile**

Status Indicators:

- ✅ **Gray "Using Explicit Configuration"** - A configuration is explicitly assigned
- ⚠️ **Yellow "Using Fallback"** - No explicit configuration; using convention-based fallback
- 🛑 **Red "No Configuration Found"** - Critical: No configuration available

Quick Actions:

- **Create New Config Asset** - Opens a save dialog to create a new configuration

Convention Over Configuration

The package follows a **convention-over-configuration** philosophy to reduce setup friction:

Fallback Resolution

If no explicit configuration is assigned in Project Settings, the system automatically searches for a configuration named:

> `Astra Rpg Health Config`

located in any `Resources` folder in your project.

Search Order

The configuration provider uses a **three-step loading strategy**:

1. **Explicit Configuration** (Project Settings)
    - Loads `AstraRpgHealthGlobalSettings` from `Resources/AstraRpgHealthGlobalSettings`
    - If it has an `ActiveConfig` assigned, use it
2. **Convention-Based Fallback**
    - Searches for `Astra Rpg Health Config` in any `Resources` folder
    - Logs a warning indicating fallback usage
3. **Error State**
    - If neither is found, logs an error with instructions
    - System will not function until a configuration is provided

> 💡 **Tip:** For production projects, always use **explicit configuration** via Project Settings for clarity and control.

Configuration Loading Strategy

The health configuration is loaded lazily on first access and cached for performance:

```
// Automatically loads configuration on first access
var config = AstraRpgHealthConfigProvider.Instance;

// Pre-load during initialization to avoid runtime overhead
AstraRpgHealthConfigProvider.WarmUp();

// Force reload (useful for testing)
AstraRpgHealthConfigProvider.Reset();
```

When is the configuration loaded?

- Automatically before the first scene loads (via `RuntimeInitializeOnLoadMethod`)
- Lazily when first accessed via `AstraRpgHealthConfigProvider.Instance`
- Explicitly when calling `WarmUp()`

Creating Configuration Assets

If for any reason you need to create a new `AstraRpgHealthConfig` asset, you can do so via two methods:

Via Project Settings

1. Open **Edit** → **Project Settings** → **Astra RPG Health**
2. Unassign any existing configuration, if any
3. Click **Create New Config Asset**
4. Choose a save location
5. The new configuration is automatically assigned

Via Asset Menu

1. Right-click in the **Project Window**
2. Select **Create** → **Astra RPG Health** → **Configuration**
3. Name your configuration
4. Assign it in Project Settings or in the Global Settings asset

---

Health Configuration Reference

> [!INFO] In the `AstraRpgHealthConfig` asset, you can hover over each field to see a tooltip with a brief description.

The `AstraRpgHealthConfig` asset contains all gameplay parameters for the health system. Below is a detailed explanation of each field.

Health

Health Attributes Scaling

Type: `AttributesScalingComponent`
Required: No
Description: Defines how entities' maximum health scales based on character attributes (e.g., Vitality, Endurance).

> **See Also:** Astra RPG Framework Scaling documentation

Generic Heal Amount Modifier Stat

Type: `Stat`
Required: No
Description: The stat that modifies **all** healing received by an entity.

Stacking Behavior:

- Combines **additively** with Source modifications

How it works:

- The stat value represents a **percentage modifier**
- Positive values increase healing, negative values decrease it
- Example: A value of `25` means +25% healing received

Example:

- Base Heal: 100 HP
- Generic Heal Amount Modifier: 25 (means +25%)
- Final Heal: 100 * 1.25 = **125 HP**

Heal Source Modifications

Type: `Dictionary<HealSource, Stat>`
Required: No
Description: Maps each healing source (e.g., Skill, Trap, Environment) to a modifier stat.

Use cases:

- Create specialized healing boosts or reductions (e.g., "Resurrection healing is 50% more effective", "Healing from potions is reduced by 20%")

Stacking Behavior:

- Stats stack additively with Generic modifications

Example:

- Incoming Heal: 100 from resurrection
- Generic Heal Amount Modifier: +30% (means +30% healing received)
- Resurrection Heal Amount Modifier: +20 (means +20% healing received)
- **Total Heal Modification:** +30% +20% = +50% → Final Heal: **150**

Damage

Default Damage Calculation Strategy

Type: `DamageCalculationStrategy`
Required: No
Description: The default strategy used to calculate net damage when an entity doesn't specify its own strategy.

Purpose:

- Provides a default damage calculation strategy to use for "regular" entities. In most cases, this is sufficient
- Can be overridden per-entity for custom damage pipelines in their `EntityHealth` component

> **See Also:** Damage Calculation Pipeline documentation

Generic Damage Modification Stat

Type: `Stat`
Required: No
Description: A universal damage modifier that applies to **all damage received**, regardless of type or source.

Usage:

- Applied in the damage calculation pipeline if `ApplyDmgModifiersStep` is included in the used strategy

Stacking Behavior:

- Combines **additively** with Type and Source modifications

Example:

- Incoming Damage: 100
- Generic Damage Modification: -20 (means -20% damage taken)
- **Damage Reduction:** -20% → Final Damage: **80**

Damage Type Modifications

Type: `Dictionary<DamageType, Stat>`
Required: No
Description: Maps each damage type (e.g., Fire, Ice, Physical) to a modifier stat.

Usage:

- Applied in the damage calculation pipeline if `ApplyDmgModifiersStep` is included in the used strategy

Use cases:

- Configure different resistances/vulnerabilities per damage type **Stacking Behavior:**
- Stats stack additively with Generic and Source modifications

Example:

- Incoming Damage: 100 Fire damage
- Generic Damage Modification: +10 (means +10% damage taken)
- Fire Damage Modification: -30 (means -30% damage taken)
- **Total Damage Modification:** +10% - 20% = -10% → Final Damage: **90**

Damage Source Modifications

Type: `Dictionary<DamageSource, Stat>`
Required: No
Description: Maps each damage source (e.g., Skill, Trap, Environment) to a modifier stat.

Usage:

- Applied in the damage calculation pipeline if `ApplyDmgModifiersStep` is included in the used strategy

Use cases:

- Create specialized defenses (e.g., "Traps deal 50% less damage")
- For keeping track of damage origins for a combat log or analytics

Stacking Behavior:

- Stats stack additively with Generic and Type modifications

Example:

- Incoming Damage: 100 Physical damage from a Trap
- Generic Damage Modification: +30% (means +30% damage taken)
- Physical Damage Modification: -10 (means -10% damage taken)
- Trap Damage Modification: -50 (means -50% damage taken)
- **Total Damage Modification:** +30% -10% -50% = -30% → Final Damage: **70**

Health Regeneration

Passive Health Regeneration Source

Type: `HealSource`
Required: No
Description: The heal source used for passive regeneration effects.

Use cases:

- Allows tracking and modifying passive regeneration separately from active healing
- Can be used for effects like "Increase Natural Regeneration by 50%"
- Tracking passive healing in analytics or combat logs

Passive Health Regeneration Stat (HP/10s)

Type: `Stat`
Required: No
Description: Determines the amount of health regenerated passively.

> ⚠ **WARNING**
> The stat value represents health regenerated **per 10 seconds**.

Calculation:

```
Health Per Tick = (Stat Value / 10) * Interval In Seconds
```

Example:

- Stat Value: 50 HP/10s
- Interval: 1 second
- Health Per Tick: (50 / 10) * 1 = **5 HP per second**

Passive Health Regeneration Interval

Type: `float`
Default: 1.0 seconds
Required: Yes (must be > 0)
Description: The time (in seconds) between passive regeneration ticks.

Configuration Examples:

Interval	Stat Value	Result
1.0s	50 HP/10s	5 HP every 1 second
0.5s	50 HP/10s	2.5 HP every 0.5 seconds
2.0s	50 HP/10s	10 HP every 2 seconds

> **WARNING**
> Smaller intervals increase CPU overhead. Recommended range: 0.5s - 2.0s

Manual Health Regeneration Stat

Type: `Stat`
Required: No
Description: Determines health regenerated when triggering manual regeneration via API.

Use Cases:

- **Turn-based systems:** Regenerate health at the end of each turn
- **Rest mechanics:** Trigger regeneration when resting at campfires
- **Time-skip systems:** Apply regeneration for elapsed time

API Usage:

```
// Trigger manual regeneration
entityHealth.ManualHealthRegenerationTick();
```

Lifesteal

Lifesteal Config

Type: `LifestealConfig`
Required: No
Description: Configuration for lifesteal mechanics (healing based on damage dealt).

Typical Settings:

- Lifesteal percentage stat
- Heal source for lifesteal effects
- Restrictions (e.g., only on critical hits, only physical damage)

> **See Also:** Lifesteal documentation

Death

Default On Death Strategy

Type: `OnDeathStrategy`
Required: Yes
Description: The strategy executed when an entity dies (if the entity doesn't have its own strategy).

Common Strategies Ideas:

- **Destroy GameObject** - Removes the entity from the scene
- **Ragdoll** - Enables ragdoll physics
- **Respawn** - Respawns the entity after a delay
- **Loot Drop** - Spawns loot and destroys the entity

Example:

```
Death → Execute Strategy → Spawn Death VFX → Drop Loot → Destroy GameObject
```

Default On Resurrection Strategy

Type: `OnResurrectionStrategy`
Required: No
Description: The strategy executed when an entity is resurrected (if the entity doesn't have its own strategy).

Common Strategies Ideas:

- **Simple Resurrection** - Restores health and enables the entity
- **Resurrection VFX** - Plays visual effects during resurrection
- **Stat Penalties** - Applies temporary debuffs after resurrection

Default Resurrection Source

Type: `HealSource`
Required: ✅ **Yes**
Description: The heal source used when an entity is resurrected.

Use cases:

- Categorizes resurrection healing separately from normal healing
- Allows effects like "Increase Resurrection Healing by 50%"
- Used for analytics and gameplay feedback

Troubleshooting

⚠️ "No Configuration found!" error

Cause: No configuration is assigned and no fallback exists.

Solution:

1. Check **Project Settings** → **Astra RPG Health**
2. Assign a configuration or create a new one
3. Alternatively, create a config named `Astra Rpg Health Config` in a `Resources` folder

⚠️ "Using Fallback" warning

Cause: No explicit configuration assigned in Project Settings.

Solution:

1. Open **Project Settings** → **Astra RPG Health**
2. Assign the fallback configuration explicitly
3. This warning is just informational and won't break functionality

⚠️ Configuration not updating in Play Mode

Cause: Configuration is cached on first access.

Solution:

```
// Force reload
AstraRpgHealthConfigProvider.Reset();
```

⚠️ Missing Resources folder

Cause: The `Assets/Resources/` folder doesn't exist.

Solution: The package creates it automatically. If it's missing, create it manually:

1. Right-click `Assets`
2. Create → Folder → Name it `Resources`
3. Restart Unity to trigger the bootstrapper

Advanced topics

Limitations

Requirements

- Astra RPG Framework
- Unity 6 or later

Package Contents

Samples

Installation instructions

Importing Astra RPG Health and its samples

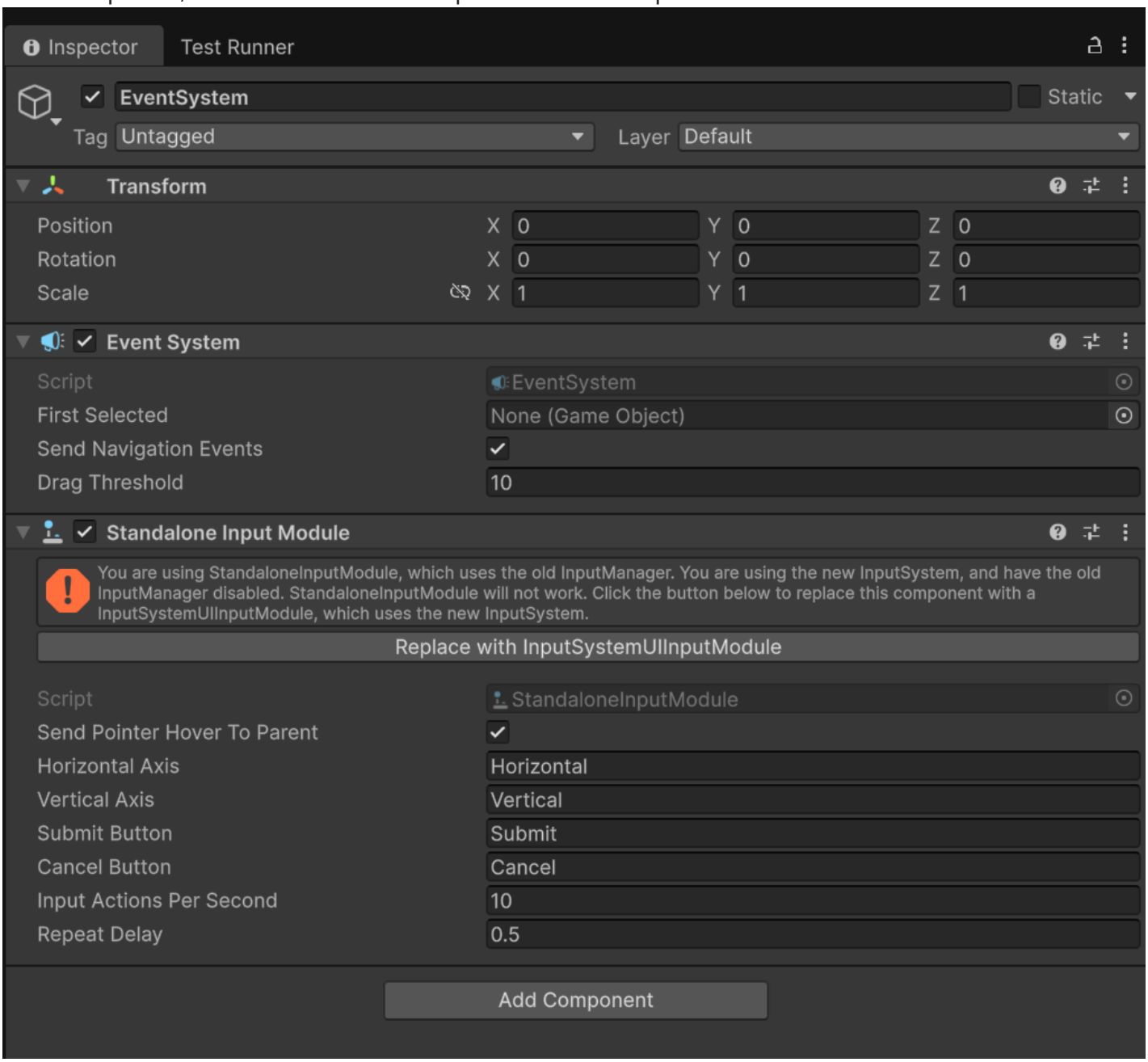
1. From the package manager, import Astra RPG HEalth. You can find the package in the "My Assets" section.
2. After importing, head to the "In Project" section, always in the Package Manager, and click on "AstraRPGHealth".
3. Click on the "Samples" tab and import the samples you desire. Find out more about the samples in the Samples documentation.
4. If you imported the "Example scene and instances" samples you need to import also TextMeshPro Essentials. Click on "Window > TextMeshPro > Import TMP Essential Resources".

Unity 6.2 and above extra steps for having the sample scene working

If you are using Unity 6.2 or above, and you imported the "Example scene and instances" samples, you need to do the following extra steps to have the sample scene working:

1. Open the "SampleScene" scene located in "Assets/Samples/AstraRPGHealth/[version of the package]/Example scene and instances/SampleScene".
2. Select the "EventSystem" GameObject in the Hierarchy.

3. In the Inspector, find the "Standalone Input Module" component. You should see an error like:



4. Click on "Replace with InputSystemUIInputModule".

The sample scene should now work correctly.

I am working on a fix to avoid having to do these extra steps in future versions of the package, while preserving compatibility with Unity versions below 6.2.

Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog.

[1.0.0] - 2025-10-30

Added

- Initial release of Astra RPG Health.

Migration Guide