



# Modding Documentation

Author: Nicholas Staracek

Version: 1.07

# Table of Contents

## [Introduction](#)

## [Section 1: Installing and Access + Basic Tips](#)

[What you need to do on the internet:](#)

[What you need to do in Steam:](#)

[What you need to do if you're on a PC system:](#)

[What you need to do in Unity:](#)

[What you need to do if you're on a Mac OSX or Linux system:](#)

[Explaining the Mod Editor window:](#)

[What does the "Edit" button do:](#)

[What does the "Build" button do:](#)

[What does the "Play" button do:](#)

[What does the "Publish" button do:](#)

[Updating your mod tools:](#)

[Basic shortcuts:](#)

[How do I access modded content in my game?](#)

## [Section 2: "The Basic Encounter" \(Tutorial\)](#)

[Creating the new encounter card:](#)

[Viewing the encounter card data:](#)

[Changing the name of the encounter card:](#)

[Assigning an art texture to the encounter card:](#)

[Assigning Traits to the Encounter:](#)

[Navigating the Encounter Behaviour Tree:](#)

[What are the different types of nodes we can create:](#)

[Creating the first node of the encounter:](#)

[Creating a choice for the player:](#)

[Adding the chance cards mini game:](#)

[Creating the chance cards:](#)

[Creating variables in the Local BB:](#)

[Creating the pain and gain results for the mini game:](#)

[Creating the text for the results of the mini game:](#)

[Activating the pain and gain cards:](#)

[Ending the encounter:](#)

[Afterthoughts and revision:](#)

## [Section 3: Creating "The Basic Equipment"](#)

[Creating new equipment cards:](#)

[Creating the Basic Sword:](#)

[Creating the Basic Artifact:](#)

[Creating the Basic Shield:](#)

[Testing your equipment mods:](#)

#### [Section 4: Creating “The Basic Trait”](#)

[Creating our new trait:](#)

#### [Section 5: “The Basic Challenge” \(Tutorial\)](#)

[Creating the new challenge card:](#)

[Challenge Card Data:](#)

[Changing the name and description of the challenge card:](#)

[Assigning an art texture to the challenge card:](#)

[Assigning our trait to the challenge card:](#)

[Creating the gold challenge token:](#)

[Creating the challenges Global BB:](#)

[Creating the variables in the Global BB:](#)

[Creating the challenge objective:](#)

[Incrementing the ‘Objective Channel Index’ & setting the ‘Objective State’ on the Entrance Card:](#)

[Setting up the data, for the ‘End’ behaviour tree, so that it’s compatible with our basic challenge mod:](#)

[Creating challenge decks and card sets:](#)

[What is a dungeon:](#)

[Creating the dungeon for the challenge:](#)

[Dungeon Data:](#)

[Assigning the config data for the dungeon:](#)

[Assigning the level exit card for the challenge:](#)

[Assigning the dungeon end card for the challenge:](#)

[Assigning the dungeon map layout:](#)

[How can I easily test a challenge in the Unity editor?](#)

#### [Section 6: “Intermediate Encounter Design”](#)

[What are Token Chains?](#)

[How can I design my Token Chains & Encounter Sets?](#)

[Adding tokens to your encounter cards:](#)

[Creating Basic Spices:](#)

[Adding timed combat and combat objectives:](#)

[Creating your own monster cards with custom values:](#)

[Awarding Fame after Encounters:](#)

[Setting player stats directly:](#)  
[How to create the other types of mini games in HoF2:](#)  
[How to create a dice gambit:](#)  
[How to create a pendulum mini game:](#)  
[How to create a wheel mini game:](#)  
[Creating Brimstone & Platinum Rarity Cards:](#)  
[Adding your own art as textures for your cards:](#)  
[Creating Loops in your behaviour trees:](#)

#### [Section 7: "Intermediate Challenge Design"](#)

[Creating a Silver token for the basic challenge:](#)  
[Creating the second challenge objective, linked to the silver token unlock for the basic challenge:](#)  
[Adding the 'Objective Channel Index' & 'Objective State' action tasks to the end of the Entrance Card, for the second challenge objective:](#)  
[Adding the 'Objective Channel Index' action task to the End Card of the basic challenge, for the second challenge objective:](#)  
[Creating an objective listener for the gold challenge objective:](#)  
[Creating your own dungeon map layouts:](#)  
[Creating your own dungeon map layout sets:](#)  
[Restricting Cards based on Rarity:](#)  
[Other dungeon config data:](#)  
[Using Dungeon Level Decks:](#)

#### [Section 8: Distribution of Content](#)

[Creating the Mod:](#)  
[Creating Addons & Addon Sets:](#)  
[How to Build & Publish mods to the Steam Workshop:](#)  
[How to update a mod that's already published on the Steam Workshop:](#)

#### [Section 9: HoF2 SDK - The Game Master's Toolkit FAQ](#)



# Change Log

Change made?	Location?	Date of Change
Added support info for adding your own art as card textures.	<a href="#">Adding your own art as textures for your cards:</a>	2018.02.06
Added support info for navigating encounter trees.	<a href="#">Navigating the Encounter Behaviour Tree:</a>	2018.02.07
Added support info for downloading mods after clicking subscribe.	<a href="#">How do I access modded content in my game?</a>	2018.02.07
Added support info for art texture requirements when you're adding your own art.	<a href="#">Adding your own art as textures for your cards:</a>	2018.02.07
Added support info for creating the mod that you build and publish.	<a href="#">Creating your Mod:</a>	2018.02.07
Added FAQ section into the documentation & a link to the section under 'Introduction'.	<a href="#">Section 9: HoF2 SDK - The Game Master's Toolkit FAQ:</a>	2018.02.08
Added support info for creating different types of Addons & Addon Sets..	<a href="#">Creating Addons &amp; Addon Sets:</a>	2018.02.08
Added support info for testing equipment mods easily.	<a href="#">Testing your equipment mods:</a>	2018.02.08
Update FAQ with answers.	<a href="#">Section 9: HoF2 SDK - The Game Master's Toolkit FAQ</a>	2018.02.09
Added an <b>important note</b> to the section where you end the encounter.	<a href="#">Ending the encounter:</a>	2018.02.09
Added support info about cloning accounts to keep your progress on a modded account.	<a href="#">How do I access modded content in my game?</a>	2018.02.12
Added support info for easily testing challenge mods in the Unity editor.	<a href="#">How can I easily test a challenge in the Unity editor?</a>	2018.02.12

Added support info for creating monster cards with custom values.	<a href="#">Creating your own monster cards with custom values:</a>	<b>2018.02.13</b>
Added support info for how to create your own card art for your mods via our provided templated and documentation.	<a href="#">Section 9: HoF2 SDK - The Game Master's Toolkit FAQ</a> <a href="#">Adding your own art as textures for your cards:</a>	<b>2018.02.14</b>
Updated Unity version.	-	<b>2018.04.09</b>

# Introduction

Welcome prospective modders! In this document you'll find a series of tutorials designed to acclimatize you to the process of creating many of the core components of a *Hand of Fate 2* (HoF2) challenge.

As you progress further, we explore some more intermediate level design tools, tricks and tips to help you create your own unique content for HoF2!

These modding tools require that you have version (2017.4.2f2) of the Unity Engine installed. You need to have this version specifically, as it's the version that both the game and editor project have been built with.

There is a **FAQ** section located at the end of this documentation, you can access it [here](#).

## Section 1: Installing and Access + Basic Tips

The HoF2 Modding Tools require you to have access to the Unity Game Engine. These tools are compatible with any Unity Game Engine license type, be it personal through to professional. These tools do not work standalone.

### What you need to do on the internet:

You'll need to download the Unity Engine (2017.4.2f2). You can download this specific version by clicking [here](#).

### What you need to do in Steam:

1. Open your Steam Client.
2. Navigate to your 'Library', located at the top of the Steam Client after logging in.
3. Download the 'Hand of Fate 2' game.
  - *Note:* If you've already done this, but you have an available update for the game, make sure you update the game.
4. Navigate to 'Library' and select 'Tools' from the dropdown menu. Within 'Tools' you'll find the 'Hand of Fate 2 SDK', you want to download this.
5. Once you've downloaded the 'Hand of Fate 2 SDK' proceed to the [What you need to do on your computer](#) steps.

### What you need to do if you're on a PC system:

Next, we'll navigate to your game files and extract the HoF2\_Modding project from within the Hand of Fate 2 SDK folder.

The game files are usually kept here: Local Disk (C:) > Program Files (x86)/Steam/steamapps/common/Hand of Fate 2 SDK, now extract the 'handoffate2\_modding.zip' to a location of your choice and remember where it is!

**Note:** You should extract the 'handoffate2\_modding.zip' to an easy to find location as you'll need to find it in the future. We recommend your 'Documents' folder as a good location.

## What you need to do in Unity:

1. Launch Unity
2. This will take you to the 'Projects' window.
  - *Note:* If this is your first time launching Unity, this window will open with the Projects Learn screen.
3. Click 'Open' which is located in the top right of the window, and navigate to where you saved the HoF2\_Modding project as noted in [What you need to do on your computer](#) and select it.
4. Unity will now launch the HoF2\_Modding project.
5. When unity has loaded the project, the 'Mod Editor' window will appear.
6. The first thing that you'll have to do is point the **Install Path** at your 'Hand of Fate 2' game files.
  - *Note:* The game files are usually kept here - C:\Program Files (x86)\Steam\steamapps\common\Hand of Fate 2

Once you've done this you can click 'Edit' on the window which will take you to the card editor and presto, you can now mod the game!

What you need to do if you're on a Mac OSX or Linux system:

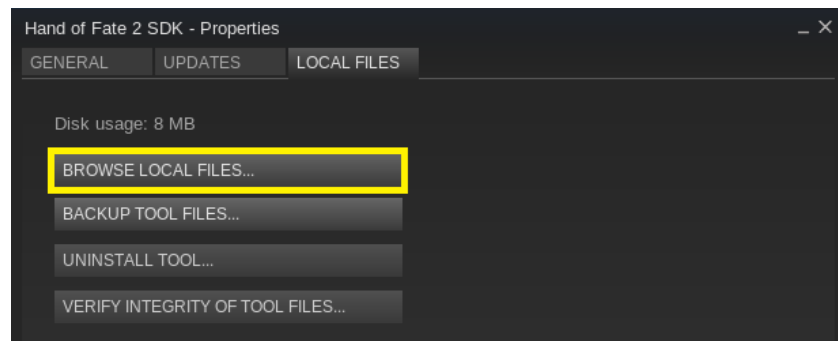
Hand of Fate 2 SDK path:

**Linux:** ~/.steam/steam/steamapps/common/Hand of Fate 2 SDK/

**Mac:** ~/Library/Application Support/Steam/SteamApps/common/Hand of Fate 2 SDK

**Note:** The best method for reaching these locations for Steam users is as follows:

1. Right click on the Hand of Fate 2 SDK from within **Steam Library > Tools**
2. Select **Properties**
3. Select the **Local Files** tab
4. Click the **Browse Local Files** button

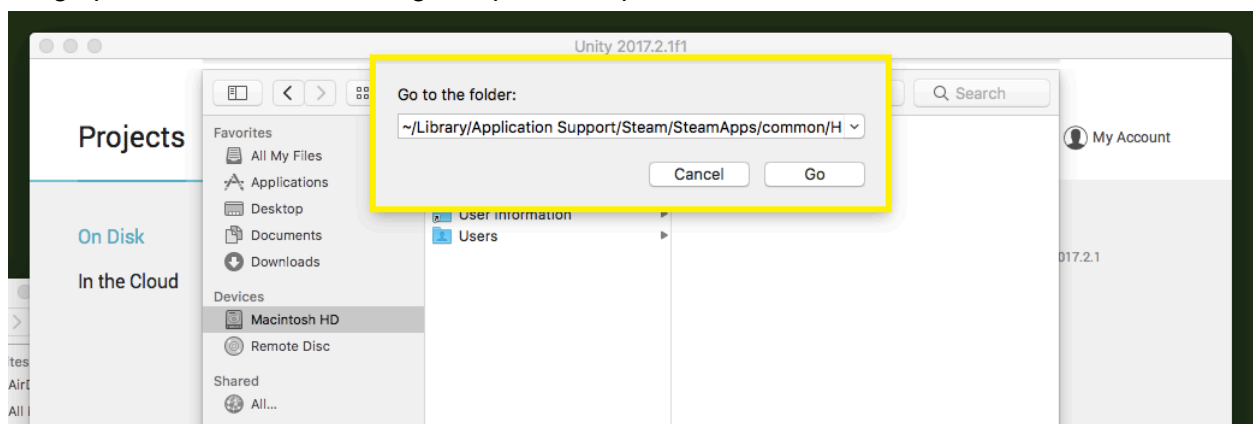


Location of your Hand of Fate 2 game files when setting the install path:

**Linux:** ~/.steam/steam/steamapps/common/Hand of Fate 2/

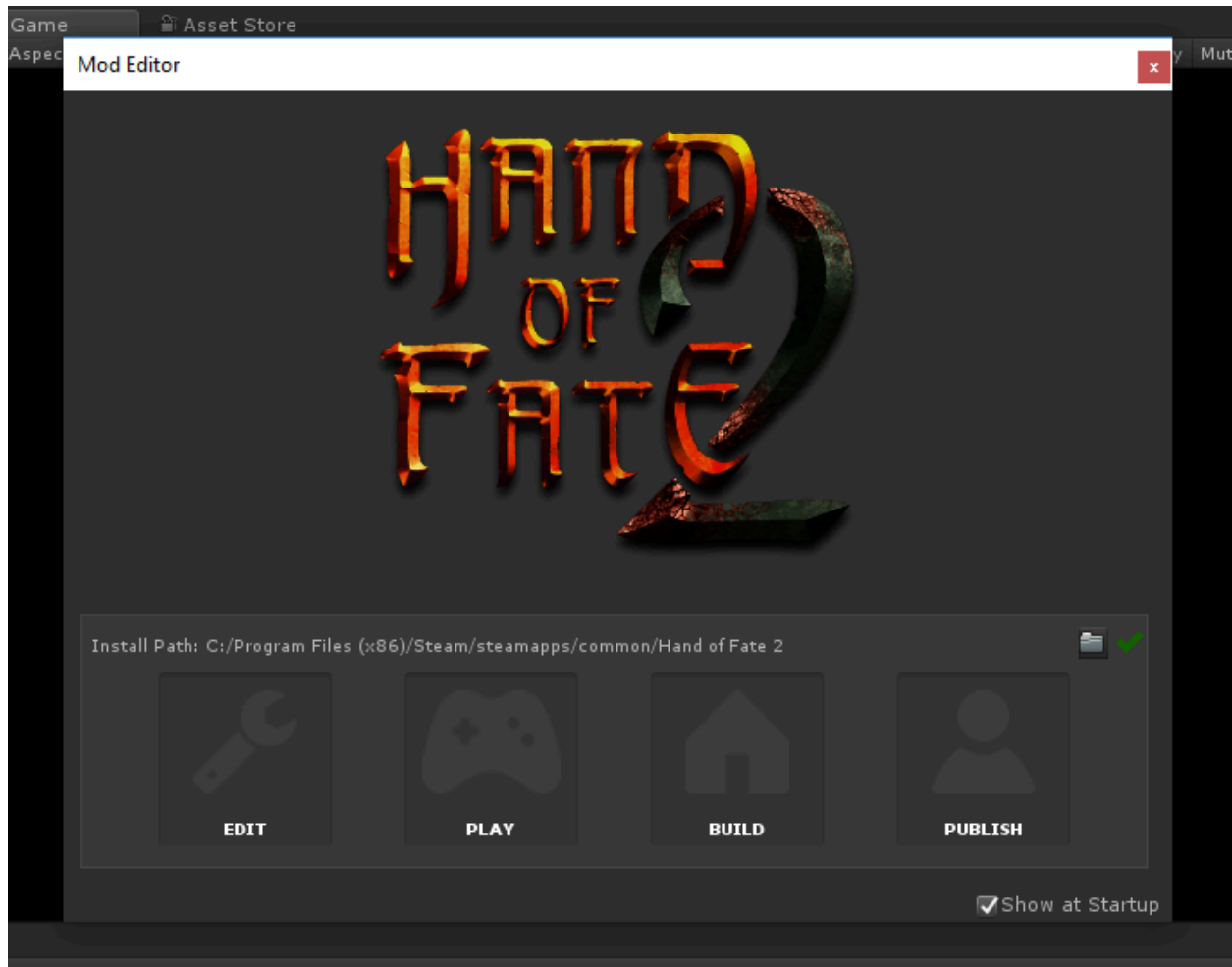
**Mac:** ~/Library/Application Support/Steam/SteamApps/common/Hand of Fate 2

**Note:** When selecting the game install path, Mac users will need to press Command+Shift+G to bring up the "Go to folder" dialog and paste the path above.



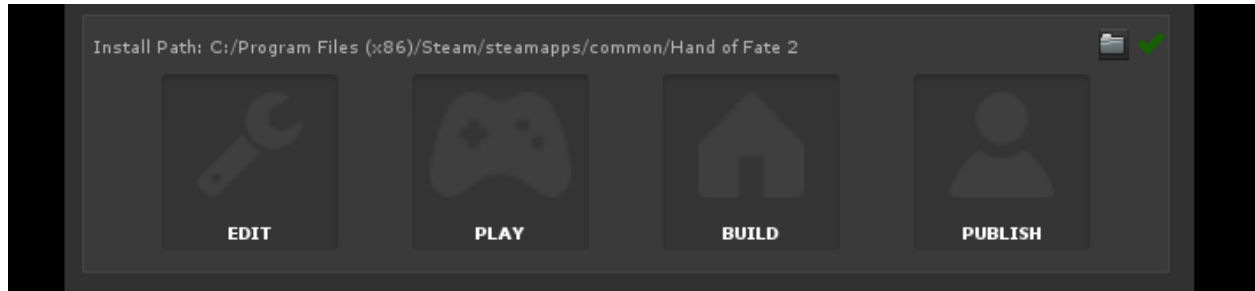
So you've done all of the steps required for installing and accessing the mod tools, congrats! The first thing you will be presented with when launching Unity is the **Mod Editor** window. This window gives you access to all of the different modding features available within these tools.

**Note:** The editor and scene view may not be black on your screen - this depends on your Unity license type. This does not matter as the tools can be used with any Unity license type.



## Explaining the Mod Editor window:

You'll notice along the bottom of the Mod Editor window that there are a variety of buttons that you can click on. Let's explore what they do.



### What does the “Edit” button do:

Clicking **Edit** is how you access the card editor. This is where you'll be creating your modded content for HoF2.

### What does the “Build” button do:

Clicking **Build** will build your modded content, allowing you to publish it to the steam workshop via the **Publish** button.

### What does the “Play” button do:

Clicking **Play** will allow you to play your modded content from within Unity. This is especially useful for testing your modded content prior to making a build of it and publishing it.

### What does the “Publish” button do:

Clicking **Publish** will publish your modded content privately (by default), or you can change this to publicly or to friends only. This is how you publish your mods to the Steam Workshop for HoF2.



## Updating your mod tools:

Occasionally when launching Unity and arriving at the **Mod Editor** window you'll notice an exclamation mark symbol in place of the tick pictured below. This will be located to the right of your **Install Path**, found on the **Mod Editor** window:



This is telling you that you need to update your mod tools. In order to update your mod tools, you'll need to follow these steps:

1. Launch Steam and update the Hand of Fate 2 SDK. Doing so will give you the latest version of the modding tools project archive (zip).
2. Open your HoF2\_Modding project folder (this will be located, wherever you've extracted it)
3. **Ensure that you make a backup of this project before following the remaining steps!**
4. Open the 'Assets' folder within.
5. Delete the 'Editor Default Resources' and 'Modding' folders located within the 'Assets' folder.
6. Extract the project archive ('handoffate2\_modding.zip') over your existing project folder.
7. You've just updated the modding tools and should be right to continue modding!

## Basic shortcuts:

Pressing **[Ctrl + Space]** while in an encounter behaviour tree will bring you to the start node of the tree if you get lost.

Pressing **[F8]** while running the game through the Unity Editor will pause the run and display where you are in a given encounter behaviour tree (if you're in an encounter currently). This is extremely useful for debugging encounters.

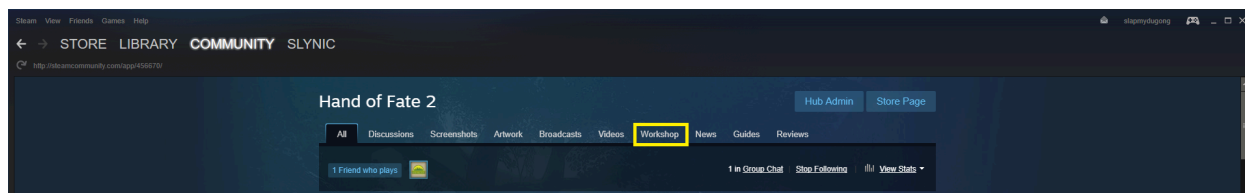
Pressing **[F5]** while in the unity editor will display the mod editor window. If you've closed your card editor window and have also closed your mod editor window, you can open the mod editor window by pressing the F5 key on your keyboard.

Pressing **[F12]** while in the Unity Editor will take you directly to the card editor window.

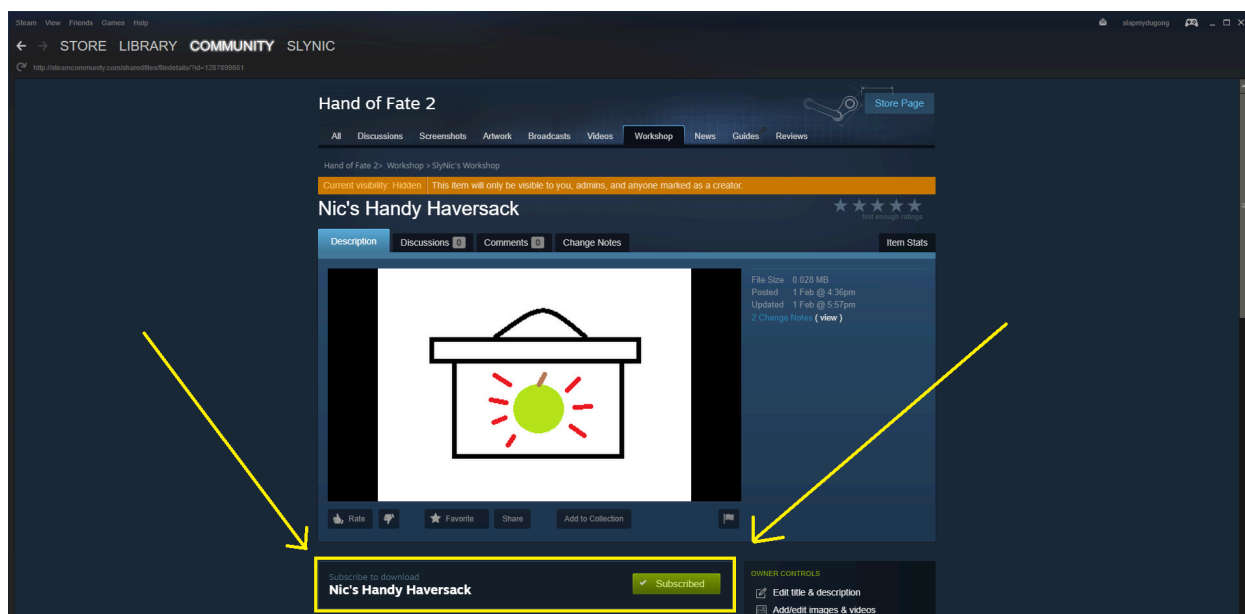
## How do I access modded content in my game?

So you've created a mod, you've uploaded it to the Steam Workshop, you've subscribed to your mod, and now you want to access your modded content in game. Here we'll look at what you need to do to enable mods on your HoF2 save file:

Launch Steam and navigate to the Steam Workshop for HoF2.



Subscribe to the mod(s) that you want from the Steam Workshop page.



**Note:** After you've clicked subscribe on the mod you want, they should have downloaded automatically. It's known that on occasion the Steam client will queue these downloads and at other times they're not displayed. We recommend that after you've subscribed to a mod, follow these steps below:

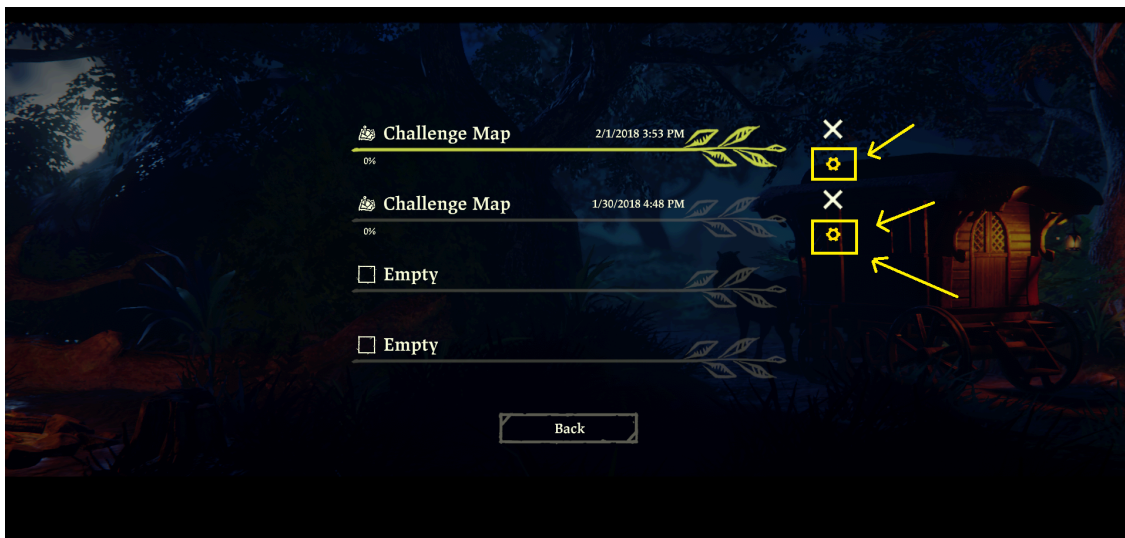
- Right click on 'Hand of Fate 2' from within your Steam Library.
- Select 'Properties'.
- Select the tab 'Local Files'.
- Click the button 'Verify Integrity of Game Files'. Doing this will force any available downloads to start.

After you've subscribed to your mod > Launch HoF2 game > Once the game has loaded you to the main menu, select **Load Game / Mods**.

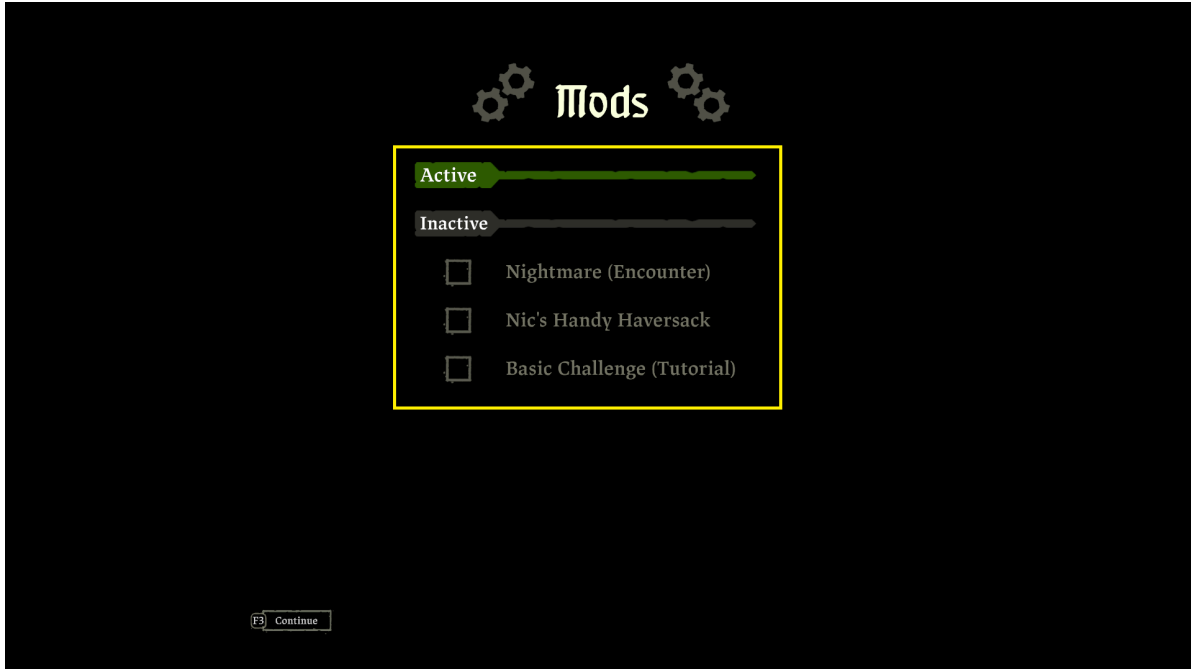


After you've selected **Load Game / Mods** you'll be presented with a list of your save files. Next to each of these save files is a cog - if you select it you can enable mods on that given save file. Alternatively you can click new game where you'll be prompted by the same message.

**Note:** If you want to create a new save file for your modded content (ie. you value your achievement progress on your other save etc). We recommend that you clone your existing save that way you still have all of your progress in that modded save file.



After you've enabled mod support on your save file, select that save file and load it. Once you do this you'll be brought to a new screen that will ask you to select mods to activate on this save file. Here is where you'll select the mods that you want to activate. After you make your selection, click play and you'll have the selected mods available to you in your game.



## Section 2: “*The Basic Encounter*” (Tutorial)

What is explored in this section of the documentation:

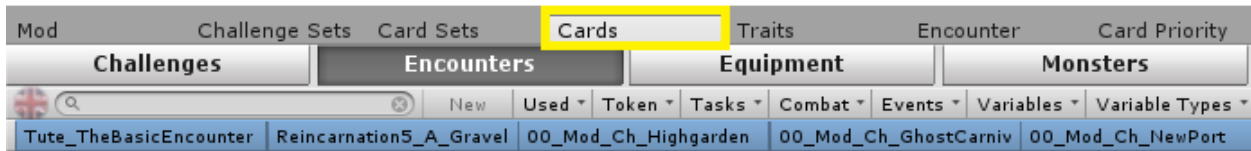
- [Creating the new encounter card:](#)
- [Viewing the encounter card data:](#)
- [Changing the name of the encounter card:](#)
- [Assigning an art texture to the encounter card:](#)
- [Assigning Traits to the Encounter:](#)
- [Navigating the Encounter Behaviour Tree:](#)
- [What are the different types of nodes we can create:](#)
- [Creating the first node of the encounter:](#)
- [Creating a choice for the player:](#)
- [Adding the chance cards mini game:](#)
- [Creating the chance cards:](#)
- [Creating variables in the Local BB:](#)
- [Creating the pain and gain results for the mini game:](#)
- [Creating the text for the results of the mini game:](#)
- [Activating the pain and gain cards:](#)
- [Ending the encounter:](#)
- [Afterthoughts and revision:](#)

What is the encounter we’re creating:

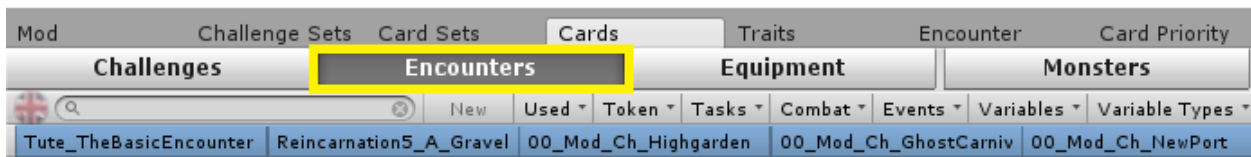
In this tutorial you’ll be creating the basic encounter. The basic encounter will feature as the transition card for the basic challenge you’ll create later in this document. The transition card is the point where we move from one dungeon level to the next. This encounter will feature a chance card mini game, a choice for the player (the option to accept and/ or to decline the offer made), a pain and gain result based off of the chance cards result, and will feature some of the core functionality that can be found in many existing encounters in HoF2.

## Creating the new encounter card:

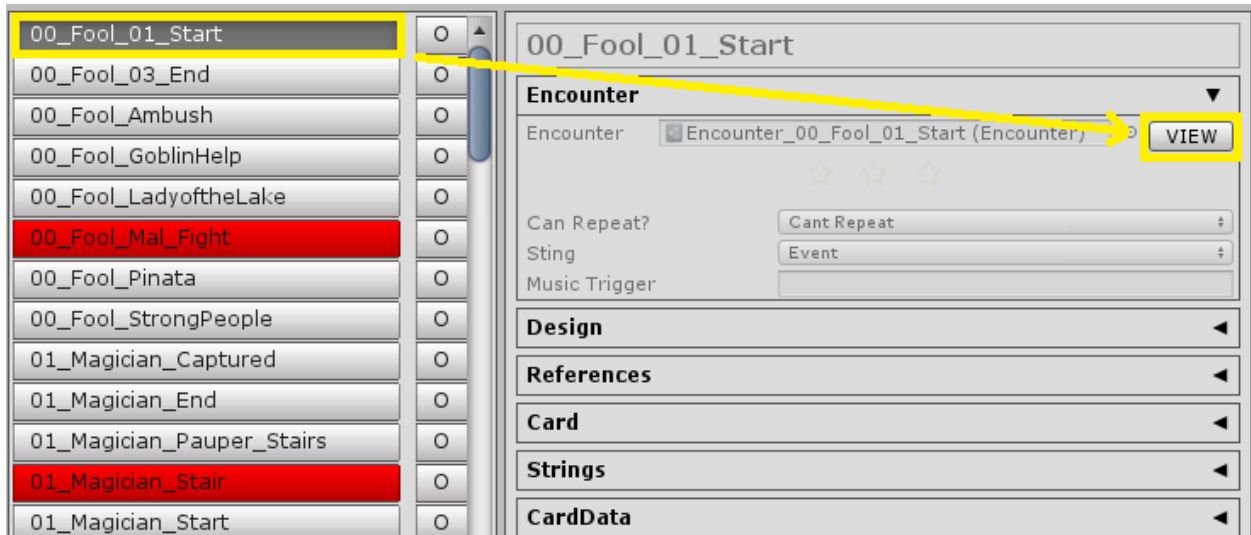
After opening the card editor, In order to start creating the “*Tute\_TheBasicEncounter*” encounter, you’ll need to create the encounter card. Navigating across the top toolbar of the card editor, you’ll spot a tab named **Cards**. Click the **Cards** tab, doing so will display a new toolbar directly underneath this one.



Navigating across the new toolbar underneath, you’ll spot a tab named **Encounters**. Click the **Encounters** tab, doing so will display a list on the left of the editor. This list contains all of the created/existing encounter cards within the card editor.

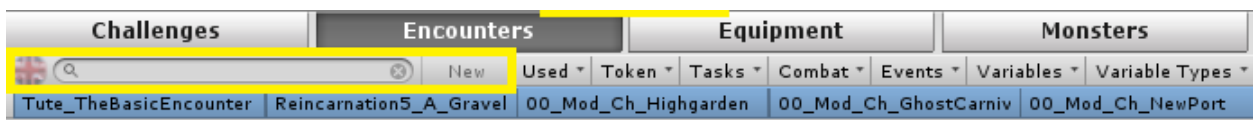


All of the encounter cards within this list are viewable. In order to view an encounter cards behaviour tree, select the card that you wish to view, and click the **View** button highlighted below.



The ability to view existing encounter cards allows modders to see the content of existing encounters in HoF2. Clicking **View** will take you to the encounters behaviour tree, here you'll be able to view the visual scripting backend of how encounters are pieced together, and how they operate. Clicking **View** allows you to navigate around the behaviour trees without making any edits to the original versions. Encounter behaviour trees should be assessed, as they're a great way of seeing how existing content in HoF2 is pieced together and set to operate.

At the top of the list of existing encounter cards, you'll spot a search bar with a button next to it labelled **New**. Click inside of the search bar and type the name "*Tute\_ABasicEncounter*" and click **New**, doing so will create the new encounter card with that name. Alternatively you can name this to whatever you'd like, but don't forget what you called it!



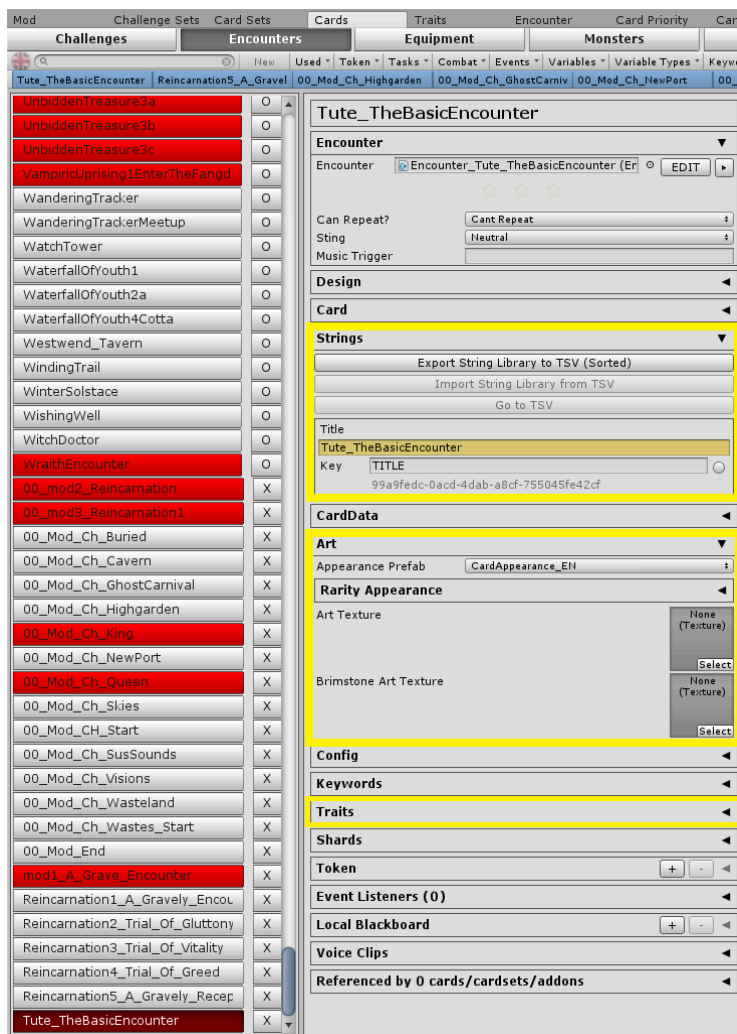


# Viewing the encounter card data:

Now that you've created the new encounter card "Tute\_ABasicEncounter", or whatever you've named it on your end, we can start to look at entering some data for our new encounter card.

Firstly, let's click on the new encounter we've just made. You can do this by navigating the list of encounter cards to the left of the editor and selecting it that way, or you can search for the name of your card directly via the search bar, this is located at the top of the list.

**Note:** Newly created cards will always appear at the bottom of this list that's located to the left of the card editor. Once you close down the card editor and reopen it, the cards will be sorted into alphabetical order.

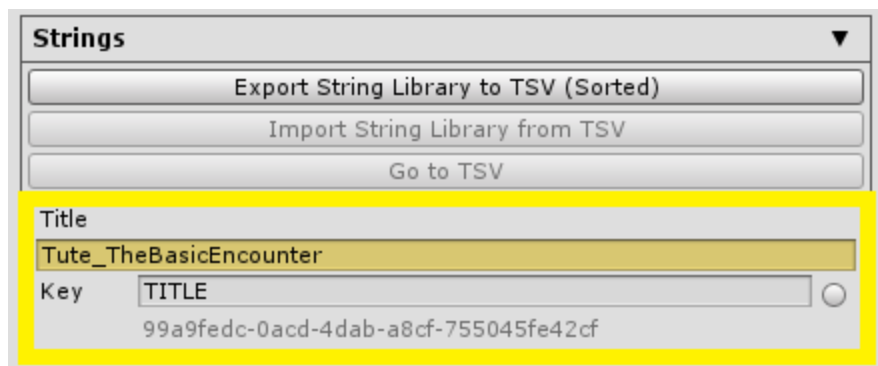


There is a lot of different data that builds an encounter card, for the basic encounter we only need to worry about the three sections highlighted by the yellow boxes in the above screen capture.

## Changing the name of the encounter card:

**Strings** are text fields that allow us to change lines of text on a card. **Strings** in the case of encounter cards, include things such as the **Title** of the encounter card which is viewable in game, and through the card editor.

You'll notice that once you've clicked on and have opened the **Strings** label of the encounter card, that the **Title** string has been pre filled for us already. This has been pre filled by the name we gave the encounter card when we created it, but we don't necessarily want the encounter (when it's played in HoF2) to read "*Tute\_ABasicEncounter*" on the card, so let's change the title string to "*The Basic Encounter*", or to whatever you'd like to name your encounter.



**Note:** You can enter strings by directly entering your text into the field highlighted above. Alternatively you may have to click the circle next to the **Key**, this will open the **StringPicker** and you can assign your strings in there.

Once you've renamed your encounter card, let's jump back to the list of encounter cards for a moment. You'll notice that even though we've changed the title string of the encounter card to "*The Basic Encounter*" (on the encounter card data itself) that the encounter card in the list is still named "*Tute\_ABasicEncounter*".

If you want to view all of the encounter cards in this list via their **Title** string, you'll need to do the following. You may have been wondering what the small union jack symbol (flag) next to the search bar means/does. If you click this, the search switches between the **Title** and the **ID**. So if we click the flag, then we'll be able to see our card is named as the title we gave it, "*The Basic Encounter*".

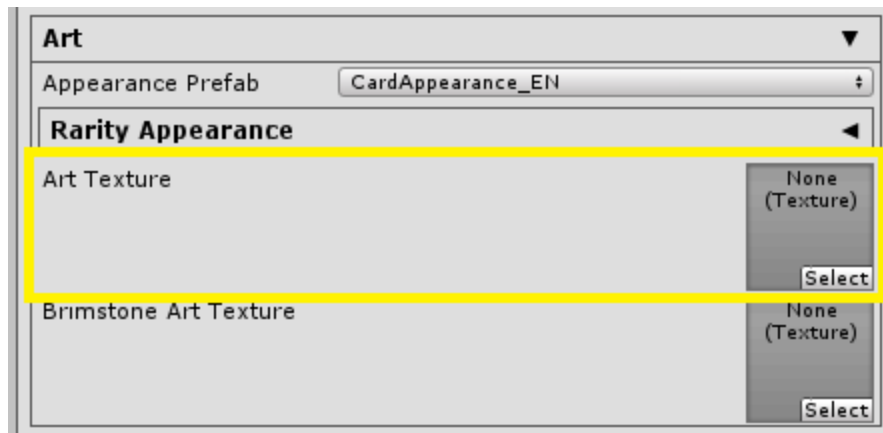


After renaming your encounter card to your new **Title** string, let's look at adding some art to the card.

## Assigning an art texture to the encounter card:

Next let's open the **Art** label and with this we'll have some new data to work with. **Art** is the card art that's going to be shown on our encounter card, in game. There are three types of card art that can be assigned to any given card. One of them is what's called an **Art Texture**, this is the basic version of the card art selected for the encounter card, however this is also where you'll assign any **Platinum** textures for your cards (this is important to know, as it's not incredibly clear). The second is the **Brimstone Art Texture**, Brimstone level cards are of greater rarity, and only encounters with a Brimstone version should have the associated texture assigned.

You'll notice two selection boxes under this label, one is named **Art Texture** and the other **Brimstone Art Texture**, for this encounter we'll only be using **Art Texture**.



After clicking the selection box next to **Art Texture**, a new window will open with a lot of art options for our encounter card. Let's search for the title "*c\_stair\_tavern*" and assign this as our **Art Texture** for the encounter. Alternatively you can select any texture that you'd like to use!

After changing both the title string and the art texture for our new encounter card, we can now look at adding some traits to our encounter card.

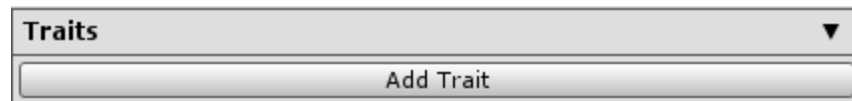
## Assigning Traits to the Encounter:

Think of encounter traits as tags for the contents of your encounter. You don't need a trait for all the bits of your encounters contents, but you should have one for the main bits so that players can have a clue as to how they should be deckbuilding with this encounter. The primary purpose for traits to be included or considered for your encounter is for reasons of deck building.

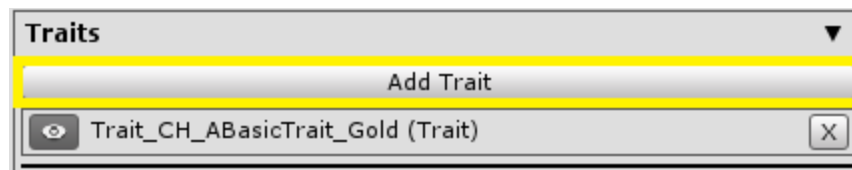
There are three traits that we'll be assigning to our encounter. These traits are:

1. *"Trait\_EN\_Res\_Gold\_gain"*
2. *"Trait\_EN\_Game\_Chance\_neutral"*
3. *"Trait\_EN\_Res\_Health\_loss"*

Let's add some traits! Start by selecting the **Traits** label found on the encounter card:

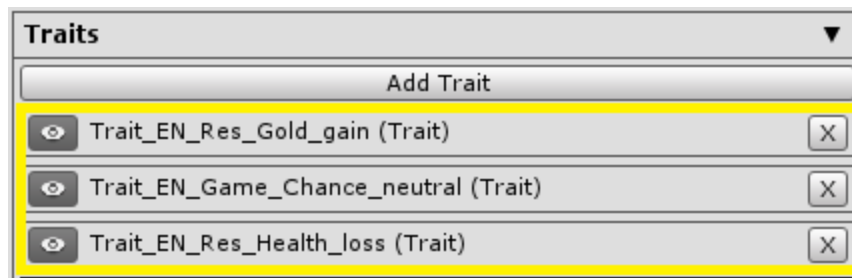


Next, click the button within the expandable label, this button is named **Add Trait**:



After you click the **Add Trait** button you will have the option to search for specific traits that already exist in HoF2. Search for the traits that we listed at the top of the page, and add these to your encounter card.

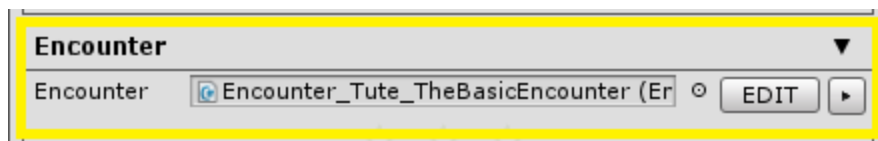
Once you've done this, your list of traits should look like the screen capture below!



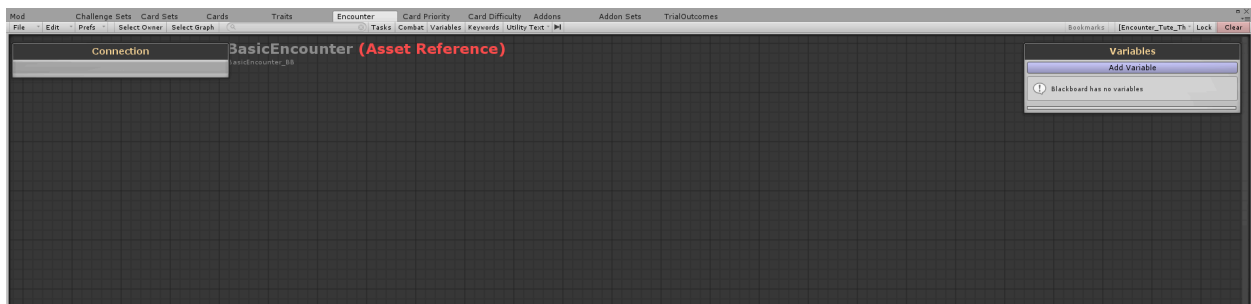
## Navigating the Encounter Behaviour Tree:

When you're viewing your encounter card, you'll notice at the top of the card a label named **Encounter**. Within this label you'll find a button named **Edit** and directly to the right of that will be a small play button.

**Note:** This small play button is one way that you can directly debug and play your encounters in game, but for now let's leave this alone as we have nothing to play.



Let's click the **Edit** button, this will take us to the encounter behaviour tree for our encounter. Below is an example of a new behavior tree, yours will look like this.



Behaviour trees are a grouping of conditional actions that can run at the same time or independently. They react to changes (conditions) immediately and fire off actions as a change is detected.

In the top right of the behaviour tree you'll find the **Local Blackboard (BB)**, this is where we store all of the different variables that will be used throughout this encounter behaviour tree.

In the top left of the behaviour tree you'll spot a window named **Connection**. Whenever you create a new node within the behaviour tree, that node's data will be shown in this window (when said node is selected). When a node is selected and the data is shown here, the name of the window changes to that of the node selected.

**Note:** Navigating the encounter behaviour tree - This is one of the fun/frustrating idiosyncrasies of the Unity Engine UI, Holding middle mouse and drag should help you navigate the encounter tree if you're ever stuck.

## What are the different types of nodes we can create:

If you right click anywhere on the behavior tree you'll have multiple options to select from. These include a variety of different things, including the ability to create different nodes within the encounter behavior tree.

We can create what's called a **Page**. A Page node holds text for our encounters and gives us the ability to create player choice.

We can create what's called an **Action**. An Action node is used to set up encounter logic, from drawing and activating cards, to starting Combat.

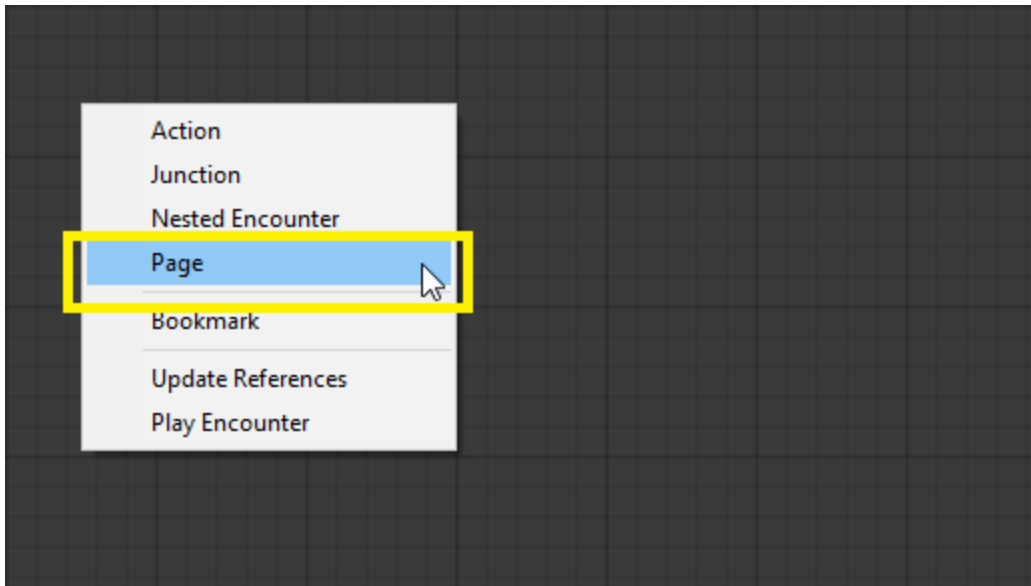
We can create what's called a **Junction**. A Junction node is a useful tool for breaking up complex connections within an encounter behavior tree. Junctions can be used to split a single connection into different directions. The field within a Junction is where you set the logic for a **loop**. Loops are created when you need to repeat a lot of nodes that are within an encounter behavior tree (saving you from creating them again and again). If you'd like to read more on loops, click [here](#).

We can create what's called a **Nested Encounter**. Nested nodes allow you to run other encounter logic, held within another encounter behavior tree. These are really useful for things that get reused a lot.

We also have the option **Play Encounter** available to us when we right-click in the encounter behavior tree. This does the same thing as clicking that small play button that we highlighted earlier. Selecting this option allows us to debug and play our encounters in the game.

## Creating the first node of the encounter:

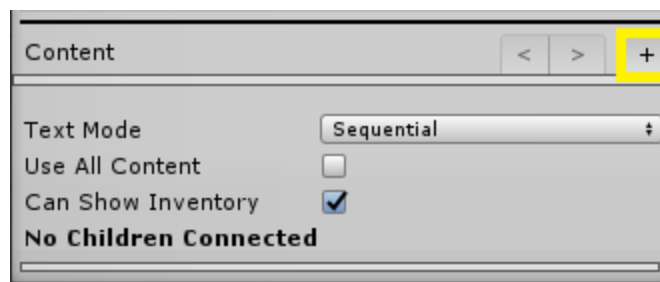
Let's create a **Page** node in our encounter behavior tree. This is visualized in the below screen capture.



After you've created your page, let's change the name of the page, that way we'll always know what this page relates to whenever we look back at this encounter behavior tree.

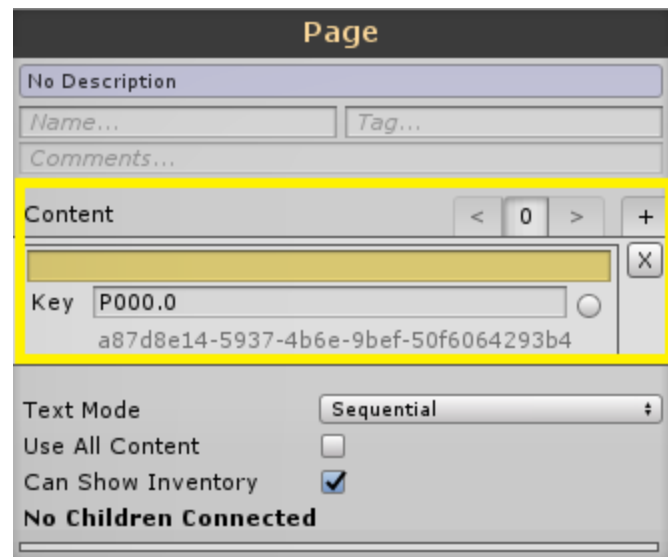
To do this, select the page you've just created. This page will now be visible in the window named **Connection** (in the top left of the encounter behavior tree). Notice how this window is currently named "*Page*" as that is the connection.

Let's enter the name "*Encounter Introduction*" as the name of our page, this way we'll always know that this page is the introduction for our encounter. Let's also click the small addition symbol next to **Content** while we are here.





After clicking the addition symbol you will have created a content field for us to input text for the page. This is where we are going to be entering our encounter introduction text. This section is demonstrated in the below screen capture, highlighted by the yellow box.



Now that we've created the **Content** for our encounters text, let's actually enter some text. Below you can find the text to enter, we've provided this for you.

### **Encounter Introduction (Start)**

"A life of risk is what you've always favored, and tonight finds you in no better company..."

After you've entered the encounter introduction text, let's create a new page and connect it under our encounter introduction page.

**Note:** You can connect nodes together by dragging a connection from the bottom of one node to the body of the other.

## Creating a choice for the player:

Let's look at creating a choice for our players, this is going to be simple with two possible outcomes as a result. First we'll be using that new page that we connected under our encounter introduction page. Next, go ahead and rename this new page, "*Player Choice*".

Let's create 2 new pages and connect these under our "*Player Choice*" page. Let's name the first page (moving left to right) "*Progress*", and the second page "*Leave*".

Let's enter some text into our pages. Below you can find the text to enter, we've provided this for you.

### Player Choice (Page)

"Your opponent gestures towards you, egging you on to challenge him in another round of knife game..."

### Progress (Page)

"You stare your opponent dead in the eye, and proceed to nod in acceptance of the challenge..."

### Leave (Page)

"You contemplate the offer for all of a second, before you leave the table and proceed to the exit of the tavern..."

Next, you will need to add the **Options** text for the "*Player Choice*" page.

This is to give context to players about the choice they're making. Let's enter "*Accept*" as our text for option A, and "*Decline*" for option B. This data is shown below.

Options < A) B) >

Text

Accept.

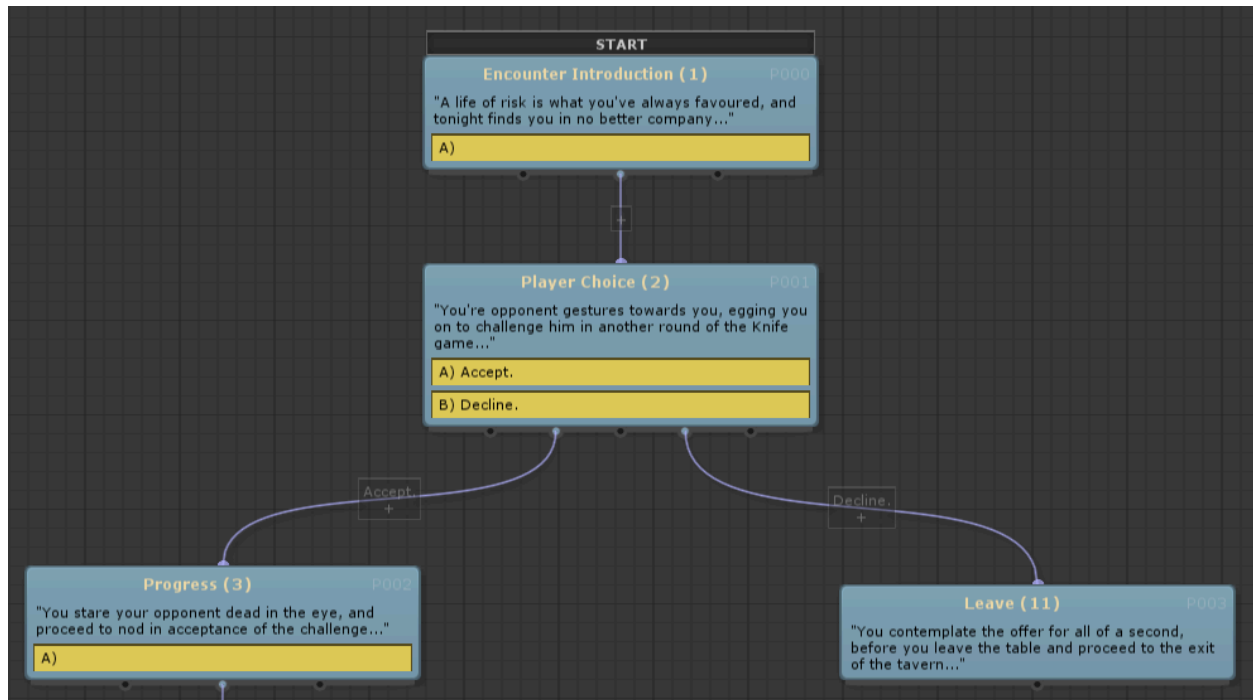
Key P001.A  
c4fa46aa-821a-4e74-be39-8cbd6bf5e5dc

Utility Text

Create

Hide on failure

Your encounter behavior tree should now look similar to the example shown below.



## Adding the chance cards mini game:

Next, let's look at creating the chance cards mini game that will act as the knife game in our encounters narrative.

Before we start let's look at the different mini games that are available for use in HoF2. There are four types of mini games in HoF2. These include the **Chance Cards**, the **Dice Gambit**, the **Pendulum**, and the **Wheel Spin**. Let's explain these in a little more detail.

**Note:** If you've been redirected here, from the intermediate encounter design section of this documentation, click [here](#) to return to that section.

### Chance Cards Mini Game:

- Chance Cards are a mini game that consists of 4 - 6 cards that are shuffled face down, followed by the player making a selection out of the available cards.
- Chance Cards are limited to having 6 cards per shuffle, and you may only assign 4 cards when creating the mini game.
- We limit the chance cards to 6 so that we can account for the blessing "*Eternal Hope*" and the special ability of the "*Ariadne*" companion. You will always have to account for a Huge Success result when creating chance cards, for the "*Eternal Hope*" blessing.
- The cards are displayed face up in front of the player prior to the shuffle.
- Each card has a different result type and these can include Success, Huge Success, Failure, and Huge Failure.
- You can also put non-chance cards into a chance cards mini game, ie. food, health, gold etc.

### Dice Gambit Mini Game:

- Dice Gambit is a new mini game in HoF2. It's a dice roll where there is a value to reach/beat and that will determine if the mini game was a success or failure.
- Companions like "*Colbjorn*" can add dice to your roll, this is a unique ability of his.
- When you're setting the dice target for the gambit, 13 was a suitable average dice target to be set. Doing so allowed us to add or subtract from this target based on if we thought the gambit was meant to be hard or easy.

### Pendulum Mini Game:

- Pendulum is another new mini game in HoF2. It's a pendulum swing where you'll need to stop the swing and land the pendulum onto either a Success, Huge Success, Failure, or Huge Failure segment.

## Wheel Spin Mini Game:

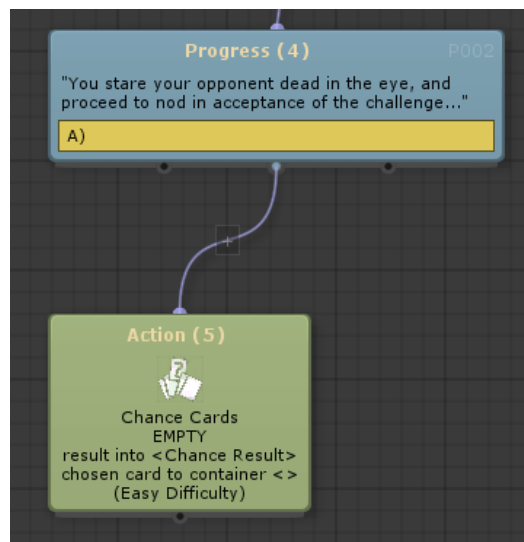
- Wheel Spin is another new mini game in HoF2. It's a wheel spin where you'll have a variety of different cards available in any given spin. These cards include: Food, Health, Pain, Gain, Gold, Weapons, Armour, Enemies, etc. The player stops the spin and therefore lands on a card, that card is the result they get.

## Creating the chance cards:

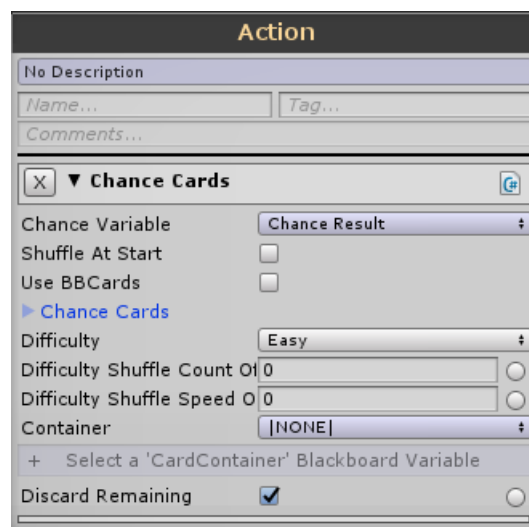
Start by creating a new node, this time creating an **Action**.

Connect this node under the “*Progress*” page in your encounter behavior tree. Now select the action so that you can view it in the connection window.

Let’s add an **Action Task**. The action task that we’re looking to add is named **Chance Cards**.

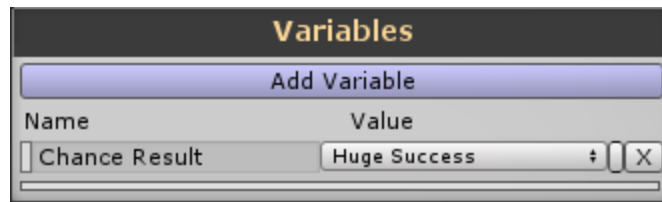
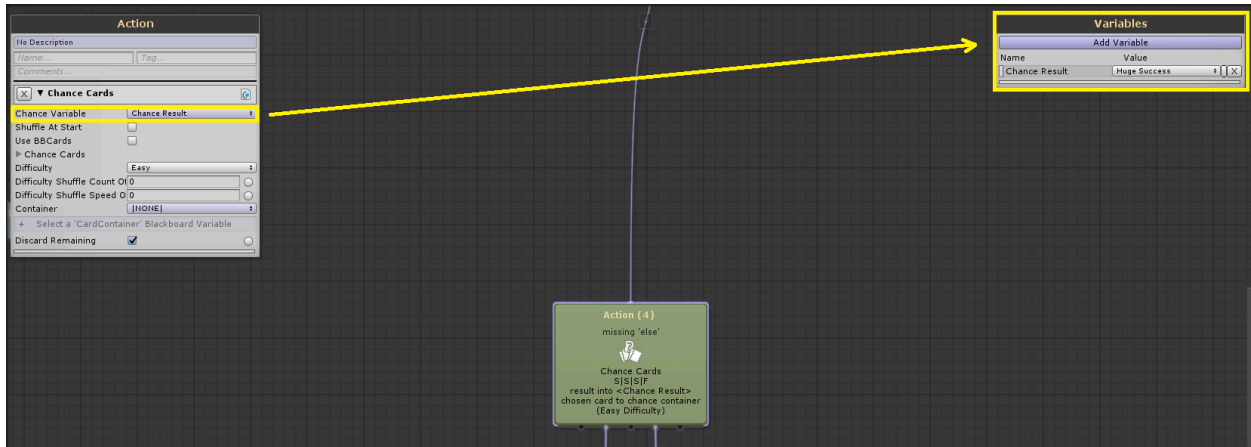


The action task **Chance Cards** has some data that we’ll need to add, let’s look at setting that up next. The data within the action task **Chance Cards** is demonstrated in the below example.



The first thing you'll spot in the **Chance Cards** data, is a variable labeled **Chance Variable**, you'll also notice that this has been set for you already, but how?

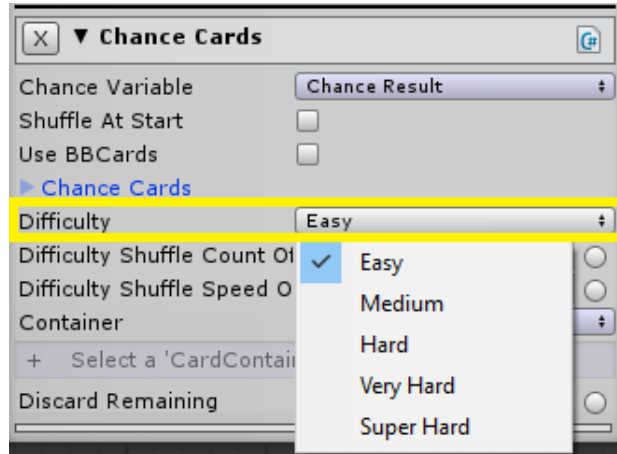
**Note:** When you create the action task **Chance Cards**, this creates a new variable within the Local BB of this encounter behavior tree, that variable is named **Chance Result**. Where you can find this in your encounter behavior tree, is demonstrated in the below examples.



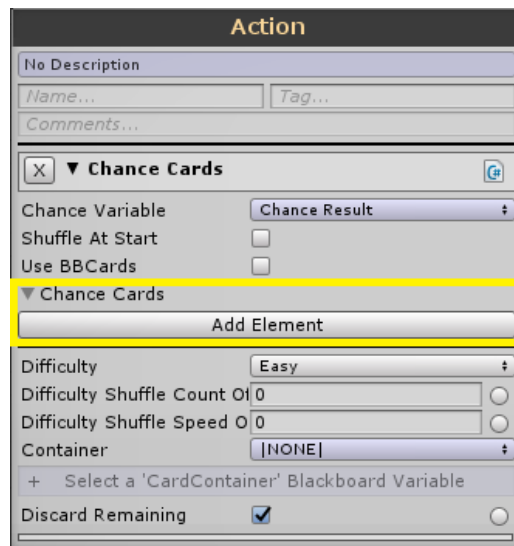
**Note:** The above example displaying the Local BB, shows the **Chance Result** set as Huge Success. If you don't change the **Chance Result** but check what value it is, it will always return Huge Success (or whatever it's set to). This is applicable across all variables made in the Local BB, that are checked against.

**Difficulty** is another piece of data found in the action task **Chance Cards**. This is a drop down list of varying difficulties for the chance card mini game. These difficulty ratings relate to the speed of which the 4 cards shuffle in front of the player, and the number of shuffles that are completed. This does not alter the amount of success and/ or failure cards that are within the shuffle. This data is demonstrated in the below example.

**Note:** Typically the difficulty setting **Easy** is never used for mini games in HoF2, but it can be a useful way to test your mini games before submission to the Steam Workshop.

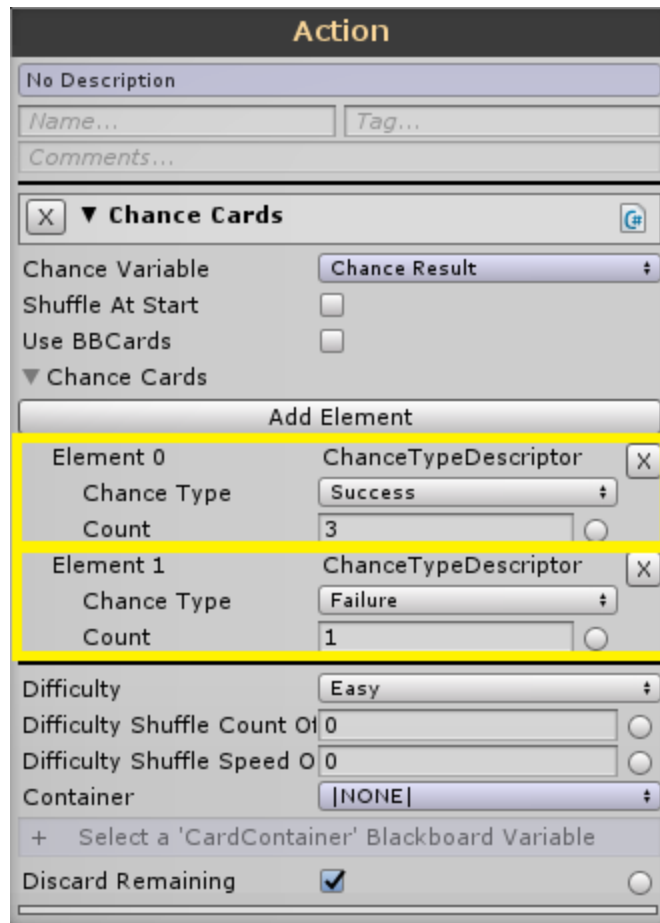


Within the action task, there will be an expandable section labelled **Chance Cards**. This is where we add the cards that are shuffled in the mini game. Once expanded, we'll be able to create an **Element**. Elements are where we store our Success, Huge Success, Failure, and Huge Failure cards, used in the shuffle. You can click **Add Element** to create a new element.





Let's create 2 elements. **Element 0** is going to store our Success cards for the shuffle, and **Element 1** will store the Failure cards for the shuffle. This is demonstrated in the below example.



Start by setting the **Chance Type** for **Element 0** to be **Success**. Then set the **Chance Type** for **Element 1** to be **Failure**.

Next, we need to set the **Count** for these. **Count** is the amount of that **Chance Type** that will be used in the chance card mini game. Think of each element as a card used in the mini game.

Let's set the **Count** of **Element 0** to be **3**, and the count of **Element 1** to be **1**. This means that there will be 3 of the Success chance type, and 1 of the Failure chance type, used in our chance cards mini game. This is all shown in the above example too.

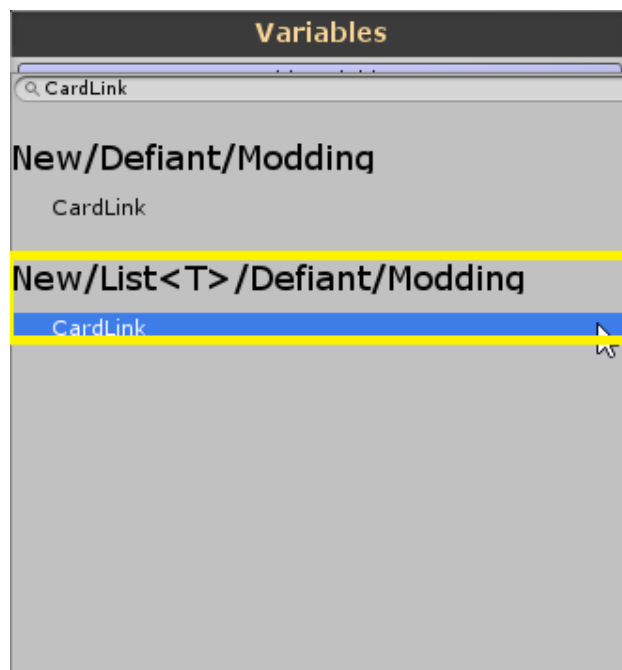
**Note:** The chance card mini game only supports 6 cards, and can only ever take 4 cards in the chance shuffle (initially). This is to account for the blessing "*Eternal Hope*", and the unique ability of the "*Ariadne*" companion, who adds one duplicate of a chosen chance card on the screen.

## Creating variables in the Local BB:

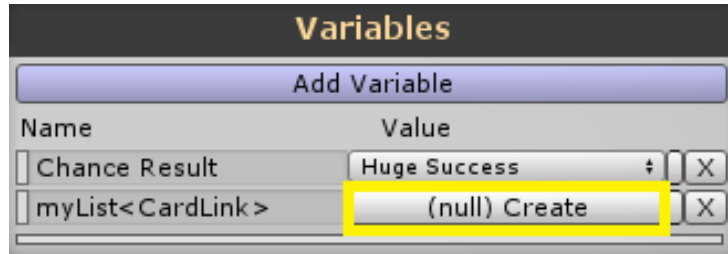
The Local Blackboard (BB) is where you'll be creating and storing your encounter trees variables. The variables stored in the Local BB are ones that are used frequently and throughout the entire encounter behaviour tree (between multiple nodes).

Let's set up a new variable in our Local BB. This variable will be used for the next part in our encounter behaviour tree.

Navigate to the Local BB and click **Add Variable**, now search for **CardLink**, this will bring back two different types of the **CardLink** variable. Make sure you create the **CardLink** variable under **New/List<T>/Defiant/Modding**. This is highlighted in the example below:



After you've created the **CardLink (List<T>)** variable, you'll see that the variable in the Local BB is returning null because the list has not been created. Click the button that reads **(null) Create**, doing so will create the list for you. It's important to rename your variables from their default string, the default for the **CardLink (List<T>)** variable will be **myList<CardLink>** so let's change this to something more identifiable like **sourceList**.



After adding the variable to the Local BB, we can start implementing the pain and gain cards that will be the result of a success/failure reveal in the chance card mini game.

## Creating the pain and gain results for the mini game:

In order to create pain and gain cards for this encounter, we need to think about what are appropriate results for the encounter narrative we're creating for our players.

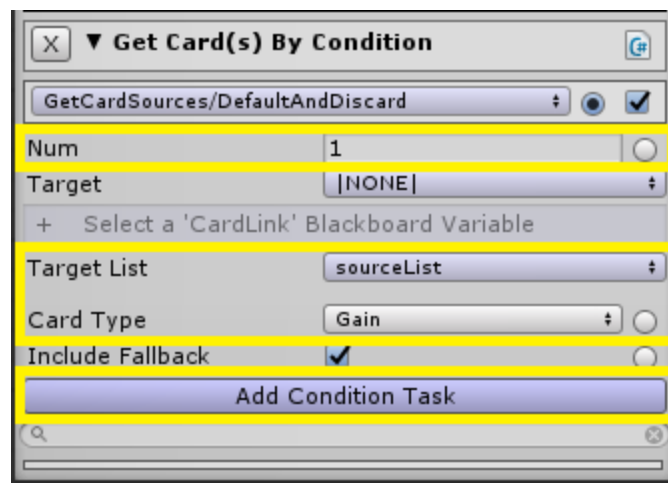
**Pain** and **Gain** cards reward players with particular things, they can award players gold through a gain gold card, or perhaps lose health through a pain health card.

Seeing as we're creating a narrative that follows a knife game with a competitive angle, it makes sense to have the gain card as the result of a successful reveal in the chance card mini game. This will be a gain of gold, giving the imitation of winning a bet. The failure reveal will award the pain card, this one will be a loss of health, tied to your character stabbing themselves in the hand.

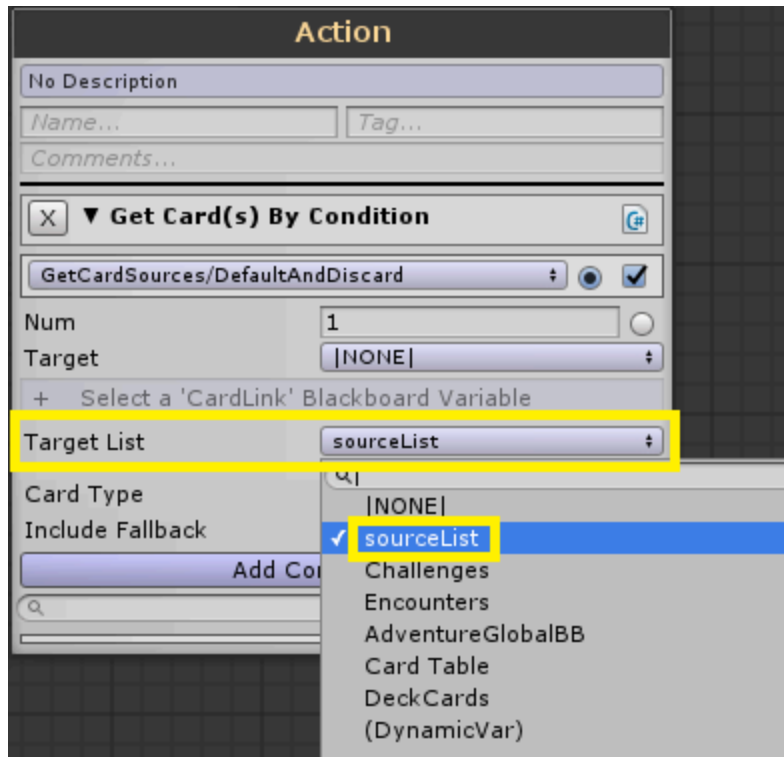
Let's look at how we can grab these types of cards, followed by how we award them according to the player's results in the chance cards mini game.

Let's start by creating a new action, and adding the action task **Get Card(s) By Condition**.

Once you're viewing the data of this action task, let's walk through the things we'll need to set up. The data highlighted below, are the ones we'll need to change.

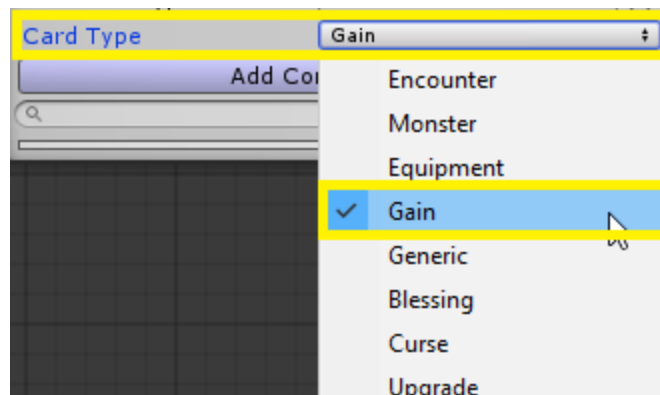


Let's have a look at the **Target List** data. Notice that **Target List** needs a variable to store the card we are retrieving. Remember when we created that **CardLink** variable in our Local BB earlier, we named it **sourceList**. This is where we'll be using that variable. Let's select **sourceList** and assign it as the **Target List**. This is demonstrated in the example below.

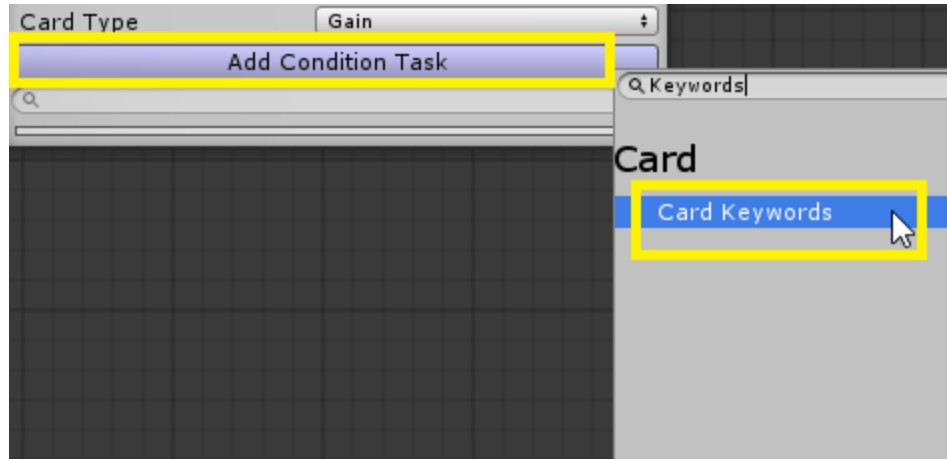


Now that we have the **Target List** assigned as our **sourceList** variable, let's move onto assigning our **Card Type**.

**Card Type** has a drop down list where we can select the type of card we want to retrieve. Let's assign **Card Type** as **Gain**. This is demonstrated in the example below.

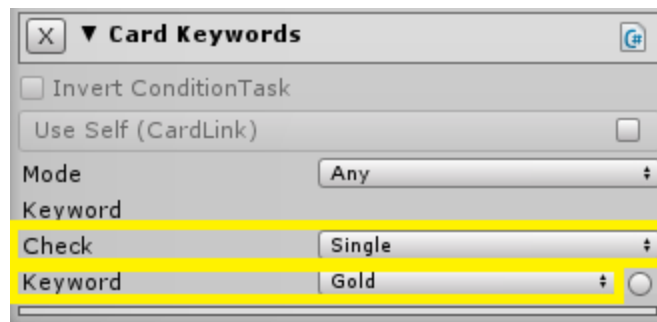


Now that our action task knows what type of card to retrieve, let's look at adding a condition task so that the action knows what type of gain card to retrieve as the reward. Click **Add Condition Task**, and let's search for the condition task **Card Keywords** and select it. This is demonstrated in the example below.



After adding the **Card Keywords** condition task, there are a few things that you'll need to do.

There are 2 bits of data that we need to set, these are labelled **Check** and **Keyword**. These are demonstrated in the example below.



The default data for **Check** will be set as **List**, but we only want one type of gain card to be searched for in this case, so let's set this to be **Single**.

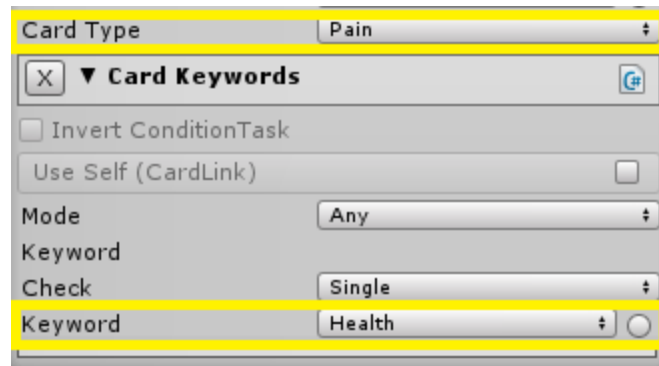
The default data for **Keyword** will be set as **Gold**, which is coincidentally what we want it to be. If you click on the expandable list next to **Keyword**, you'll see all of the different keywords we can search for on a card.

After the **Card Keywords** condition task has been created and set up, that's the gain card done!

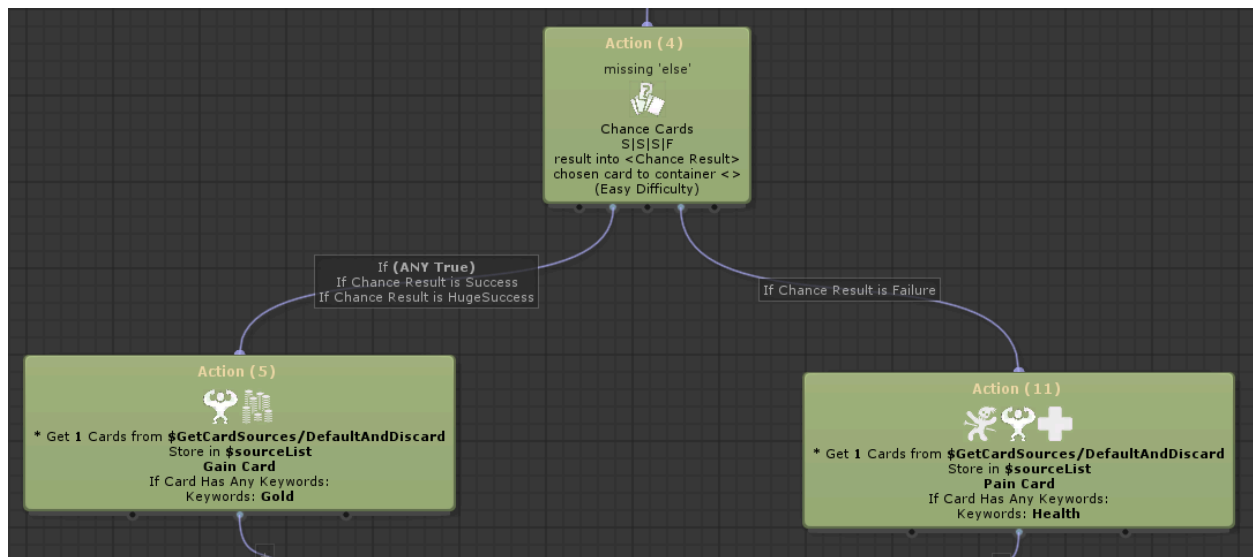
Next we'll be creating the pain card for a failure result on the chance cards mini game. Instead of repeating all of the steps we've just been through, let's use what we've just made again.

Duplicate the action for retrieving the gain card and let's change some data in it.

Let's change the **Card Type** to be **Pain**. Let's also change some data in the **Card Keywords** condition task. Change the **Keyword** to be **Health** instead of **Gold**. Our second action, for retrieving our pain card should look like the example below.



After making the appropriate changes to the second action, connect both of these nodes underneath the chance card's mini game action. This is demonstrated in the example below.



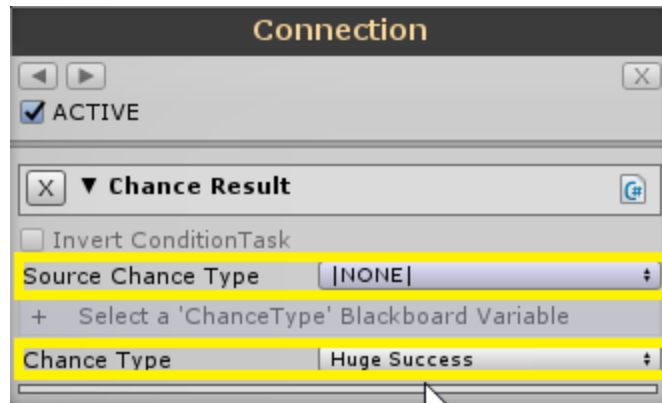
Now that you've connected these nodes under the chance cards mini game, we'll need to add our condition checks, that check for the result of the chance cards mini game and then respond appropriately.

In the above example, there are two connections coming from underneath the chance card mini game action (one will be red on your screen, and the other one will be blue).

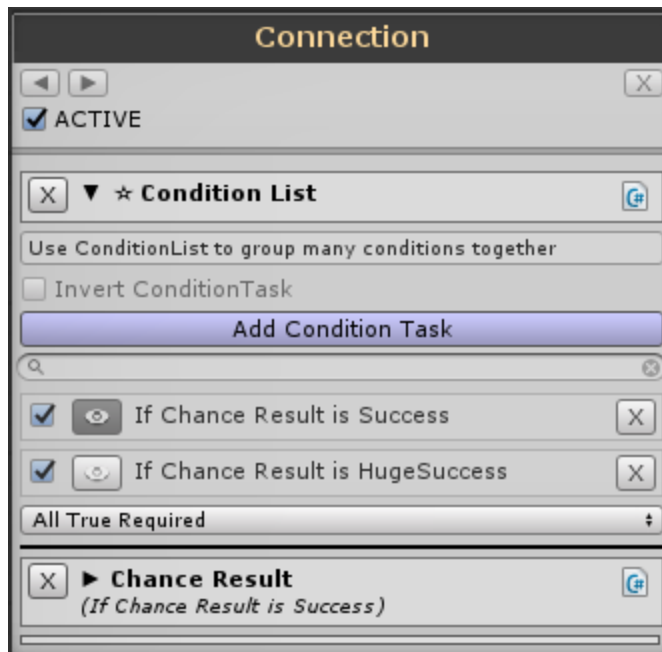
These are where we add our condition checks.

Click on the red line to get started and create a **Condition List**, a **Condition List** allows us to run multiple **Condition Tasks** instead of just 1.

Search for the condition task **Chance Result** and create it. This condition task needs you to assign a **Source Chance Type**, set this to be the **Chance Result** variable. Next you need to select the **Chance Type**, set this one to be **Success**. This process is demonstrated below.

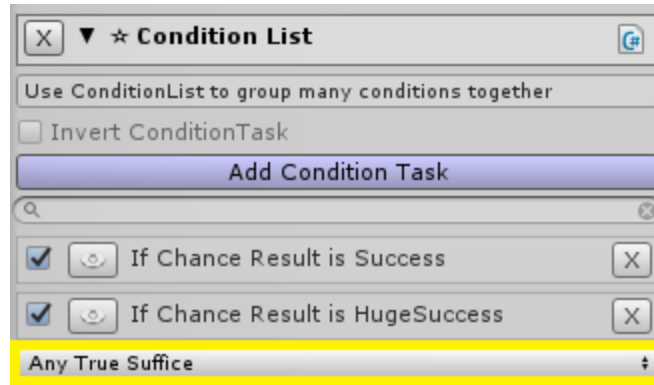


Let's create another **Chance Result** condition task. Let's assign the **Source Chance Type** to be the **Chance Result** variable. Now select the **Chance Type**, set this one to be **Huge Success** this time. Why have we done this? We have done this to account for the "*Eternal Hope*" blessing. Demonstrated below are these two condition tasks that are apart of our condition list.

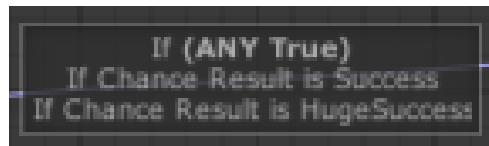




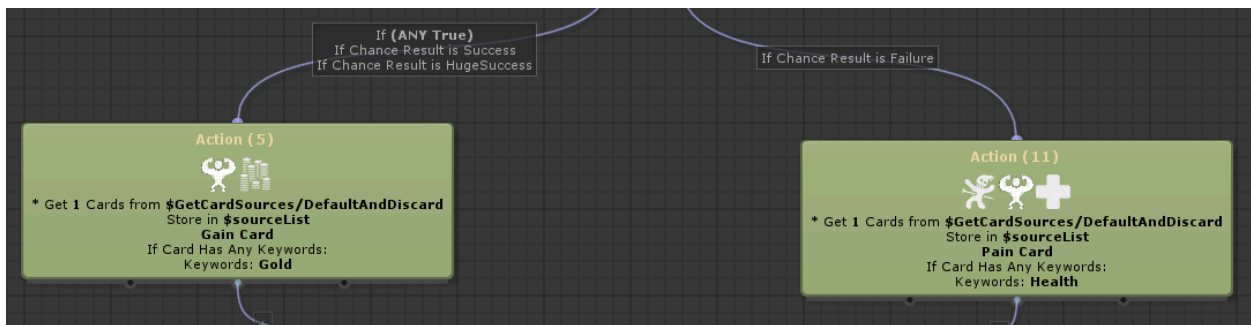
while you're looking at the **Condition List**, you will spot a piece of data which is set by default to **All True Required**. We want to set this to **Any True Suffice**. If we were to keep it assigned as the default option, the chance result would need to be a HS and S at the same time, instead of one or the other. This is demonstrated below.



The conditions we just made on that connection, should now look like the screen capture below.

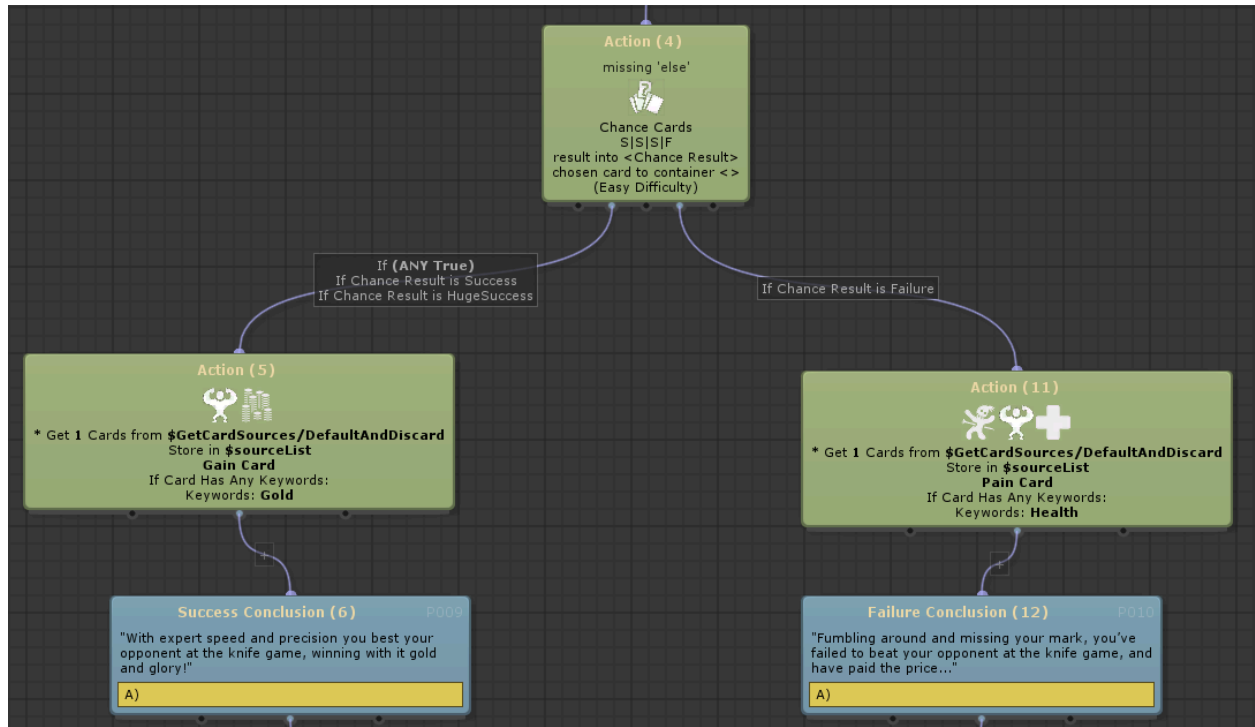


Now proceed to add this same condition task to the other connection, but this time set the **Chance Type** to be **Failure**, also you don't need to create a **Condition List** this time. Now that these condition tasks have been set, the system knows to follow the left branch if the chance mini game result is a success and likewise move down the right branch for a failure result. This is demonstrated in the example below.



# Creating the text for the results of the mini game:

Start by creating 2 new pages and connect one under each of the **Get Card(s) By Condition** actions we just created. It should look similar to the example below.



Let's rename the page on the left *"Success Conclusion"* and the page on the right *"Failure Conclusion"*.

Next let's add the text to both of these pages. Below you can find the text to enter, we've provided this for you.

### Success Conclusion (Page)

"With expert speed and precision you best your opponent at the knife game, winning with it gold and glory!"

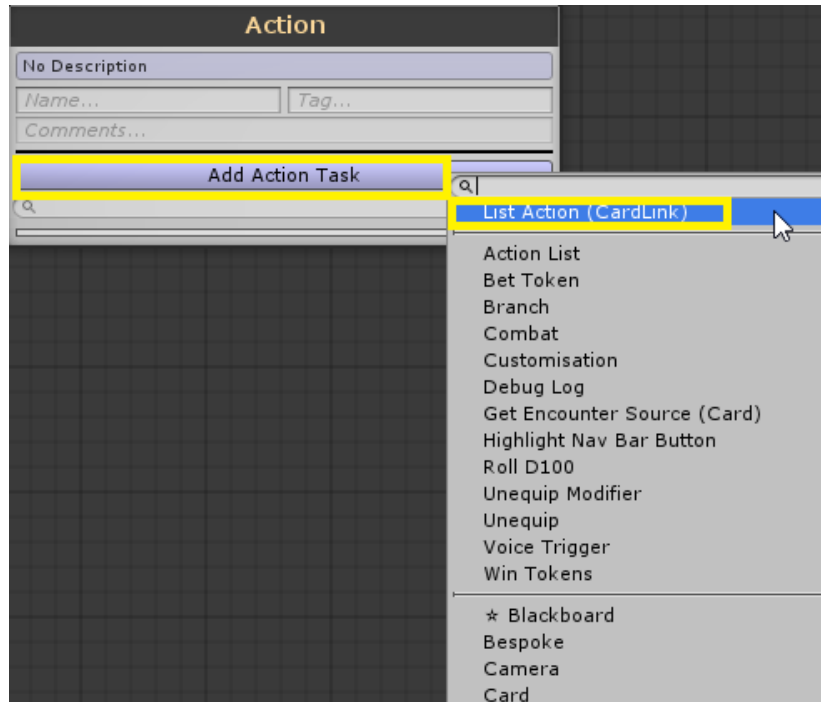
### Failure Conclusion (Page)

"Fumbling around and missing your mark, you've failed to beat your opponent at the knife game, and have paid the price..."

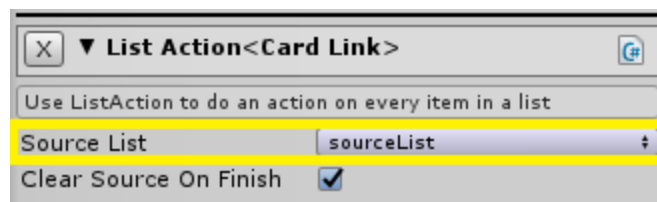
Next let's set up the pain and gain card activations.

## Activating the pain and gain cards:

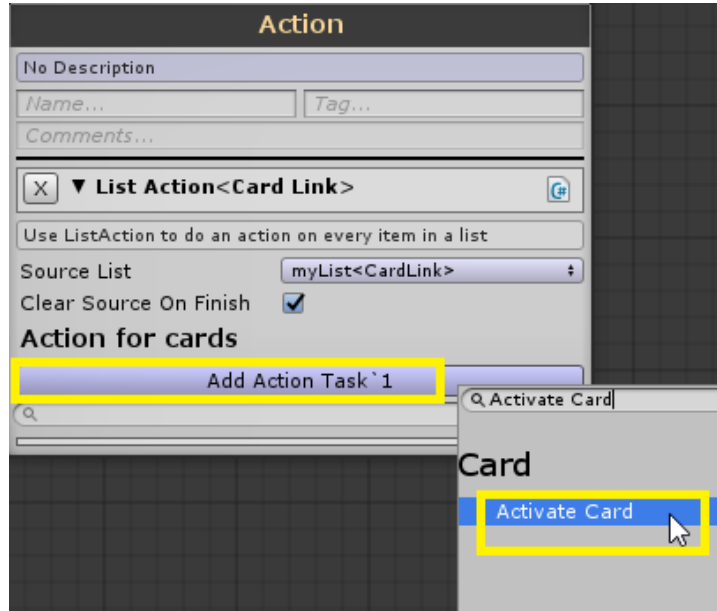
Let's start by creating a new action, and adding the action task **List Action (CardLink)**.



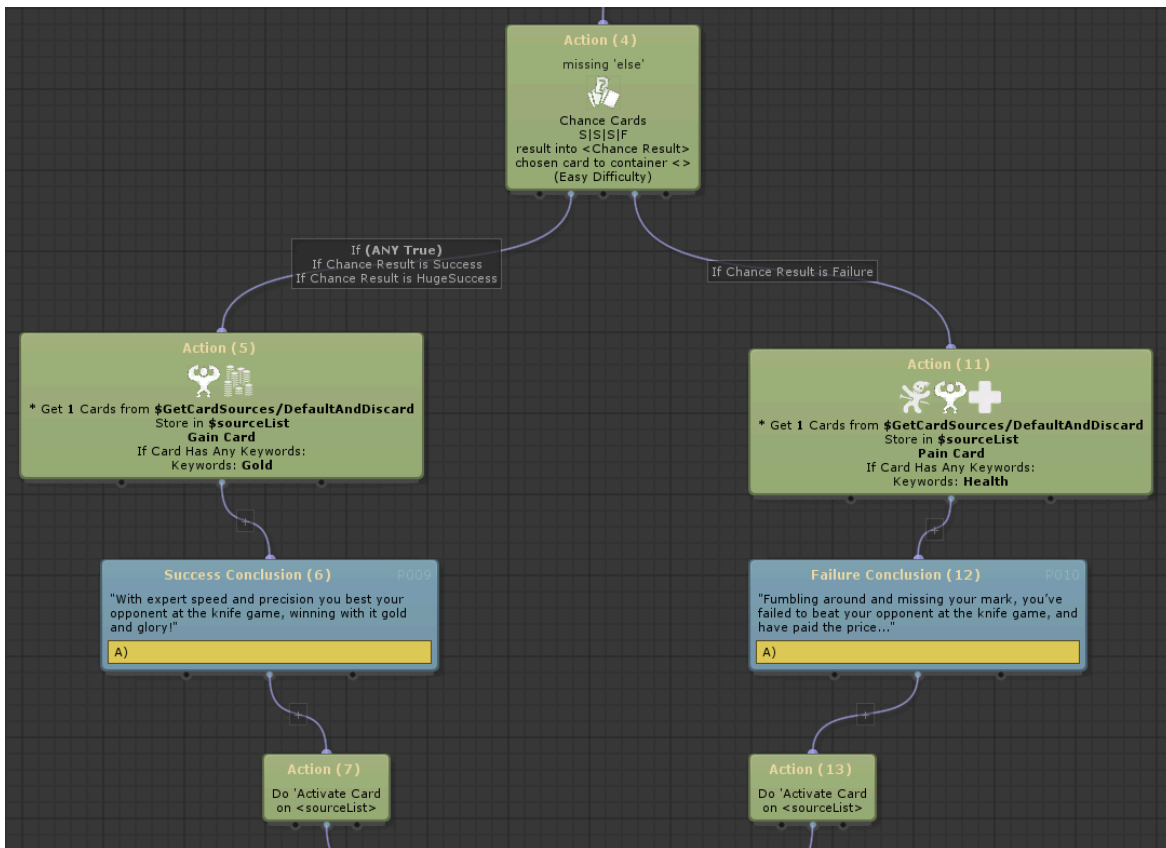
After creating this action task you'll need to assign the **Source List**. Let's assign this data with our **sourceList** variable (the one stored in our Local BB). This is demonstrated in the example below.



When you create the action task **List Action (CardLink)**, this requires you to create a second action task, that operates as the action for the cards stored in **sourceList**. Click **Add Action Task ' 1** and search for **Activate Card** and select it. This is demonstrated in the example below.

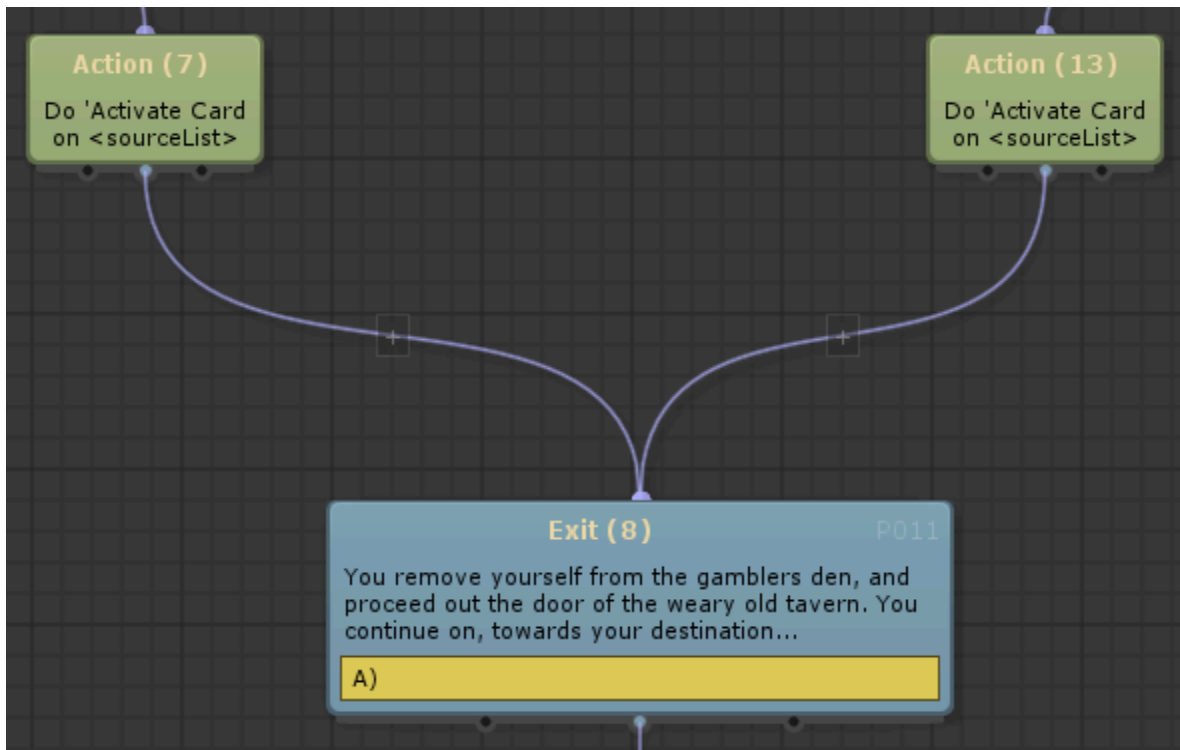


After you've done these steps, jump back into the encounter behavior tree and duplicate this action. Now link one of these under each of the pages you created before. This is demonstrated in the example below.



## Ending the encounter:

Let's look at ending this encounter. Start by creating a new page and connect it underneath both card activation actions. This is demonstrated in the example below.



Let's rename this page to "Exit".

Let's enter some text for the end of our encounter. Below you can find the text to enter, we've provided this for you.

### Exit (Page)

"You remove yourself from the gamblers den, and proceed out the door of the weary old tavern.  
You continue on, towards your destination..."

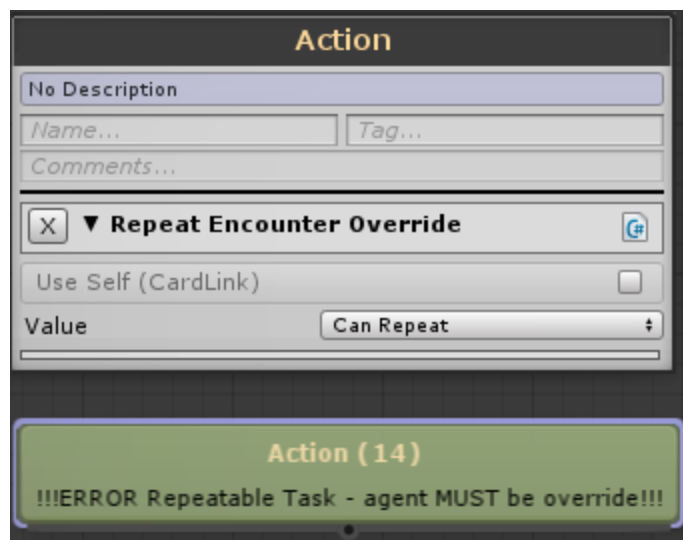
**Note:** Remember that this encounter is the transition card for the basic challenge that we'll be creating later in the documentation, so it should be ending the encounter but also be leading in its conclusion.

**Important Note:** If you're following this tutorial with the intention of having a single encounter card as your mod, you do not need to follow the remaining steps! The following steps are for making the encounter card into the stair cards used in the challenge tutorial.

---

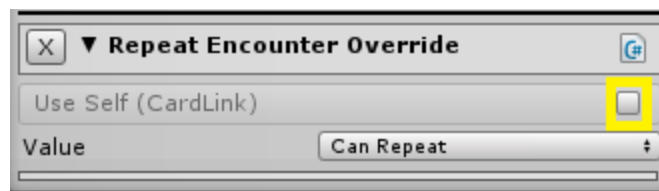
Now because this encounter is intended to be the transition card for our basic challenge, we need to set this encounter to be not repeatable after completion.

Let's create a new action and add the action task **Repeat Encounter Override**. This action and it's data are demonstrated in the example below.

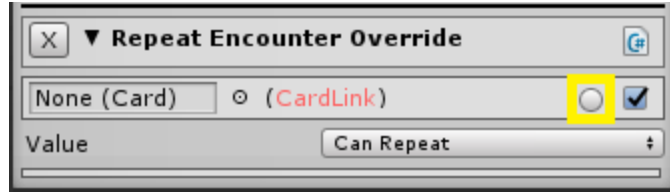


We want to set the **Value** to be **Cant Repeat**.

Let's check the small box highlighted in the below screen capture. After checking this box you'll have some new data to work with.

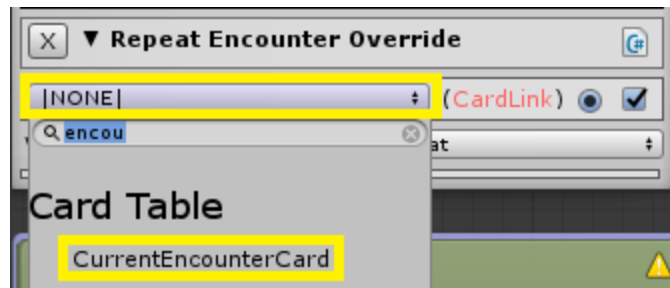


Let's now check the small circle just to the left of that box we just checked prior. This is demonstrated in the example below.

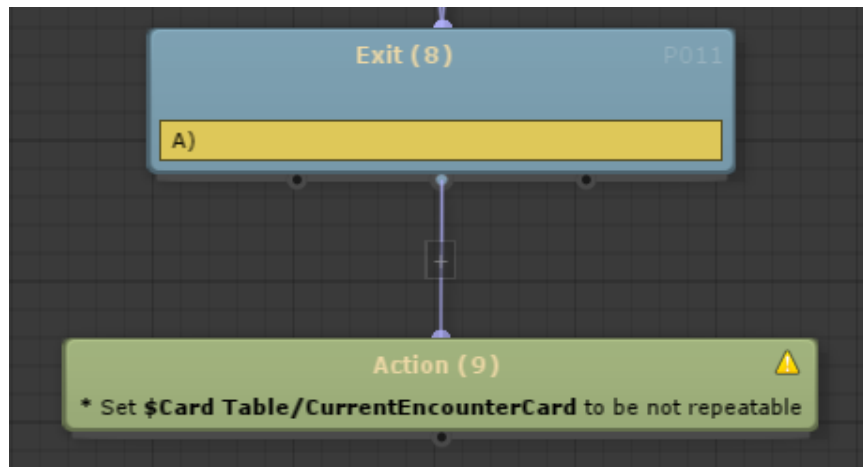


Once you've checked this circle, the data within the action task will change to a drop down list.

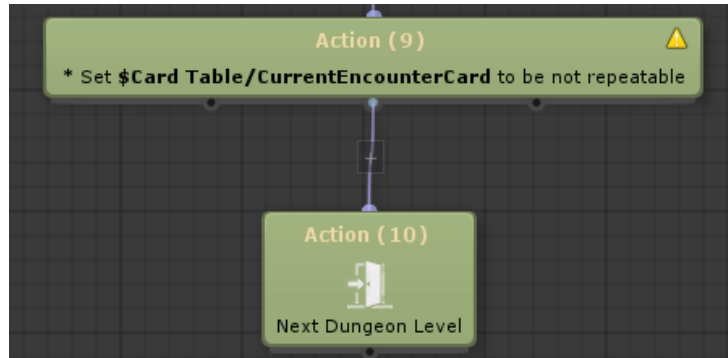
Enter the list and select **Card Table > CurrentEncounterCard**, and assign this. Alternatively you can search for the name **CurrentEncounterCard**, demonstrated in the example below.



After you've assigned the current encounter card to be not repeatable, let's connect this action under our exit page. This is demonstrated in the example below.



Directly under this action for setting the current encounter card to can't repeat, we have to create and connect a new action, which is going to be **Next Dungeon Level**. Create a new action and add the action task **Next Dungeon Level**. This tells the game to progress onto the next dungeon level at the end of this encounter. This is demonstrated in the example below.



Lastly, let's jump back to our encounter card and change one bit of data found under the **Encounter** label, to ensure that our card can be repeatable if the player does not finish the encounter and chooses to decline. You'll spot the data named **Can Repeat?**, we want to set this variable to **Can Repeat**, by default this is set to can't repeat.

Tute_TheBasicEncounter	
<b>Encounter</b> ▼	
Encounter	Encounter_Tute_TheBasicEncounter (Er) <input type="button" value="EDIT"/>
☆☆☆	
Global Blackboard	<input type="button" value="Create"/>
<b>Can Repeat?</b>	Can't Repeat
Sting	Neutral
Music Trigger	



## Afterthoughts and revision:

Throughout this tutorial you've learnt and been shown how to create a lot of the core features found in many of the encounters playable in HoF2, and through implementation and creating the tutorial encounter yourself. There is a lot more that can be done within the encounter behavior tree, and the encounter cards themselves, it's truly very expansive.

For more tips and tricks on how to implement more intermediate to advanced design to your encounter mods, refer to the ["Intermediate Encounter Design"](#) section of this modding documentation.

## Section 3: Creating “*The Basic Equipment*”

What is explored in this section of the documentation:

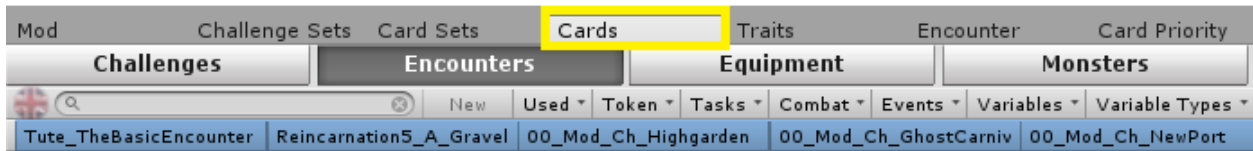
- [Creating new equipment cards:](#)
- [Creating the Basic Sword:](#)
- [Creating the Basic Artifact:](#)
- [Creating the Basic Shield:](#)

What are the equipment pieces we’re creating:

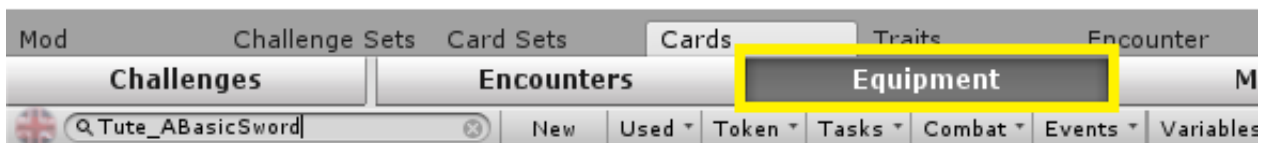
In this tutorial you’ll be creating three pieces of equipment that will be unlocked at the end of the basic challenge as the silver and gold token unlocks. These are going to be three basic equipment pieces, 1 sword that applies critical hits, 1 shield that reflects projectiles, and 1 artifact that heals you.

## Creating new equipment cards:

After opening the card editor, In order to start creating the “*Tute\_ABasicSword*”, “*Tute\_ABasicShield*”, and “*Tute\_ABasicArtifact*” equipment pieces, you’ll need to first create the equipment cards. Navigating across the top toolbar of the card editor, you’ll spot a tab named **Cards**. Click the **Cards** tab, doing so will display a new toolbar directly underneath this one.



Navigating across the new toolbar underneath, you’ll spot a tab named **Equipment**. Click the **Equipment** tab, doing so will display a list on the left of the card editor. This list contains all of the existing equipment cards.



while in the card editor you’ll spot a search bar, with a button next to it labelled **New**.



Click inside of the search bar and type the name “*Tute\_ABasicShield*” and click **New**, doing so will create a new equipment card with that name. Now repeat this step, and create the equipment cards “*Tute\_ABasicSword*” & “*Tute\_ABasicArtifact*”.

All of the equipment cards within the list are viewable. In order to view an equipment cards data, select the card that you wish to view. These cards are broken up into different lists for organizational purposes and ease of navigation, this is demonstrated in the below example, highlighted by the yellow box.

Weapon (67)	AbsolvedPlight_v1	<input type="radio"/>
Shield (18)	AbsolvedPlight_v2	<input type="radio"/>
Armour (16)	AdvancedAxe	<input type="radio"/>
Helm (19)	AdvancedSword	<input type="radio"/>
Gloves (0)	AdvancedSword_noToken	<input type="radio"/>
Artifact (27)	ArcaneEdge	<input type="radio"/>
Ring (26)	AssassinsBlade	<input type="radio"/>
Unassigned (0)	BasicAxe	<input type="radio"/>

Whenever you create a new equipment card, it will always be found in the **Unassigned** category of the existing equipment cards list.

Weapon (67)	AbsolvedPlight_v1	<input type="radio"/>
Shield (18)	AbsolvedPlight_v2	<input type="radio"/>
Armour (16)	AdvancedAxe	<input type="radio"/>
Helm (19)	AdvancedSword	<input type="radio"/>
Gloves (0)	AdvancedSword_noToken	<input type="radio"/>
Artifact (27)	ArcaneEdge	<input type="radio"/>
Ring (26)	AssassinsBlade	<input type="radio"/>
Unassigned (0)	BasicAxe	<input type="radio"/>

After you've created your 3 equipment cards we'll look at assigning them to their relevant lists. You should have 3 equipment cards currently in the **Unassigned** category, similar to the example below.

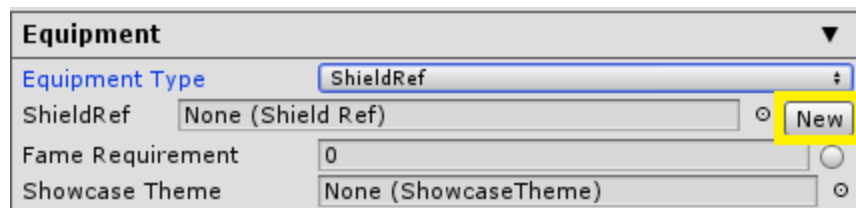
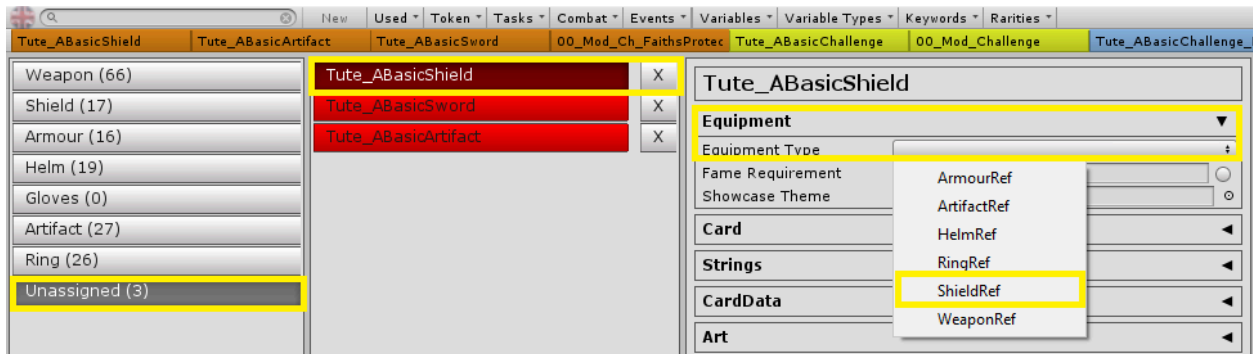
Weapon (66)	Tute_ABASICShield	<input checked="" type="checkbox"/>
Shield (17)	Tute_ABASICSword	<input checked="" type="checkbox"/>
Armour (16)	Tute_ABASICArtifact	<input checked="" type="checkbox"/>
Helm (19)		
Gloves (0)		
Artifact (27)		
Ring (26)		
Unassigned (3)		

When you select one of these unassigned equipment cards, and proceed to view the data of that card, you'll notice (at the very top of the selected cards data) that there is a label named **Equipment**, and within that, there is a drop down list named **Equipment Type**. **Equipment Type** allows you to assign an equipment reference to any unassigned equipment cards.

For example, "*Tute\_ABasicShield*" would need to be assigned with the **ShieldRef** equipment type, so that the card knows that it is a shield equipment piece. After assigning the card with the **ShieldRef** equipment type, you'll need to create a new reference. To do this, click the **New** button next to the **ShieldRef** label. Now, the "*Tute\_ABasicShield*" equipment card will be assigned to the **Shield** list of equipment cards.

**Note:** Lists don't refresh or update straight away, so try clicking into a new tab and returning to this one if your equipment cards have not moved into the category you've assigned.

These steps are demonstrated in the examples below. After completing these steps appropriately for the 3 cards we've just created, we can then move onto adding the data into our equipment cards.



Tute_ABasicShield		Tute_ABasicArtifact	Tute_ABasicSword	00_Mod_Ch_FaithsP
Weapon (66)	BastionOfPurification			O
Shield (18)	BrigandsShield			O
Armour (16)	BroadShield			O
Helm (19)	Buckler			O
Gloves (0)	GoldenShield			O
Artifact (27)	ImperialShield			O
Ring (26)	NazgarsReckoning			O
Unassigned (2)	RamShield			O
	ShadowOfTheEmpire			O
	ShieldOfAegle			O
	SouthernJustice			O
	WanderersBounty			O
	FortitudesBreath			O
	Innocence			O
	RustyShield			O
	Shield			O
	00_Mod_Ch_FaithsProtector			X
	Tute_ABasicShield			X

## Creating the Basic Sword:

Now let's create the "Tute\_ABasicSword" equipment card. You'll find it under the **Weapon** category.

Weapon (67)	DevilsTouch_v2	0
Shield (18)	DuellingDaggers	0
Armour (16)	ElvenStaff	0
Helm (19)	ExecutionersAxe	0
Gloves (0)	ExquisiteBlade	0
Artifact (27)	ExquisiteFake	0
Ring (26)	FameTest	0
Unassigned (1)	FrostFang	0
	ImperialMight	0
	HrethasIre	0
	HugeHammer	0
	HumbleHammer	0
	HuntersBlades	0
	LawAndOrder	0
	LawAndOrder_v2	0
	NeglectedSword	0
	NorthernerBlade	0
	QuestingMace0	0
	QuestingMace1	0
	QuestingMace10	0
	QuestingMace2	0
	QuestingMace3	0
	QuestingMace4	0
	QuestingMace5	0
	QuestingMace6	0
	QuestingMace7	0
	QuestingMace8	0
	QuestingMace9	0
	RustedAxe	0
	RustedBlades	0
	SacrificialBlades	0
	ScorchedBlade	0
	SistersOfVengeance	0
	SpikedMace	0
	ThePeeler	0
	Deatharangs	0
	ThunaersTalons	0
	00_Mod_Ch_OrnamentalBlade	X
	Tute_ABasicSword	X

Select the card and you'll be able to view the card's data. Highlighted below are the points of data that we'll need to change.

Tute_ABASICSword	
<b>Equipment</b>	
Equipment Type	WeaponRef
WeaponRef	Tute_ABASICSwordREF 1 (WeaponRef) [New]
<b>Tute_ABASICSwordREF 1</b>	
Script	WeaponRef
Skip Intro	<input type="checkbox"/>
Intro	None (ShowCaseDefinition)
Show Case Desc	
Weapon	BasicSword
Damage	10
Speed	0
Category	Sword
Handedness	One Handed
<b>AbilityRef</b>	
Script	AbilityRef
Cooldown	0
Gold Cost	0
Health Cost	0
Attack Counter Target	10
Fame Requirement	0
Showcase Theme	None (ShowcaseTheme)
<b>Card</b>	
Strings	
CardData	
Art	
Config	
Keywords	
Traits	

Let's start by assigning an existing model to our equipment card, and an appropriate **Damage** rating. Followed by assigning the appropriate **Category** & **Handedness** for our sword.

Weapon	Tute_ABASICSword 1
Damage	0
Speed	0
Category	
Handedness	



First the **Weapon** data. This is going to be the model that we assign to our equipment card. In order to view the existing models, select the small circle highlighted in the example below.

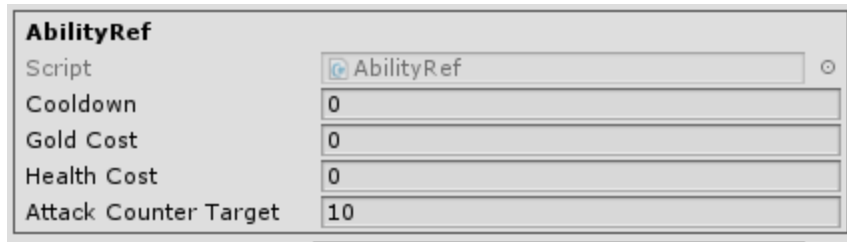
Weapon	BasicSword	⊙
Damage	10	
Speed	0	
Category	Sword	⬇
Handedness	One Handed	⬇

This will display a list of all the available equipment models in HoF2. Let's select the sword named "*BasicSword*". Next we will need to assign the **Damage** for our weapon, let's go ahead and set that value to **10**.

Next we will need to assign the **Category** for our weapon, let's go ahead and set this to **Sword**.

Then let's assign the **Handedness** for our weapon, let's go ahead and set this to **One Handed**.

Let's take a look at the **AbilityRef** data on the equipment card. This requires us to set an appropriate **Cooldown**, **Gold Cost**, **Health Cost**, **Attack Counter Target** for the ability attached to our weapon.



AbilityRef	
Script	AbilityRef
Cooldown	0
Gold Cost	0
Health Cost	0
Attack Counter Target	10

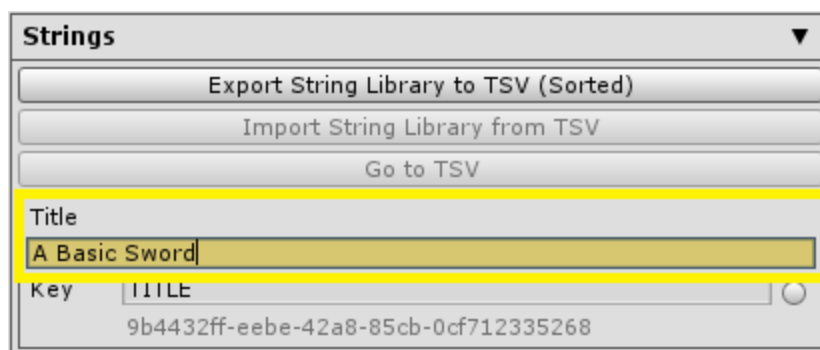
Let's start by assigning the **Attack Counter Target** for our weapon, this is the amount of consecutive strikes that players will have to reach, to be able to use the weapon's ability. We'll set this value to **10**, which is good starting ground.

**Cooldown** is associated with the amount of time that needs to be passed after a weapon ability has been used by players until they can use it again. **Cooldown** doesn't really apply, if you're using an **Attack Counter Target**. This data is typically used when you're creating **Artifact** mods.

**Gold Cost** is the amount of gold that it costs the player to use the ability on the equipment card.

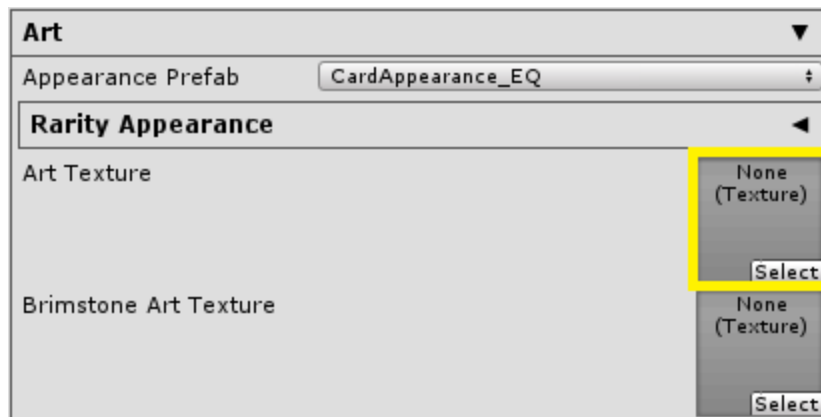
**Health Cost** is the amount of health that it costs the player to use the ability on the equipment card.

Next we'll change the name of our weapon equipment card. To do this, expand the **Strings** label and enter the title "*A Basic Sword*", or whatever you'd like it to be, just make sure you remember it!

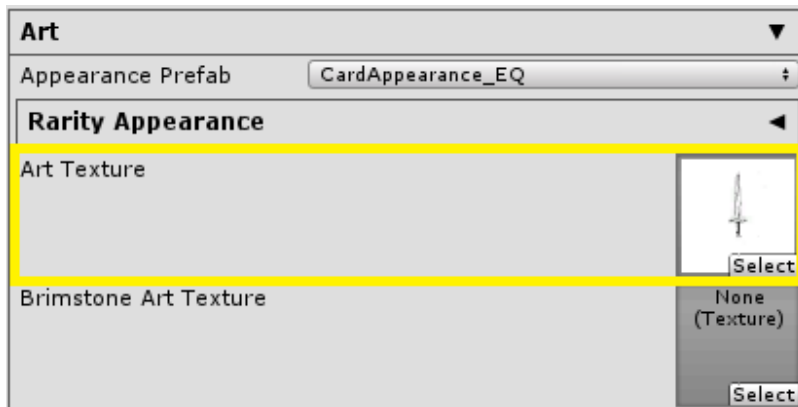


Strings	
Export String Library to TSV (Sorted)	
Import String Library from TSV	
Go to TSV	
Title	A Basic Sword
Key	TITLE
	9b4432ff-eebe-42a8-85cb-0cf712335268

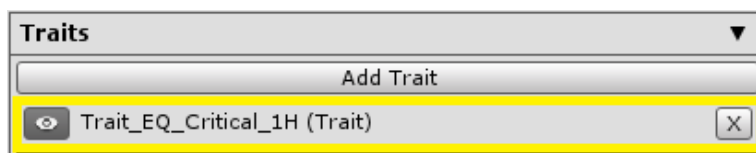
Next we'll assign an **Art Texture** to our weapon equipment card. Expand the **Art** label and enter the list of available art textures.



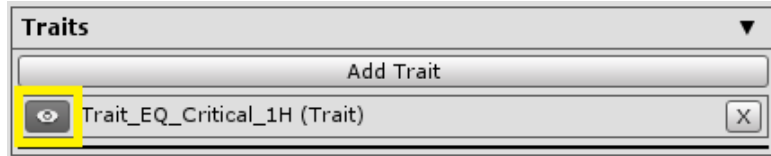
Search for the art texture *"c\_eq\_sword\_basic"* and select this texture, assigning it to the card's data for our sword. Alternatively, you can set this to whatever you'd like.



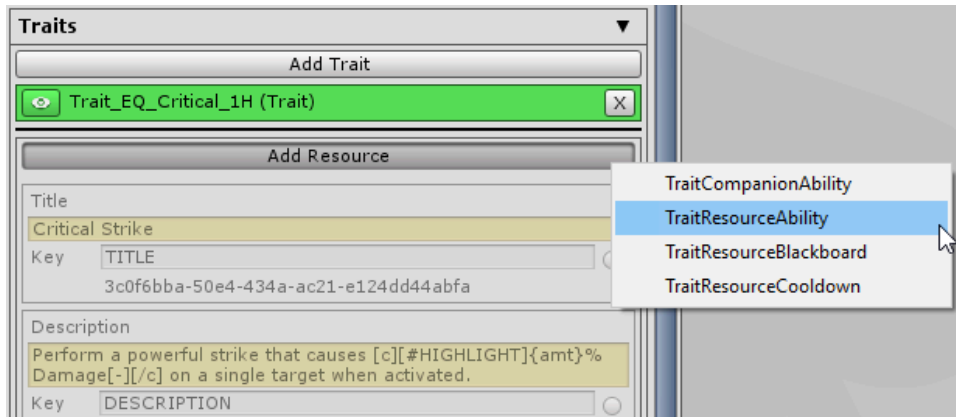
Finally, let's add an existing trait to our sword. To do this, expand the **Traits** label and let's add a new trait by clicking the button **Add Trait**. Let's search for the trait *"Trait\_EQ\_Critical\_1H"* and add this to our equipment cards data. *"Trait\_EQ\_Critical\_1H"* allows players to perform a powerful strike on a single enemy, when activated.



Now, the thing with existing traits, is that you need to set up the variables that they talk to. Start by expanding this trait by clicking the eye button demonstrated below.



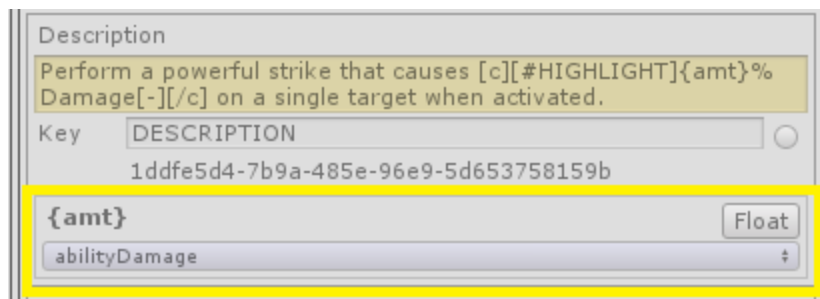
Your expanded view should look like the example below. The only thing that you have to do here is click **Add Resource**. This is where you add the appropriate resource type for the trait you're using. The type that you need to assign is the *"TraitResourceAbility"*; this is demonstrated below.



**Note:** Not all traits need a resource, typically only ones that talk with other variables that you have to set, need this resource.

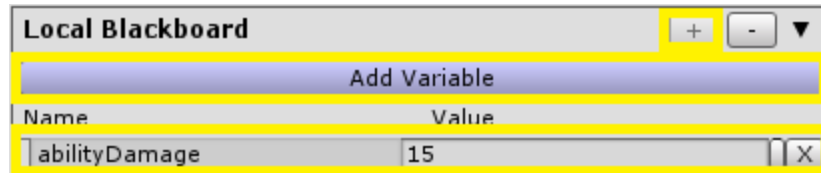
After you've added the **TraitResourceAbility** resource, you'll spot a piece of data within this trait that takes a float variable named **abilityDamage**.

This has already been created in the existing trait, and this variable is what the trait is looking for in order to apply damage to the ability.



We need to set this **abilityDamage** float variable, within the equipment card itself. To do this, you'll notice a section of this equipment card labeled **Local Blackboard**.

You can create the new Local BB by clicking the small addition button. Next, create the new float variable by clicking **Add Variable**. Once created, name it **abilityDamage** and set a value for the percentage of damage to be applied to the enemy target when struck.

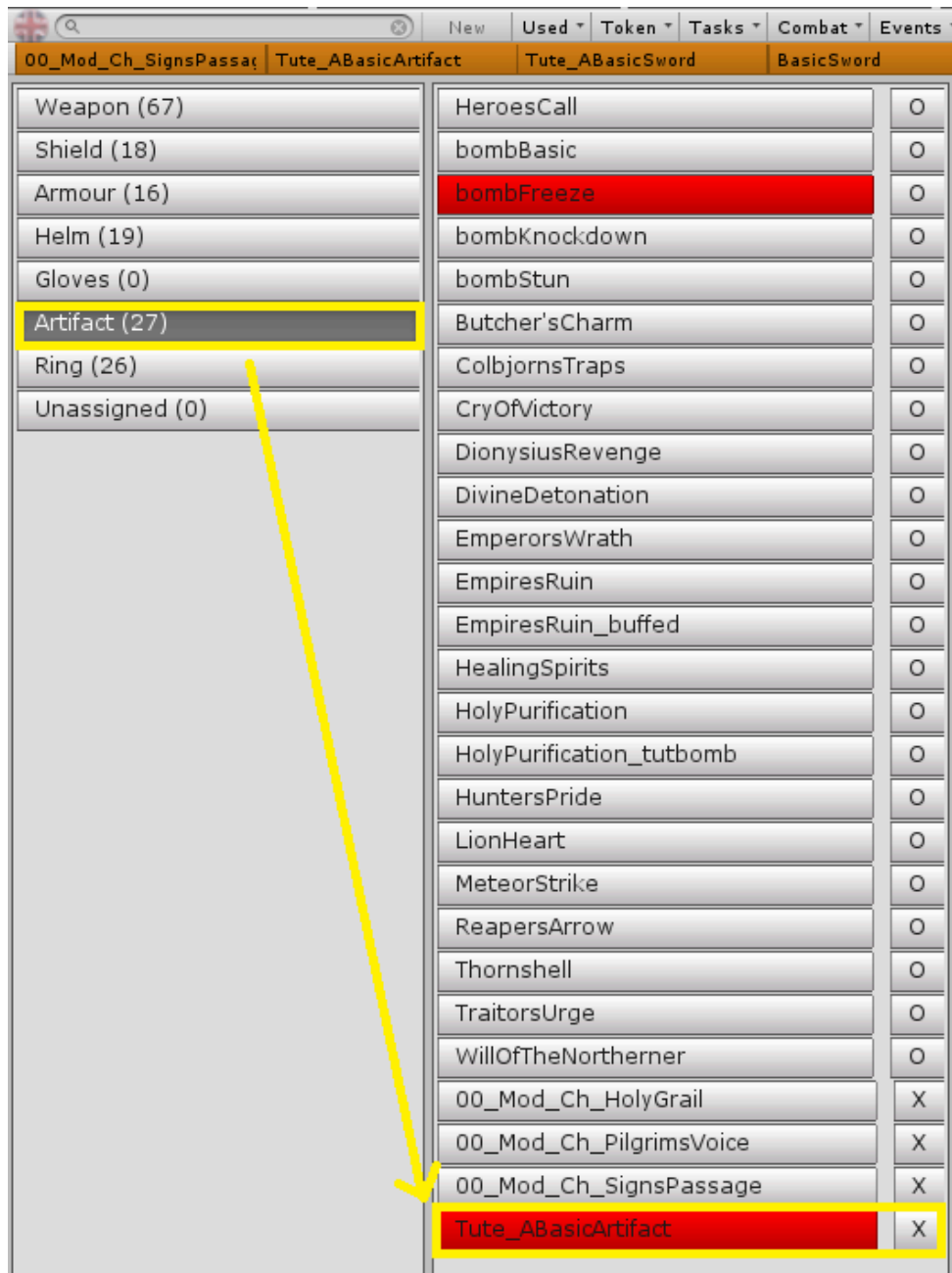


To demonstrate how this translates to in game, here is the basic sword when it's viewed by players through the inventory, after all of the above has been completed.



## Creating the Basic Artifact:

Now let's create the "Tute\_ABasicArtifact" equipment card next.



Select the card and you'll be able to view cards data. Highlighted below are the parts of data that we'll need to change when creating our artifact.

The screenshot shows a configuration form for 'Tute\_ABASICArtifact'. The form is organized into several sections, each with a yellow highlight:

- Equipment**: Contains 'Equipment Type' (ArtifactRef), 'ArtifactRef' (Tute\_ABASICArtifactREF (ArtifactRef)), and a 'New' button.
- Tute\_ABASICArtifactREF**: Contains 'Script' (ArtifactRef), 'Skip Intro' (checkbox), 'Intro' (None (ShowCaseDefinition)), 'Show Case Desc' (radio button), 'Artifact' (None (Artifact) - highlighted in red), and 'Quantity' (0).
- AbilityRef**: Contains 'Script' (AbilityRef), 'Cooldown' (0), 'Gold Cost' (0), 'Health Cost' (0), and 'Attack Counter Target' (0).
- Card**: A section header with a left-pointing arrow.
- Strings**: A section header with a left-pointing arrow.
- CardData**: A section header with a left-pointing arrow.
- Art**: A section header with a left-pointing arrow.
- Config**: A section header with a left-pointing arrow.
- Keywords**: A section header with a left-pointing arrow.
- Traits**: A section header with a left-pointing arrow.

Let's start by assigning an existing model to our equipment card, and an appropriate **Quantity** value for the item.

This is a close-up view of the 'Tute\_ABASICArtifactREF' section from the previous screenshot. The 'Artifact' field is now set to 'Tute\_ABASICArtifact' and is highlighted in yellow. The 'Quantity' field remains at '0'.

First the **Artifact** data. This is going to be the model that we assign to our equipment card. In order to view the existing models, select the small circle highlighted in the example below.

Tute_ABASICArtifactREF	
Script	ArtifactRef
Skip Intro	<input type="checkbox"/>
Intro	None (ShowCaseDefinition)
Show Case Desc	<input type="radio"/>
Artifact	Tute_ABASICArtifact
Quantity	0

This will display a list of all the available equipment models in HoF2. Let's select the artifact named "*HealingSpirits*". Alternatively, you can set this to whatever artifact model you like!

Next, let's assign the **Quantity** value for our artifact, let's go ahead and set this value to **3**.

Let's take a look at the **AbilityRef** data for this equipment card. This requires us to set an appropriate **Cooldown**, **Gold Cost**, **Health Cost**, **Attack Counter Target** for our artifact.

AbilityRef	
Script	AbilityRef
Cooldown	0
Gold Cost	0
Health Cost	0
Attack Counter Target	10

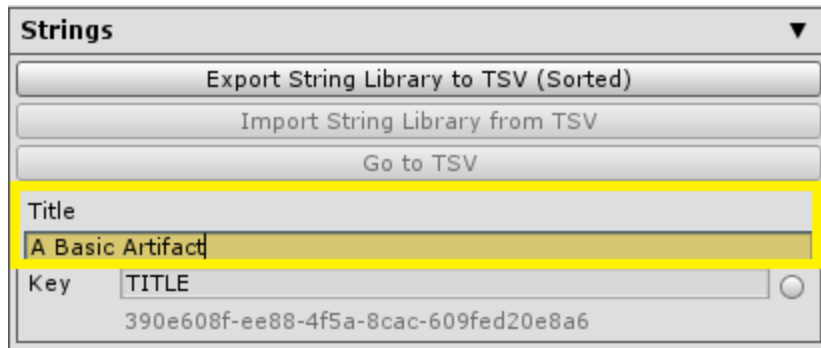
**Attack Counter Target** does not apply for our artifact, so this can be **0**.

**Cooldown** does apply, so let's set this value to be around **5-7**, the choice is totally yours!

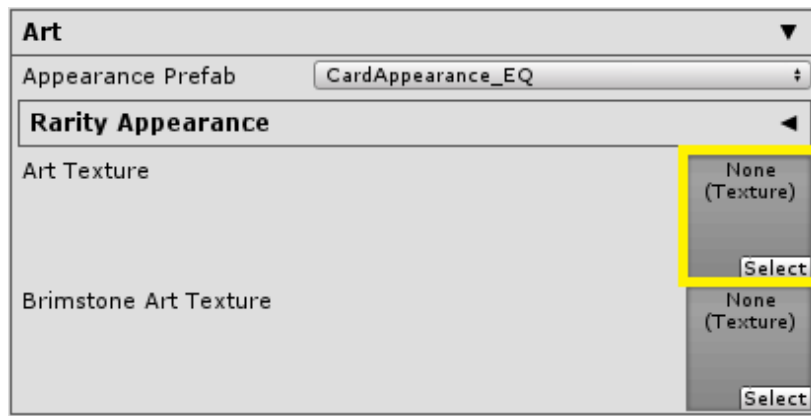
**Gold Cost & Health Cost** we don't need to worry about so let's leave them at **0**.



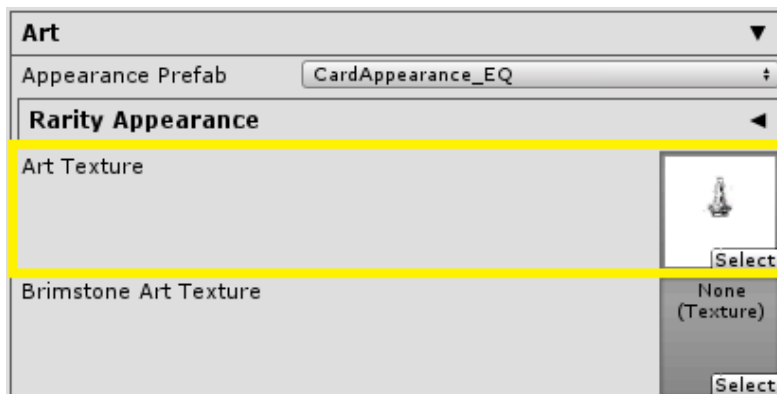
Let's start by changing the name of our equipment card. To do this, expand the **Strings** label and enter the title "A Basic Artifact", or you can name the artifact whatever you'd like.



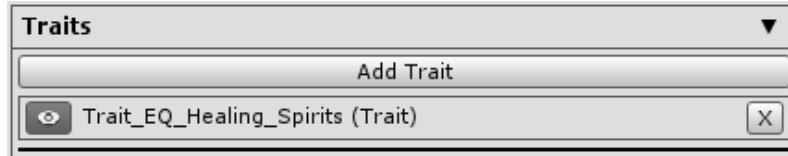
Next, let's assign an **Art Texture** to our equipment card. To do this, expand the **Art** label and enter the list of available art textures.



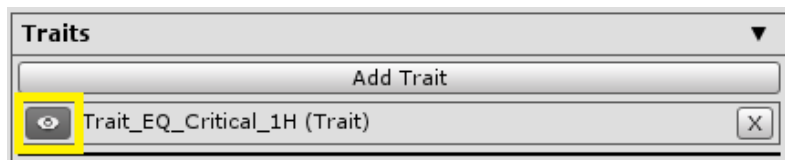
Search for the art texture "c\_art\_vial" and select this texture, assigning it to the cards data for our artifact. Alternatively, you can assign whichever texture you'd like to use for your artifact.



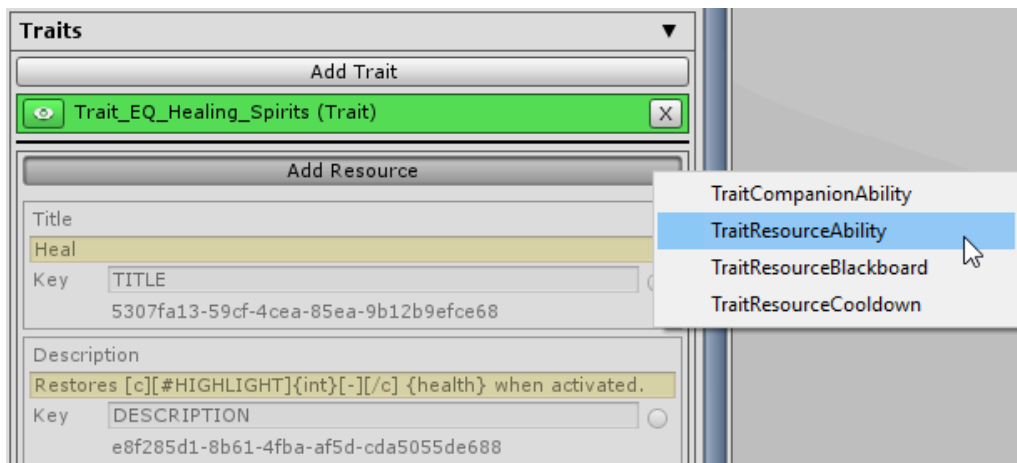
Next, let's add the existing trait to our artifact. To do this, expand the **Traits** label and let's add a new trait by clicking the button **Add Trait**. Let's search for "*Trait\_EQ\_Healing\_Spirits*", and add this to our equipment card.



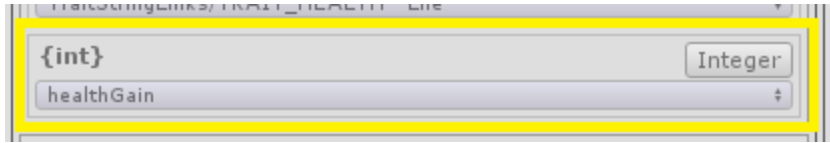
Now, the thing with existing traits, is that you need to set up the variables that they talk to. Start by expanding this trait, by clicking the eye button demonstrated below.



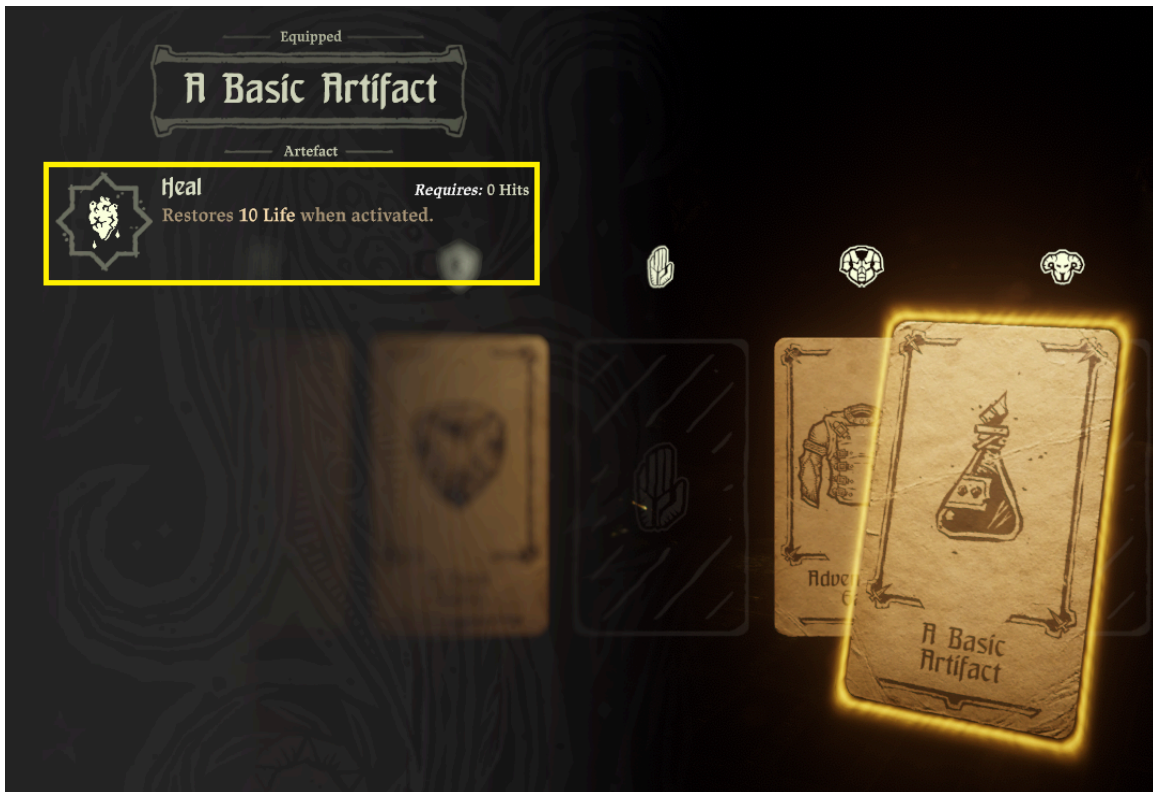
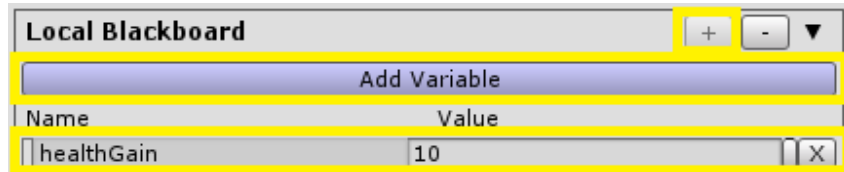
Your expanded view should look like the example below. The only thing that you have to do here is click **Add Resource**. This is where you add the appropriate resource type for the trait you're using. The type that you need to assign is the **TraitResourceAbility**, this is demonstrated below.



After you've added the **TraitResourceAbility** resource, you'll spot a piece of data within this trait, that takes an integer variable named **healthGain**. This has already been created in the existing trait, and this variable is what the trait is looking for in order to apply health to the player when the artifact is used.

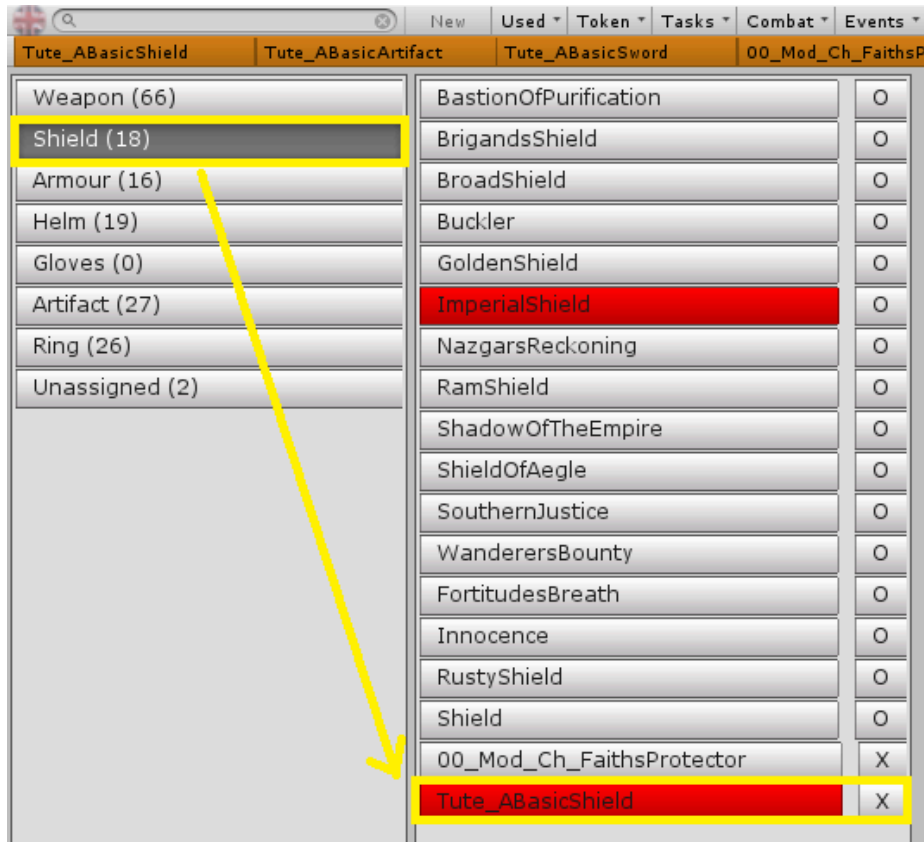


We need to set this **healthGain** integer variable within the equipment card itself. To do this, you'll notice a section of this equipment card labelled **Local Blackboard**. You can create the new Local BB by clicking the small addition button. Next, create the new **integer variable** by clicking **Add Variable**. Once created, name it **healthGain** and set a value for the amount of health to be gained by the player on use.



## Creating the Basic Shield:

Lastly, let's create the "Tute\_ABasicShield" equipment card.



Select the card and you'll be able to view the cards data. Highlighted below are the parts of data that we'll need to enter.

Tute_ABasicShield	
<b>Equipment</b> ▼	
Equipment Type	ShieldRef
ShieldRef	Tute_ABasicShieldREF 1 (ShieldRef) <span>New</span>
<b>Tute_ABasicShieldREF 1</b>	
Script	ShieldRef
Skip Intro	<input type="checkbox"/>
Intro	None (ShowCaseDefinition)
Show Case Desc	<input type="radio"/>
Defence Value	15
Shield	BasicShield
Fame Requirement	0
Showcase Theme	None (ShowcaseTheme)
<b>Card</b> ◀	
<b>Strings</b> ◀	
<b>CardData</b> ◀	
<b>Art</b> ◀	
<b>Config</b> ◀	
<b>Keywords</b> ◀	
<b>Traits</b> ◀	
<b>Shards</b> ◀	
Token	<input type="button" value="+"/> <input type="button" value="-"/> ◀
<b>Event Listeners (0)</b> ◀	
<b>Meta Modifiers (0)</b> ◀	
Local Blackboard	<input type="button" value="+"/> <input type="button" value="-"/> ◀
<b>Voice Clips</b> ◀	
<b>Referenced by 0 cards/cardsets/addons</b> ◀	

Let's start by assigning an existing model to our equipment card, and an appropriate **Defence** rating.

The screenshot shows the configuration interface for the equipment card 'Tute\_ABasicShield'. The 'Equipment' section is expanded, showing various fields. The 'Shield' field is highlighted in yellow and contains the value 'Tute\_ABasicShield 1'. Other fields include 'Defence Value' (0), 'Fame Requirement' (0), and 'Showcase Theme' (None (ShowcaseTheme)).

First things first, the **Shield** data. This is going to be the model that we assign to our equipment card. In order to view the existing models, select the small circle highlighted in the below.

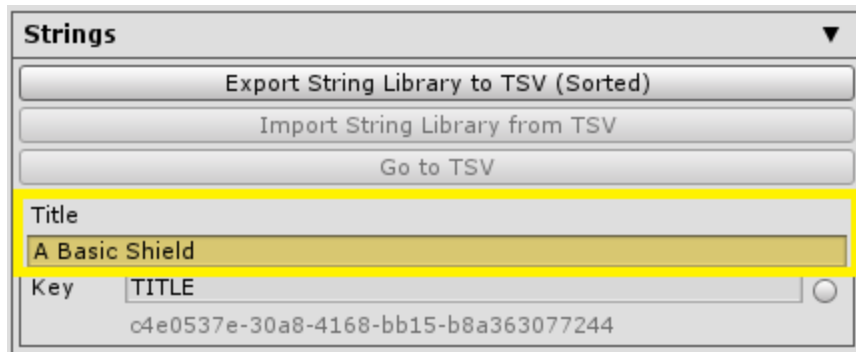
This is a close-up of the 'Shield' field in the configuration interface. The field contains the value 'Tute\_ABasicShield 1'. A small circle next to the field name is highlighted in yellow, indicating that it should be clicked to view the available models.

This will display a list of all the available equipment models in HoF2. Let's select the shield named "*BasicShield*", or whichever you'd like to choose as your shield. Now, let's assign the **Defence** rating for our shield. There are a few guidelines that we employ for setting defence values across all of the different types of equipment in HoF2.

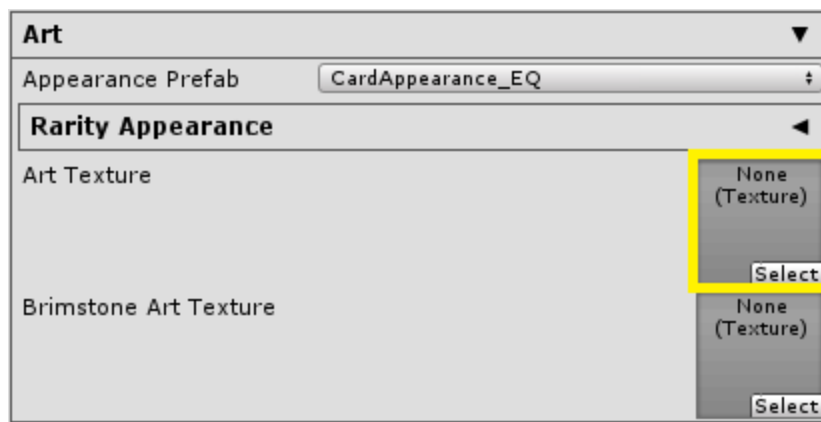
- **Armour:** 5 - 25 (*Light: 5 | Medium: 10 - 15 | Heavy: 20 - 25*)
- **Shield:** 10 - 25
- **Helm:** 0 - 15
- **Artefact:** n/a
- **Ring:** 0 - 10

Now, go ahead and set the **Defence** rating for the shield to be between 10 - 25, it's totally up to you!

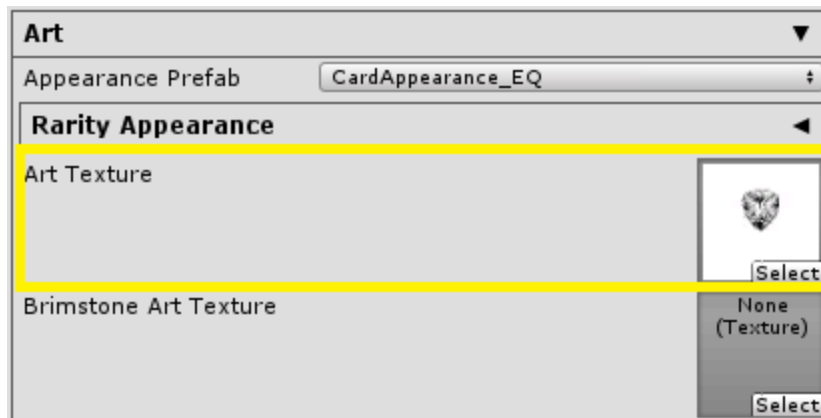
Next, let's change the name of our equipment card. To do this, expand the **Strings** label and enter the title "A Basic Shield", or whatever you'd like it to be for your shield.



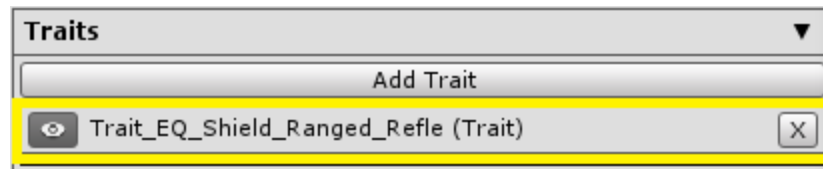
Now, let's assign an **Art Texture** to our equipment card. To do this, expand the **Art** label and enter the list of available art textures. Demonstrated in the example below.



Search for the art texture "c\_eq\_shield\_basic" and select this to be your texture. If you'd like to set your own choice, go for it!



Next, let's add some existing traits to our shield. To do this, expand the **Traits** label and let's add a new trait by clicking the button **Add Trait**. Let's search for the trait "*Trait\_EQ\_Shield\_Ranged\_Refle*" and add this to our equipment cards data.



There is no need for adding resources to this trait, so that's the shield done!

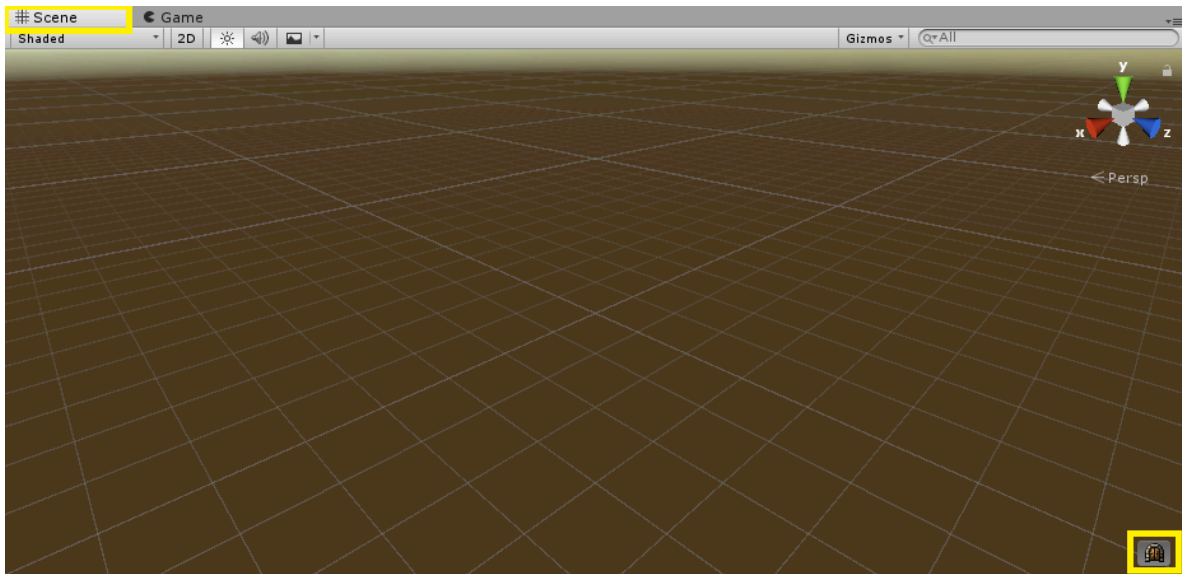
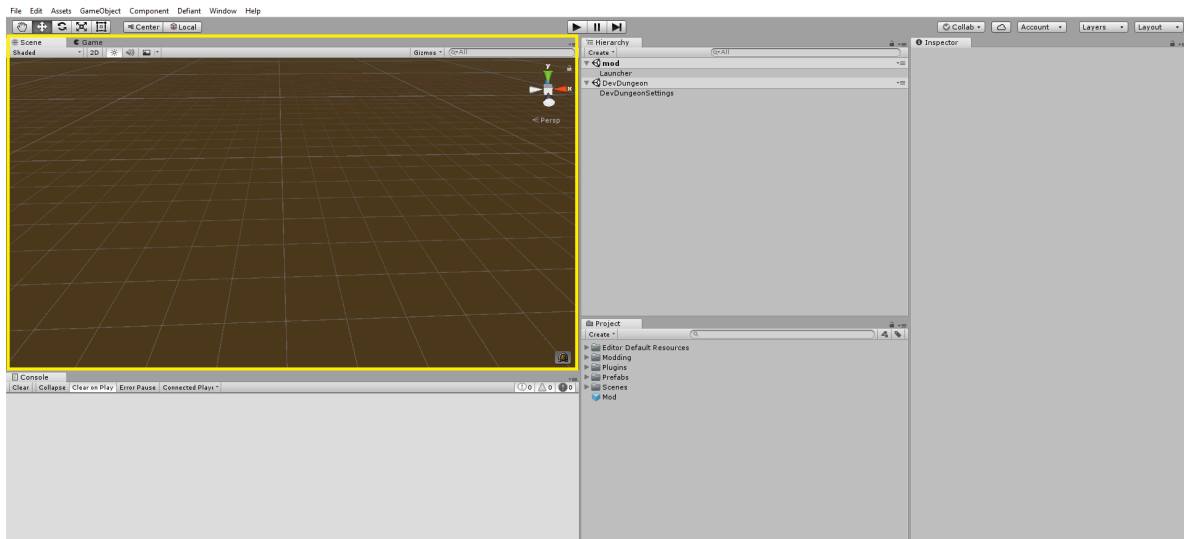


## Testing your equipment mods:

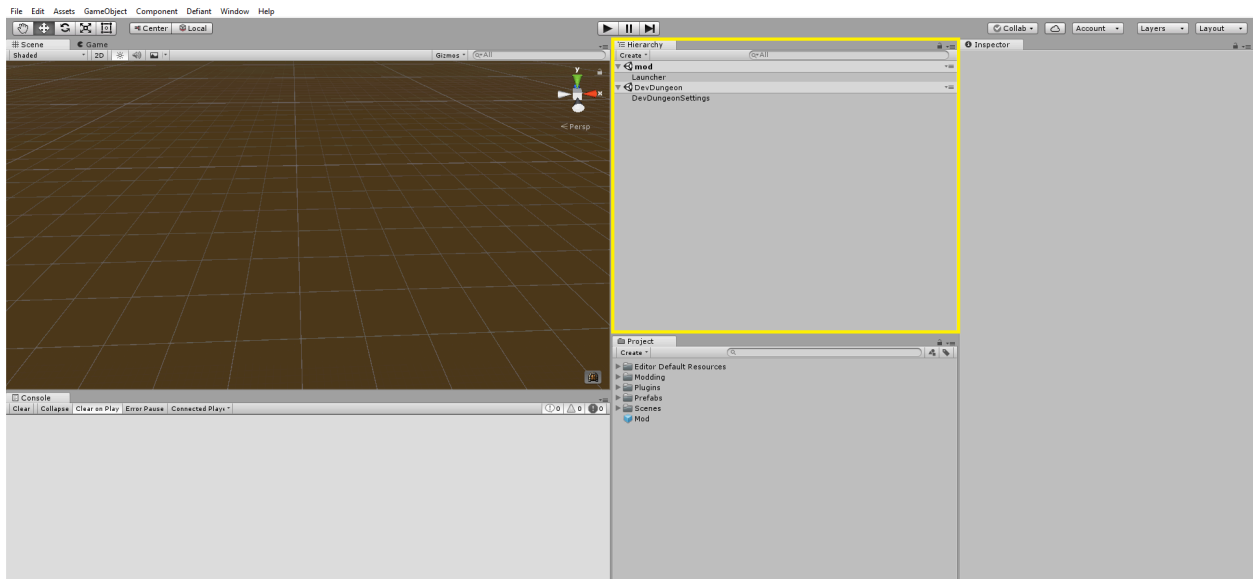
Throughout these tutorials you've been shown how to create 3 different equipment pieces found in HoF2. There is a lot more that can be done with creating equipment mods so experiment!

The easiest way to test your equipment mods is to run them directly through the **DevDungeon**, and test them in combat, through playing an encounter named **Test Combat**.

If you look in the scene view of the Unity editor (below), you'll spot a small dungeon door icon.

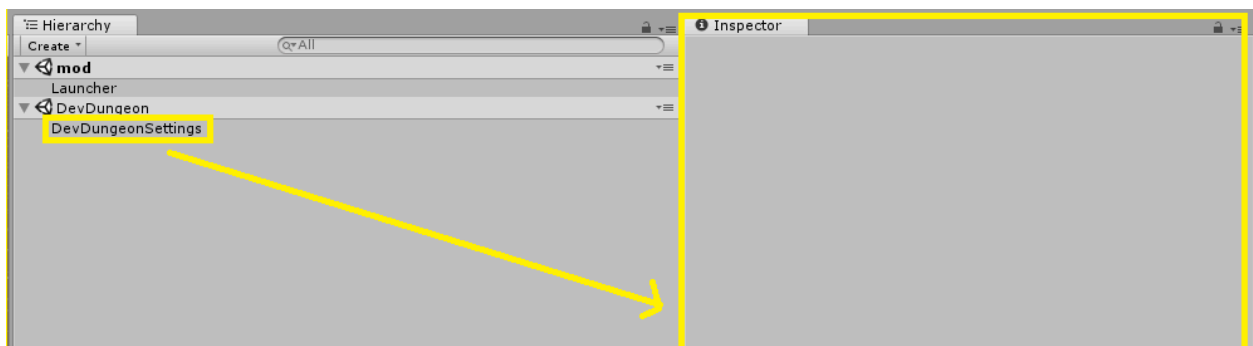
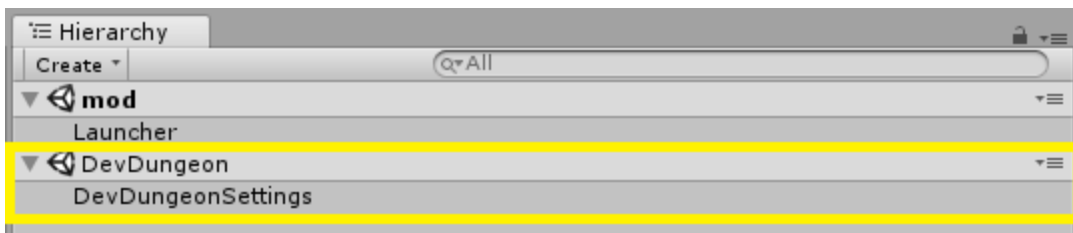


After clicking this dungeon door icon, navigate over to your **Hierarchy** which will now have a new gameObject named **DevDungeon** (below).

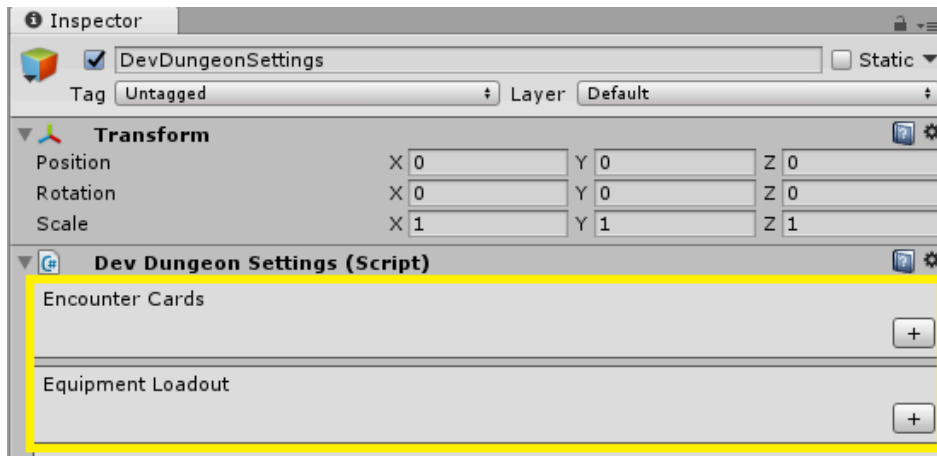


This **DevDungeon** gameObject is expandable, expanding it will display **DevDungeonSettings**.

Select the **DevDungeonSettings** and navigate over to the unity inspector. This is where we'll set up the **Test Combat** encounter along with our equipment mods that we want to test.

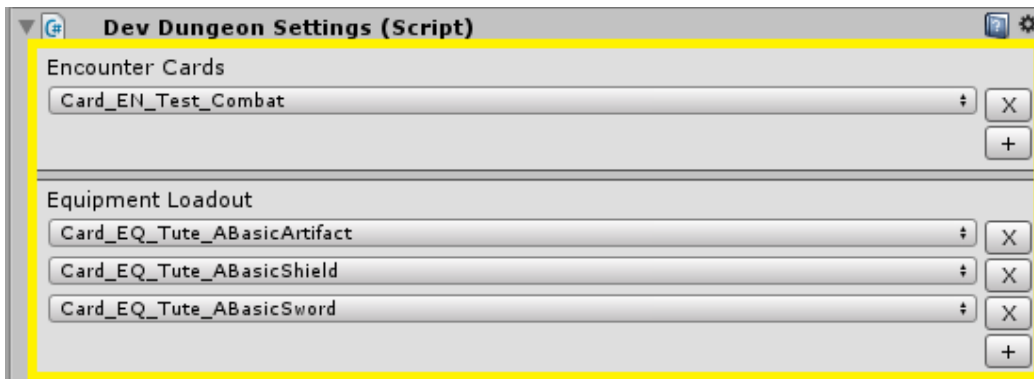


The **DevDungeonSettings** will look like below (viewed in the inspector). We'll be focusing on that data which is highlighted below.



**Encounter Cards** is where we'll add the **Test Combat** encounter. After you've clicked the small addition symbol in the bottom right of this data section, search for the encounter card named *"Card\_EN\_Test\_Combat"* and assign it here, this is the encounter that you'll be playing.

**Equipment Cards** is where we'll add the equipment mods that you want to test, by assigning them here they will be in your character's inventory when you start the game and proceed to enter the **Test Combat** encounter. This is what makes **DevDungeon** and **Test Combat** super useful when testing any equipment mods.



Click **File > Save Scenes** to save your changes to the settings. Now open the **Mod Editor** window by pressing **F5** and click **'Play'** to test your equipment mods!

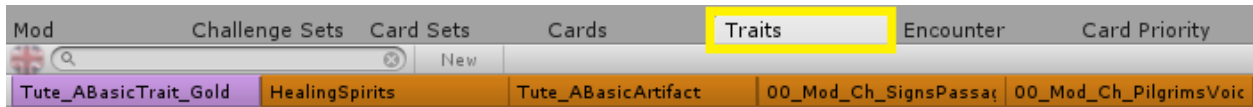
## Section 4: Creating *“The Basic Trait”*

What Trait are we creating:

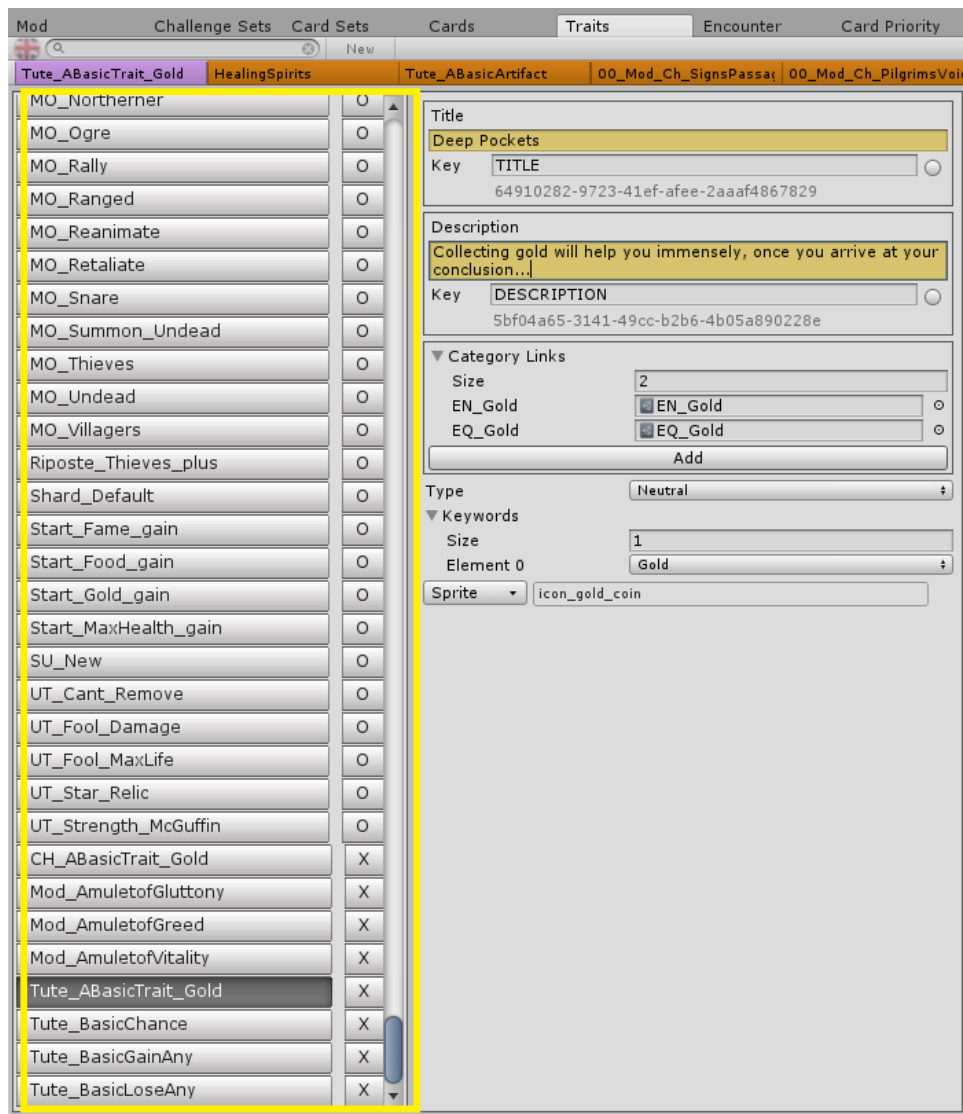
In this tutorial you’ll be creating the trait used in the Basic Challenge tutorial. The objective for the Basic Challenge is to collect 100 gold before reaching the end of the challenge, doing so will increase your odds of success in the final fight. This trait will therefore highlight any Gold associated cards for the players, so that they know what to be deck-building towards and highlight cards that may help them complete the challenge objective.

## Creating our new trait:

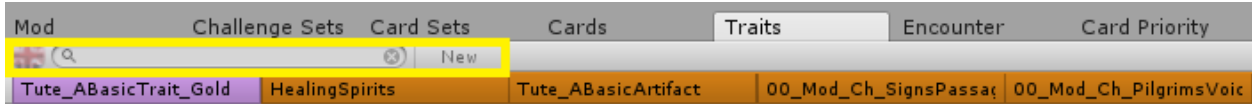
Creating Traits works a little differently to creating the cards we've created previously. Traits are not their own card, therefore they're not created under the **Cards** tab of the card editor. Instead you'll spot a tab named **Traits** along the top toolbar, so let's start by selecting this tab.



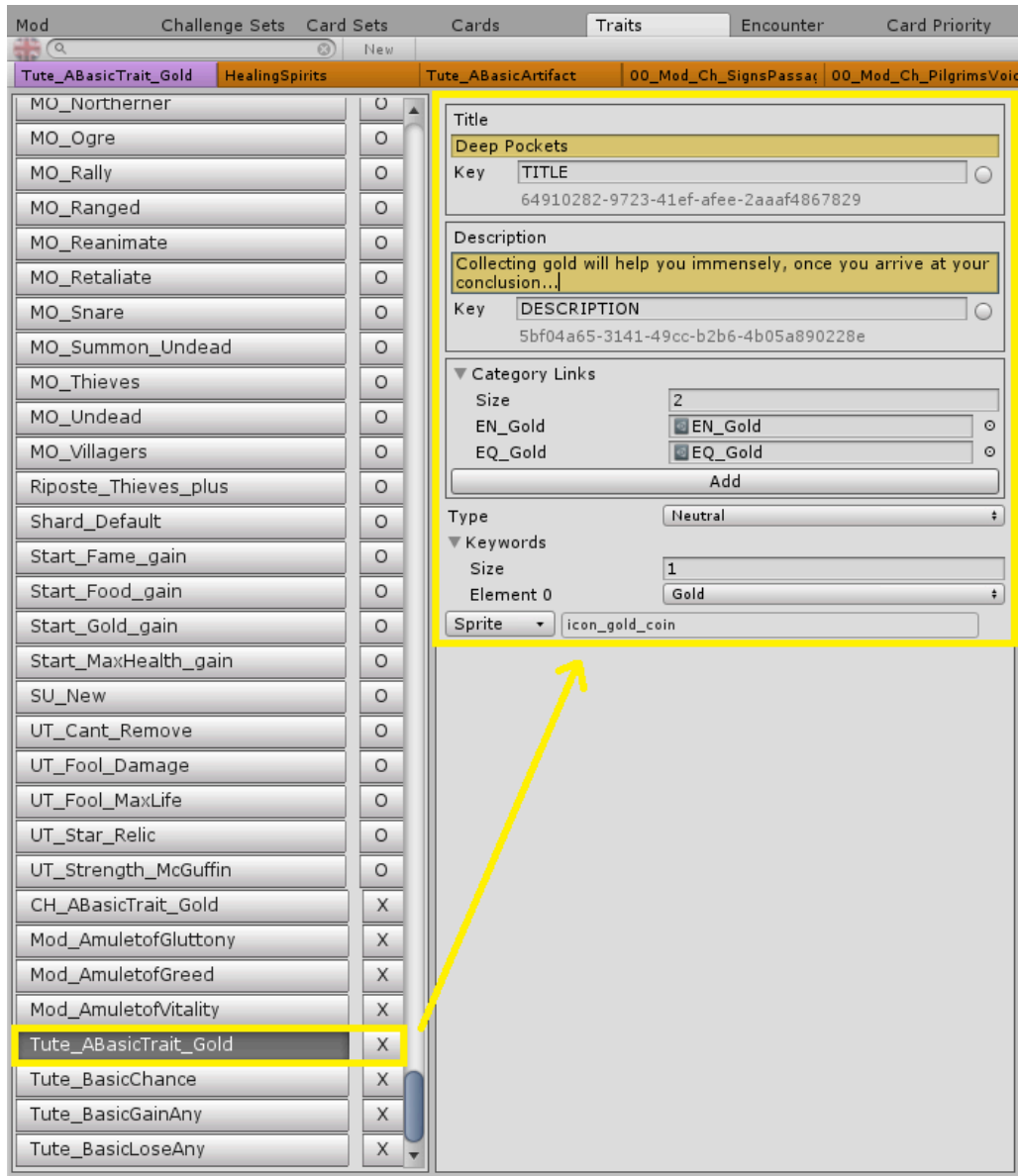
Selecting this tab opens a list to the left of the editor with all of the existing traits in HoF2. It will look like the example below.



To create our new trait we'll be entering our desired name into the search bar located at the top of this list. Let's name the Trait, "Tute\_ABasicTrait\_Gold", and click **New**. Our trait will now be in the list to the left.



Let's find our trait and view its data. This is demonstrated in the example below



At first look, the data for our trait will have some similar labels as we've seen previously, but there will also be some newer ones.

The image shows a screenshot of a trait configuration form. The form is divided into several sections, each highlighted with a yellow border. The sections are: 1. Title: A text input field containing "Deep Pockets". 2. Key: A dropdown menu set to "TITLE" with a unique ID "64910282-9723-41ef-afef-2aaaf4867829" below it. 3. Description: A text input field containing "Collecting gold will help you immensely, once you arrive at your conclusion...". 4. Key: A dropdown menu set to "DESCRIPTION" with a unique ID "5bf04a65-3141-49cc-b2b6-4b05a890228e" below it. 5. Category Links: A section with a "Size" input set to "2", and two checkboxes labeled "EN\_Gold" and "EQ\_Gold", both of which are checked. Below this is an "Add" button. 6. Type: A dropdown menu set to "Neutral". 7. Keywords: A section with a "Size" input set to "1" and an "Element 0" dropdown set to "Gold". 8. Sprite: A dropdown menu set to "icon\_gold\_coin".

The **Title** is the same as if we were entering the name for our new card. Let's call our Trait "*Deep Pockets*".

The **Description** is where we'll input the descriptive text of what our Trait means to the card it's attached to. This Trait will be attached to our Basic Challenge card, when we get to creating it. The challenge objective is to gain 100 gold so that you can bribe the highest amount of enemies out of the final combat, but we don't just want to flat out say that, let's add a little flavour and keep it a bit cryptic for players. Let's enter the text below as our Traits description.

**"Collecting Gold throughout the levels, will help you greatly in the challenges conclusion..."**

**Category Links** are specific links to different cards found in the game. For example, our Trait should be linked to cards that have any relation to Gold, as that's the nature of our Trait. Adding these links allows us to identify cards that are associated with Gold, and then highlight them to players when they're in the deck building phase of the game. Let's increase the **Size** to be **2** and **Add** the **Category Links**, **EN\_Gold** & **EQ\_Gold**. This is demonstrated in the example above.

**Type** is the type of Trait that we're creating. We're not creating a **Positive** or **Negative** Trait, so let's just set ours to **Neutral**.

**Keywords** are tags for the game system, when it's looking for cards. These are set so that the game knows when it's found a related card based off of the keywords set. Our keywords will be Gold because that's the only type of card we want the game to be looking for.

**Sprite** is where we set the icon for our trait. This is attached to the card and is viewable when the card with this trait attached, is inspected in game. This icon should be linked to the nature of your trait and its use. Let's set our sprite to be **icon\_gold\_coin**.



## Section 5: “The Basic Challenge” (Tutorial)

What is explored in this section of the documentation?

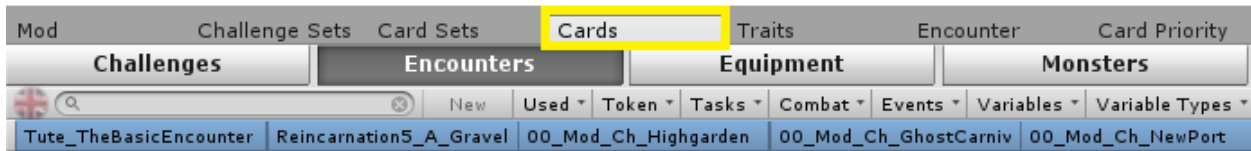
- [Creating the new challenge card:](#)
- [Challenge Card Data:](#)
- [Changing the name and description of the challenge card:](#)
- [Assigning an art texture to the challenge card:](#)
- [Assigning our trait to the challenge card:](#)
- [Creating the Gold challenge token:](#)
- [Creating the challenges Global BB:](#)
- [Creating the variables in the Global BB:](#)
- [Creating the challenge objective:](#)
- [Incrementing the ‘Objective Channel Index’ & setting the ‘Objective State’ on the Entrance Card:](#)
- [Setting up the behaviour tree for the End Card so that it’s compatible with your basic challenge mod:](#)
- [Creating challenge decks and card sets:](#)
- [What is a dungeon:](#)
- [Creating the dungeon for the challenge:](#)
- [Dungeon Data:](#)
- [Assigning the config data for the dungeon:](#)
- [Assigning the level exit card for the challenge:](#)
- [Assigning the dungeon end card for the challenge:](#)
- [Assigning the dungeon map layout:](#)
- [Afterthoughts and revision:](#)

What is the challenge we’re creating:

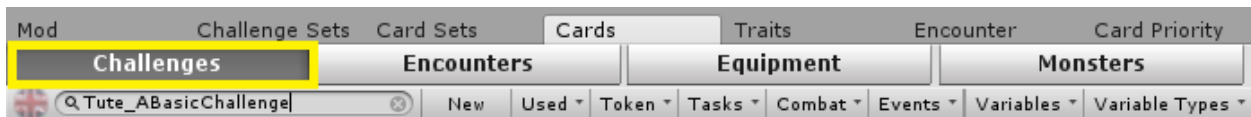
In this tutorial you’ll be creating the basic challenge. The basic challenge will feature as the main piece of modded content that you’ll be creating as a result of following this documentation. All of these individual pieces that you’ve created so far, will be used in creating this challenge mod. The challenge objective has players attempting to collect 100 gold before they reach the bad boss dude at the end, gold will significantly better the players chances in the boss fight as they can use the gold they’ve collected, to bribe off enemies that would otherwise be in this final combat.

## Creating the new challenge card:

In order to start creating the *"Tute\_TheBasicChallenge"* you'll need create the challenge card. Navigating across the top toolbar of the card editor, you'll spot a tab named **Cards**. Click the **Cards** tab, doing so will display a new toolbar directly underneath this one.



Navigating across the new toolbar underneath, you'll spot a tab named **Challenges**. Click the **Challenges** tab, doing so will display a list on the left of the editor. This list contains all of the existing encounter cards.



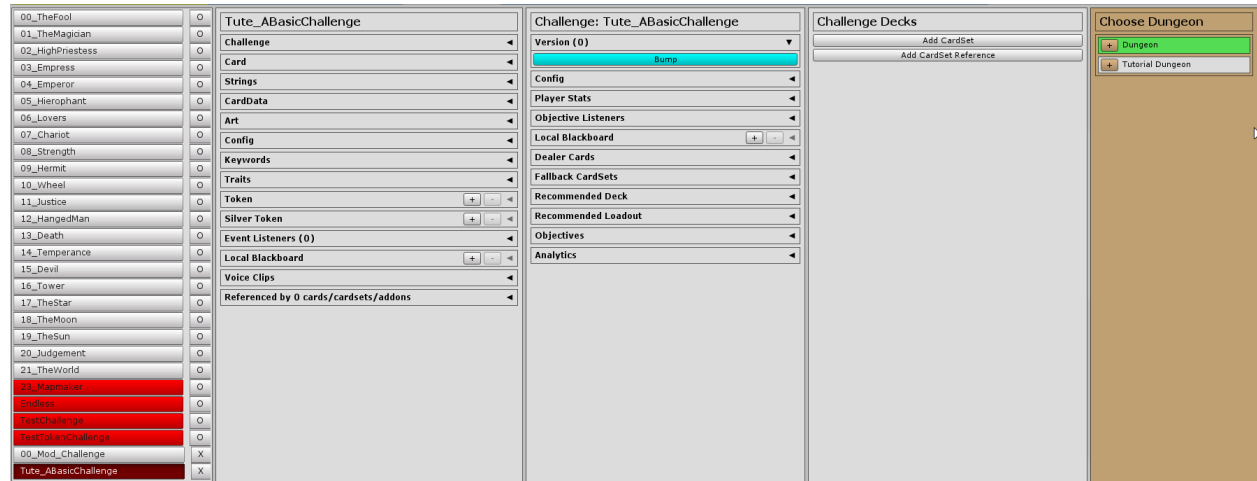
All of the challenge cards within this list are viewable. In order to view a challenge cards data, select the card that you wish to view, this will open that card where you can now view its data. The ability to view existing challenge cards allows modders to see the content of existing challenges in HoF2. At the top of this list of existing challenges you will spot a search bar with a button next to it labelled **New**.



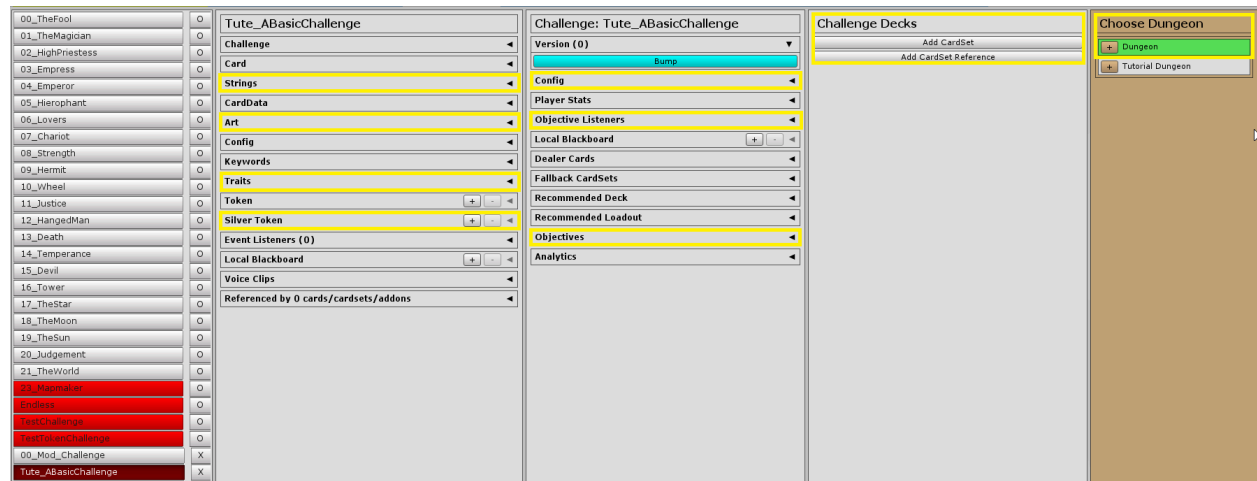
Click inside of the search bar and type the name *"Tute\_ABasicChallenge"* and click **New**, doing this will create the new challenge card and add it to the list on the left.

# Challenge Card Data:

Now that we've created the new challenge card "Tute\_ABasicChallenge", we can start to unpack the data that builds a challenge in HoF2. Firstly let's select our challenge card that we've just created. After its opened, the contents will look like the example below.



Now let's begin identifying the data that you'll need to set up when creating the basic challenge. Highlighted in yellow are the things that you'll need to do, don't worry we'll explain it all to you throughout this tutorial.

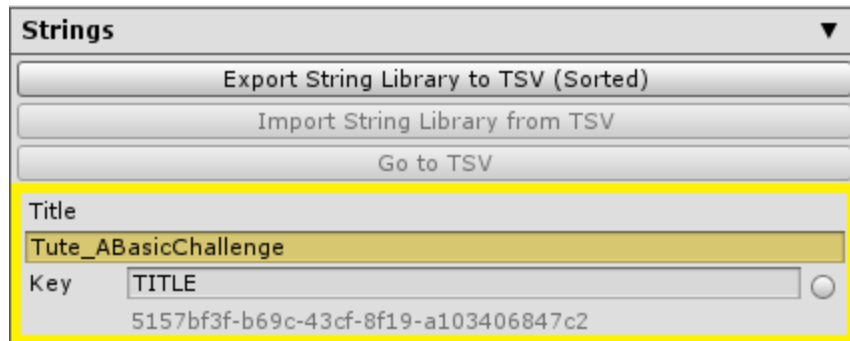


The following areas are the ones that you'll be focusing on throughout this tutorial:

- *Strings*
- *Art*
- *Traits*
- *Silver Token*
- *Challenge Config*
- *Objectives*
- *Challenge Decks*
- *Choose Dungeon*

## Changing the name and description of the challenge card:

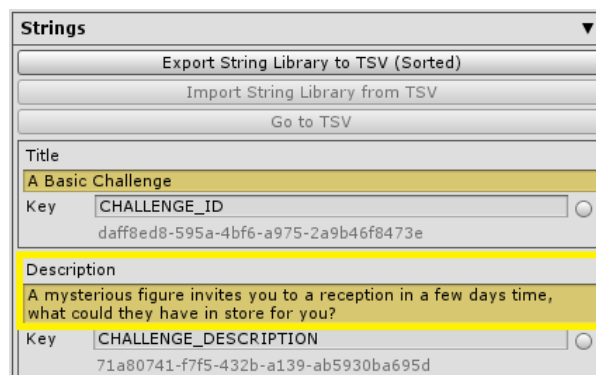
Let's open the **Strings** label of the card, we don't want our challenge card (when it's viewed in HoF2) to read "*Tute\_ABasicChallenge*", so let's change the title string to "*A Basic Challenge*". Alternatively you can name this challenge to whatever you'd like it be.



**Note:** We've changed the title string of the challenge card to "*A Basic Challenge*", the challenge card when searched for in the list of created cards is still named "*Tute\_ABasicChallenge*". You can change this by selecting the small union jack in the left corner of the card editor.

Let's set the **Description** for our challenge card. This is demonstrated below. Let's enter the following description text:

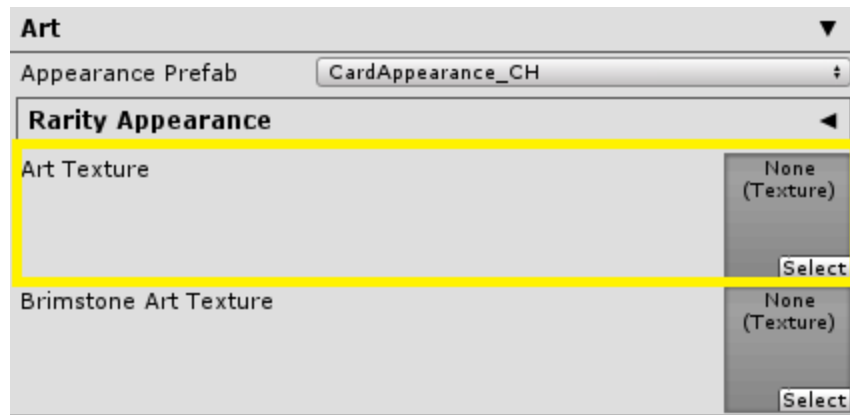
**A mysterious figure invites you to a reception in a few days time, what could they have in store for you?**



**Note:** You may have to click the circle next to the **Key** to enter text for your **Description**, this will open the **StringPicker** and you can assign your strings in there.

## Assigning an art texture to the challenge card:

Next let's open the **Art** label and with this we'll have some new information to work with. You'll notice two selection boxes under this label, one is named **Art Texture** and the other **Brimstone Art Texture**, for this challenge we'll only be using **Art Texture**.

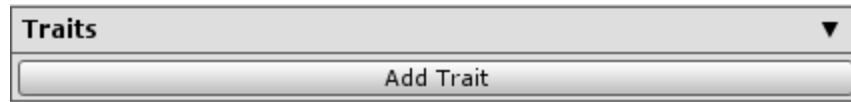


After clicking the selection box next to **Art Texture**, a new window will open with a lot of art options for our encounter card. Let's search for the texture **c\_challenge\_world** and assign this as our **Art Texture** for the challenge card.

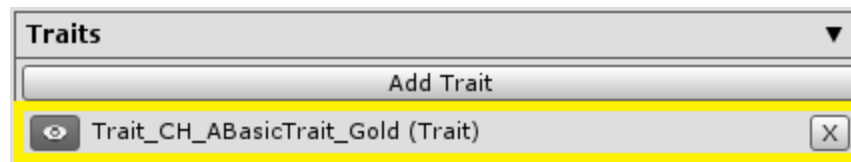
After changing both the title string and the art texture for your new challenge card, we can now move onto adding the *"Tute\_ABasicTrait\_Gold"* that we created earlier in this documentation.

## Assigning our trait to the challenge card:

Next let's open the "Traits" label and see what we can do.



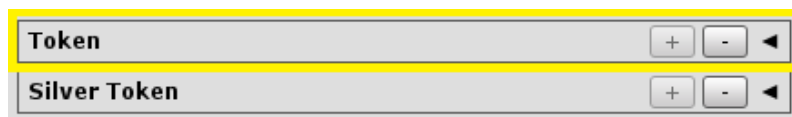
After opening the "Traits" label let's proceed by adding the "Tute\_ABasicTrait" to the challenge card. If you click the button labelled "Add Trait" and search for the trait "Trait\_Tute\_ABasicTrait\_Gold" you can assign it to the challenge card. The challenge card now had the trait we made for it.



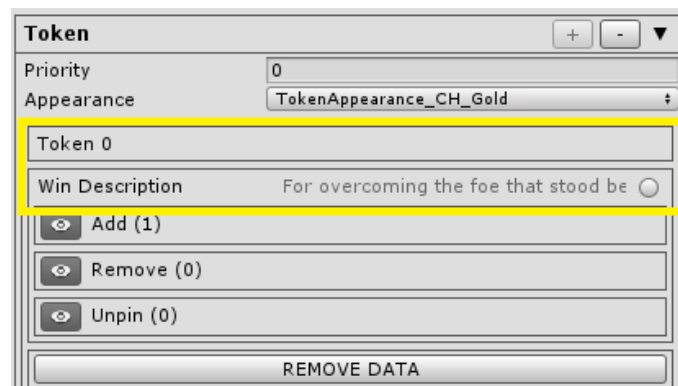
## Creating the gold challenge token:

Challenges have two different types of tokens attached to them, these are the Silver and Gold tokens. These are the rewards for completing the challenge, and the objectives of a challenge. Gold tokens are typically harder to achieve and have more steps, whereas silver tokens are just the default reward for completing a challenge. Challenges need a Gold token, but they do not need a Silver one, unless you'd like your challenges to have one.

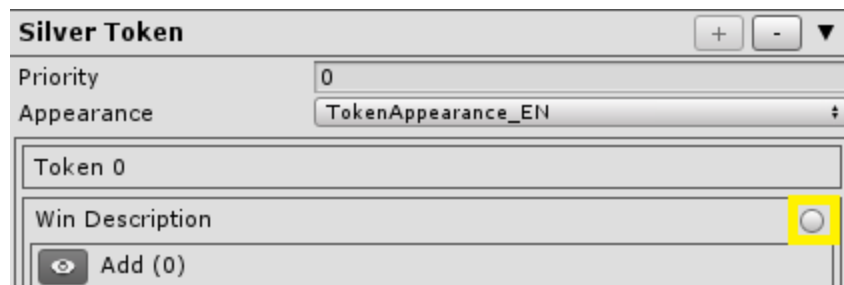
Let's look at setting up the Gold token for the challenge. Challenges always need a Gold token, but do not always need a Silver token. Click the small addition symbol next to the **Token** label, to create the Gold token for the challenge.



Let's look at the data that you'll need to worry about. Highlighted in yellow (below) are the **Win Description** label, and the **Add** expandable list.



Let's start with setting the **Win Description** label to the Gold token, for when it is unlocked by our players. Clicking on the small circle dot next to the **Win Description** label will open the **StringPicker**.





After the **StringPicker** has opened, we'll add some text for when the Gold token is unlocked at the end of the challenge. We'll be entering the text into the section highlighted below.

Challenge_TUTE_ABASICCHALLENGE		
->	SILVER_TOKEN_UNLOCK	For completing the basic challenge...
->	CHALLENGE_ID	A Basic Challenge
->	CHALLENGE_DESCRIPTION	A mysterious figure invites you to a reception in a few days time, v
->	GOLD_TOKEN_UNLOCK	For overcoming the foe that stood before you...

Below you can find the text to enter for the content fields:

---

<b>GOLD_TOKEN_UNLOCK</b>
For overcoming the foe that stood before you...

---

After you've entered this text, click the button to the left of this input field. What this will do, is it will assign the string we just entered as the tokens **Win Description**.

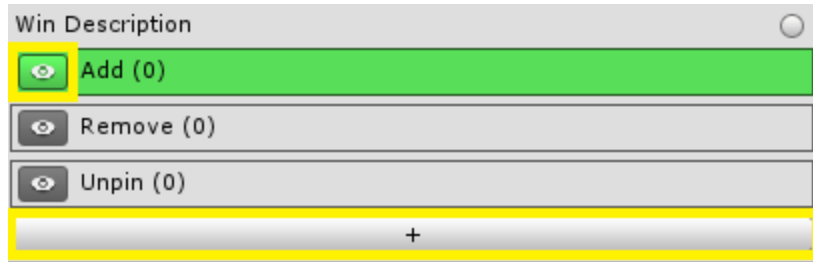
Challenge_TUTE_ABASICCHALLENGE		
->	SILVER_TOKEN_UNLOCK	For completing the basic challenge...
->	CHALLENGE_ID	A Basic Challenge
->	CHALLENGE_DESCRIPTION	A mysterious figure invites you to a reception in a few days time, v
->	GOLD_TOKEN_UNLOCK	For overcoming the foe that stood before you...

After assigning the string to the **Win Description**, your Gold token component should be like the screen capture below.

Token 0
Win Description      For overcoming the foe that stood be <input type="radio"/>

Let's now look at the expandable section under **Win Description** labelled **Add**. This is a list where we can add the cards that we want to be unlocked and added to the players deck, when the Gold token is won.

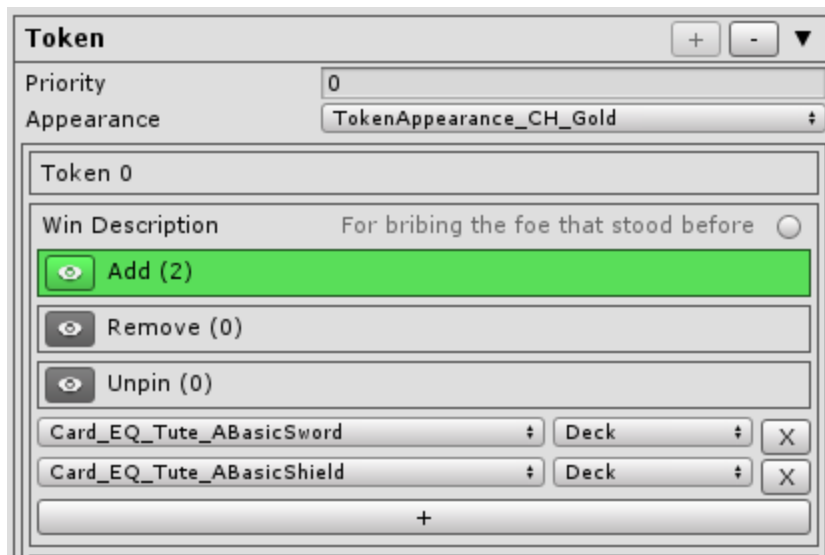
By clicking the eye symbol to the left of **Add**, this data will turn the colour green and a new addition button will appear at the bottom of this data. This has been highlighted for you in the below screen capture.



Let's click the addition button and search for the equipment cards we created earlier in this documentation. These were:

- "Card\_EQ\_Tute\_ABasicSword"
- "Card\_EQ\_Tute\_ABasicShield"

Now, let's add them to the list of cards to be unlocked at the end of the challenge, when the Gold token is won. The screen capture below demonstrates what this section should look like.



Well done, you've just set up the Gold challenge token with appropriate cards to be unlocked and added to the players deck when the token is won.

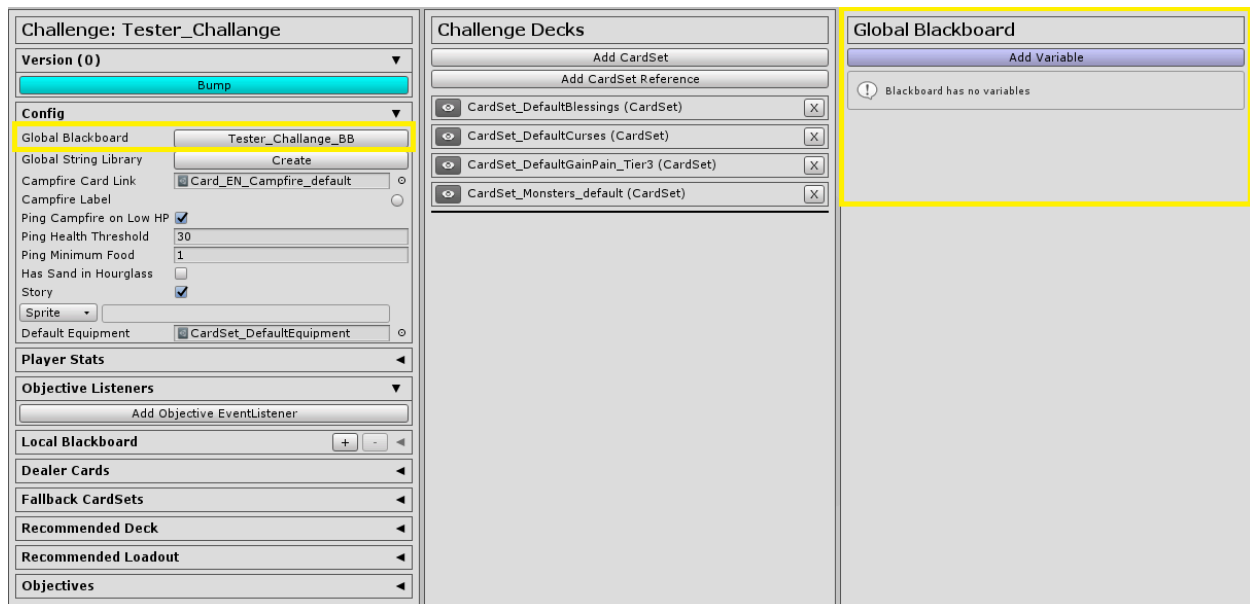
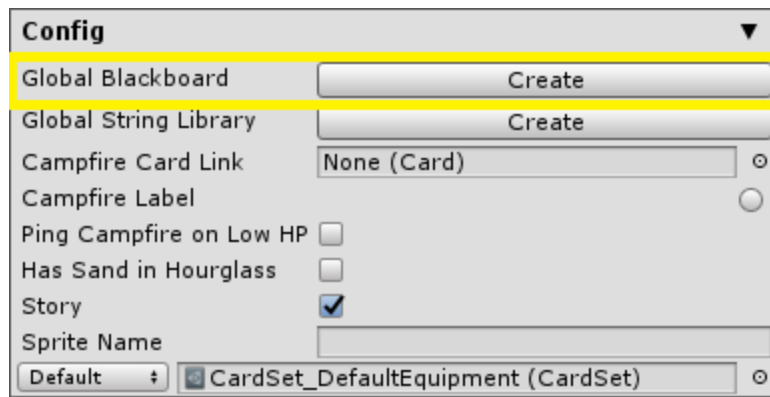
Lastly, you may have spotted a piece of data named **Appearance**. This sets the appearance type for our token, by default this is set to **Token\_Appearance\_EN** meaning **Encounter**, but this is a challenge token, a Gold one at that. Let's change this tokens **Appearance** to be **Token\_Appearance\_CH\_Gold**. This is demonstrated for you, below.

Token		+	-	▼
Priority	0			
Appearance	TokenAppearance_CH_Gold			+

Next we'll look at setting up the **Config Data** for the challenge card, specifically looking at creating the challenges Global Blackboard (BB), followed by how we can create variables for the challenge.

## Creating the challenges Global BB:

One of the things that we have to do is create the **Global BB** for the challenge. You'll see a button labelled **Create** next to the label Global Blackboard, let's click this and a new set of data will appear on the challenge card, this new data can be found next to the **Challenge Decks**. Demonstrated below.

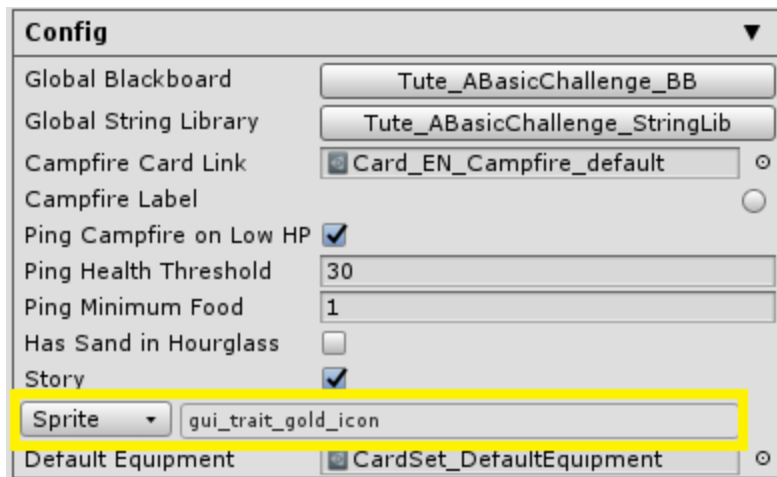


The variables created in the Global BB of a challenge card, are openly accessible within any encounter behaviour tree. So if you're making challenge specific encounters that affect the challenge objective, these are where you'd create and store your variables so that everything was talking to each other. This is how they differ from variables made in a Local BB.

While we're still in the **Config Data** for the challenge card, let's look at setting an appropriate **Sprite** for the challenge marker that is displayed on the challenge map. This is demonstrated below:



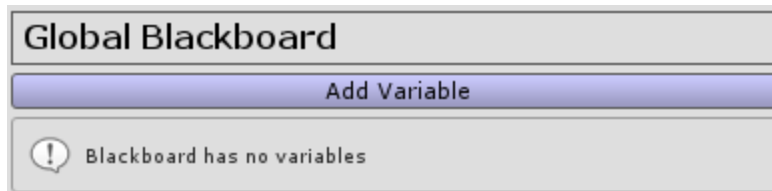
You'll spot an expandable tab within **Config Data** labelled **Sprite**. This is demonstrated below:



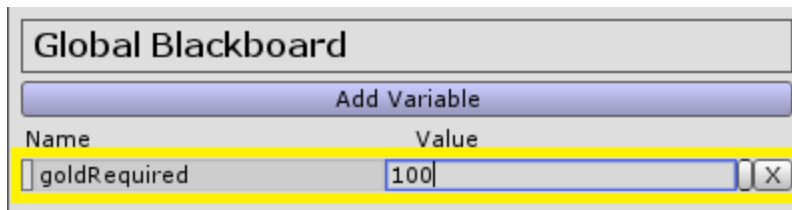
Within this, you can select a sprite for your challenge marker when it is shown on the challenge map. Let's select the *"gui\_trait\_gold\_icon"* as the sprite for our challenge marker.

## Creating the variables in the Global BB:

One thing that you'll notice in the **Global BB** is the big purple button labelled **Add Variable**, let's click this and create an **Integer**.



After creating your integer let's rename the variable to *goldRequired*. Now let's set the value of the variable to be 100.



---

**Note:** *goldRequired* is going to be the variable referenced in our challenge objective, which we'll be creating next.

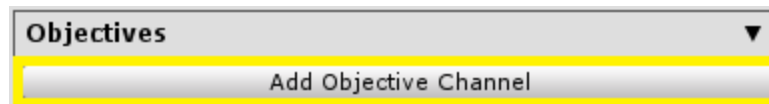
---

## Creating the challenge objective:

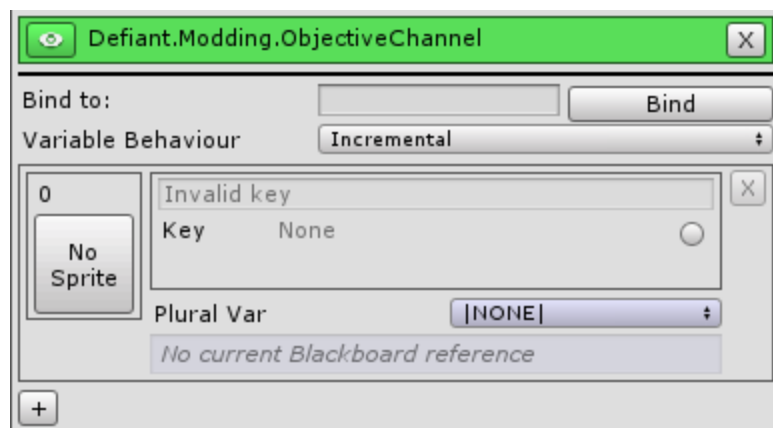
When creating challenge objectives, what you're actually setting up, is the UI that displays in the top right of the screen during gameplay, that keeps track of the challenge progress and reminds players what they're trying to achieve during a challenge.

The actions that you do here are only related to setting up this UI, and not setting up the tracking of things (although in our example that is semi automatic as it's tracking player gold).

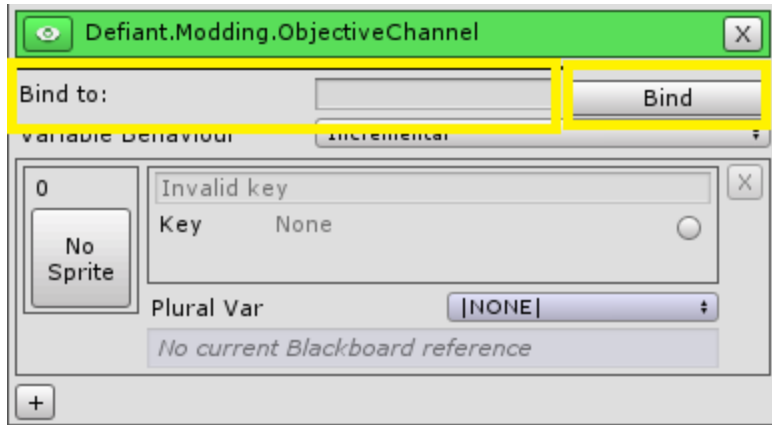
Now, let's look at creating the challenge objective that's going to be in our challenge. After expanding the component labelled **Objectives**, you'll have a new button appear named **Add Objective Channel**, let's click that and create a new channel, now.



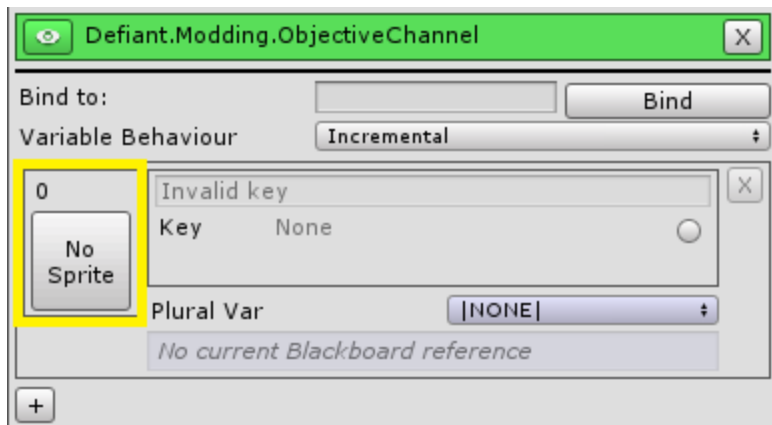
After creating the new objective channel, let's expand its contents and rename it to something easier for us to identify.



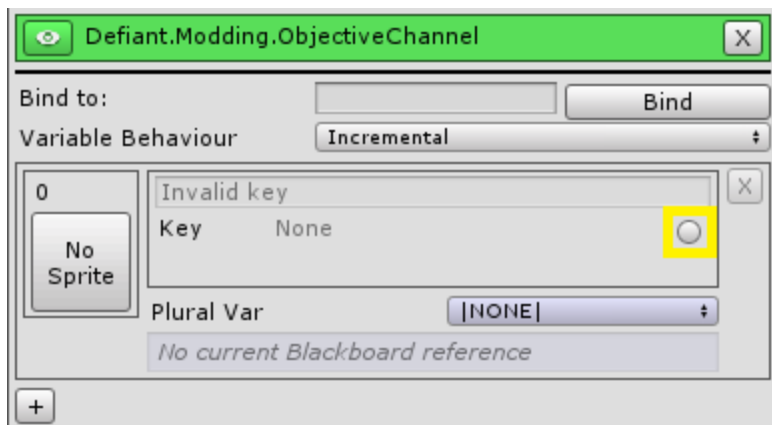
In order to rename the objective channel, click inside of the text field next to the name **Bind to**, let's rename this objective channel "*Obj Gold*". Now click the button **Bind**, and the name of the objective channel will be changed.



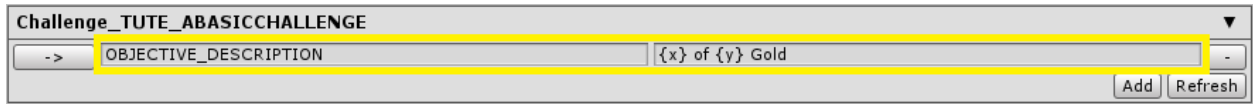
Next let's click the button labelled **No Sprite** and assign an appropriate sprite to accompany our challenge objective, search for "icon\_gold\_coin" and assign it as the sprite.



After we've assigned the sprite to our challenge objective, let's look at adding the objective itself. Let's start by clicking on the small circle dot next to the **Key** label, this will open the **StringPicker**.







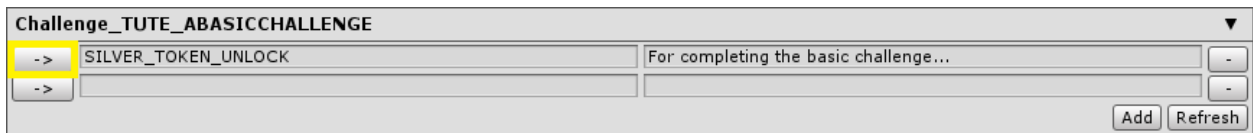
Below you can find the text to enter for the content fields pictured above.

---

**OBJECTIVE\_DESCRIPTION**  
{x} of {y} Gold

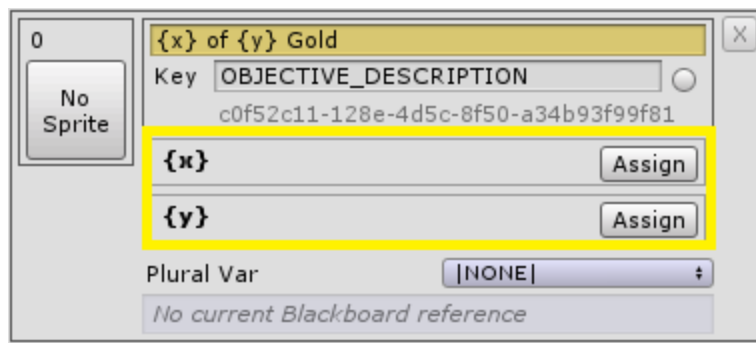
---

After you've entered the text, click the button to the left of the input field. What this will do, is it will assign the string to the **Objective**.

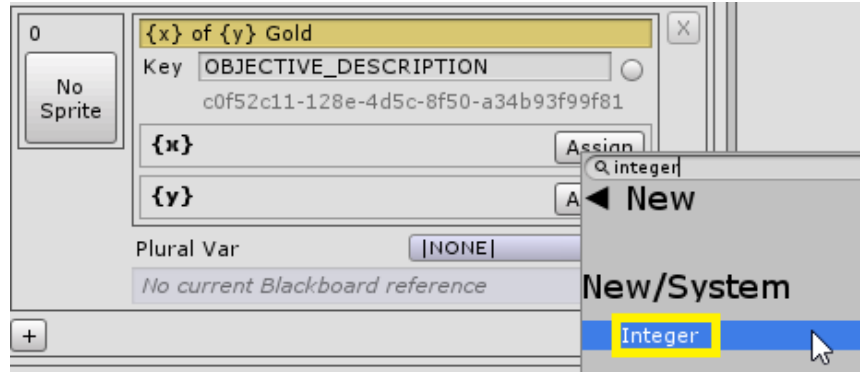


Now, after assigning the string to the **Objective**, we'll have some new information that we can work with, where we can start to implement the logic for this objective.

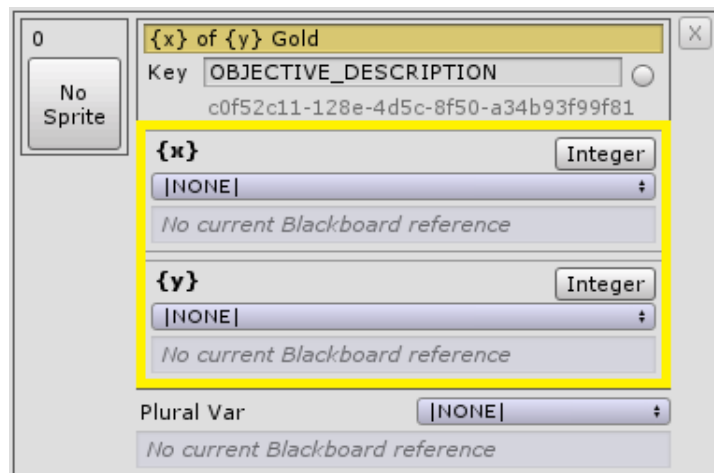
When you return to the challenge card, you will now have two new fields that require you to assign variables to them, these will be labelled "{x}" and "{y}".



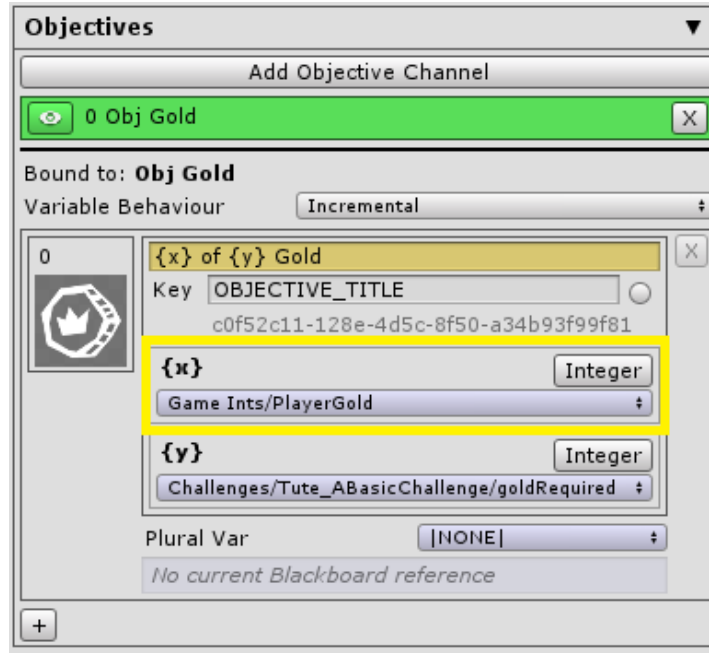
Click the button labelled **Assign** next to "{x}" and select the integer variable. After you've done this, do the same for "{y}".



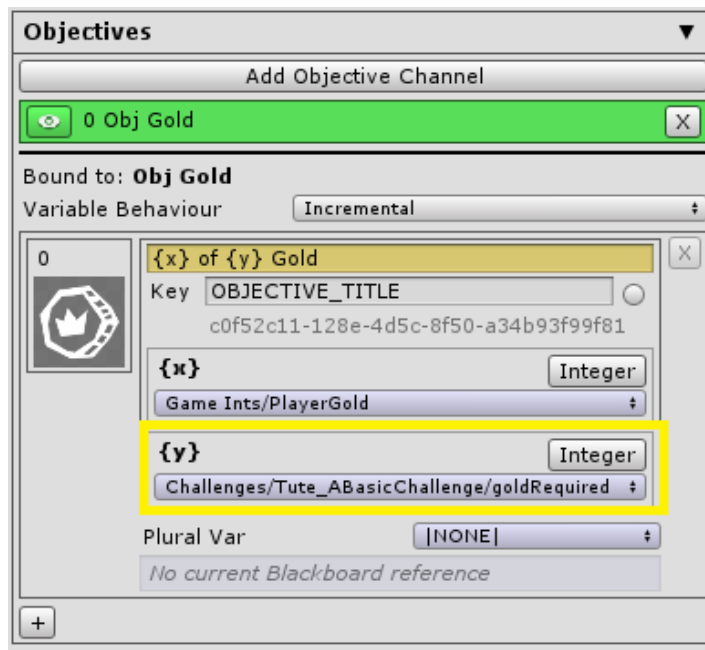
After you've assigned both "{x}" and "{y}" to be integer variables, they should look like the screen capture below, where it now asks you for a Blackboard reference.



So, remember back to when we created that "goldRequired" variable in our challenges Global BB? Well, now is the time to use it. Let's click on the first drop down box under the name "{x}" and search for "PlayerGold", now assign it.



Let's now click on the second drop down box under the name "{y}" and search for "goldRequired", now assign it.



We've now just created our challenge objective for gaining an amount of 100 gold throughout the challenge, in order to be able to bribe off enemies that you'll face in the final combat of the challenge.

## Incrementing the 'Objective Channel Index' & setting the 'Objective State' on the Entrance Card:

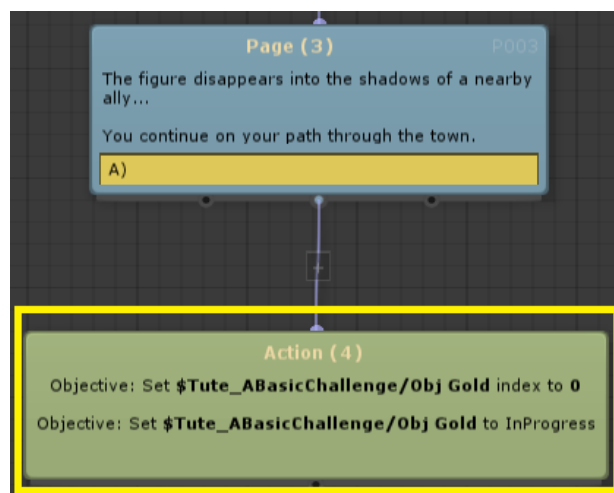
Now, this is where things start to get a little more intermediate. We'll now need to jump into the **Entrance Card** for our Basic Challenge, this card is named "*Tute\_ABasicChallenge\_Entr*", and what we need to do is set what is called the '**Objective Channel Index**' followed by setting the '**Objective State**' to '**In Progress**'.

The first thing you **have** to do is make these two encounters editable to you. You'll spot the '**O**' button next to the existing encounter cards, this is shown in the example below:

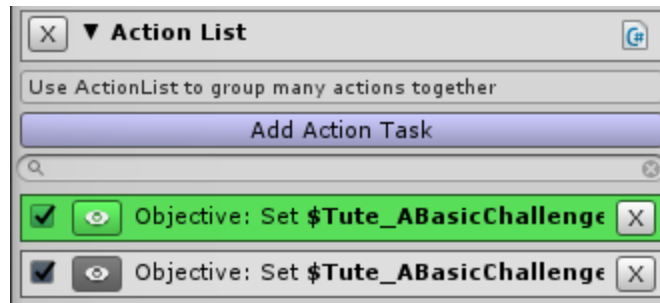
CorruptedCity	<input type="radio"/>
TradingHouse	<input type="radio"/>
Tute_ABasicChallenge_End	<input type="radio"/>
Tute_ABasicChallenge_Entr	<input type="radio"/>
Tutorial_Handedness	<input type="radio"/>
WanderingTracker	<input type="radio"/>
WanderingTrackerMeetup	<input type="radio"/>

After clicking the '**O**' to make these editable, select the **Entrance Card** and click **Edit**. You'll now be able to edit this encounter behaviour tree.

After you've entered the encounter behaviour tree for the "*Tute\_ABasicChallenge\_Entr*" encounter card, let's jump down to the bottom of our node graph, **Page (3)** will be the ending node. Let's create a new action node, demonstrated below:



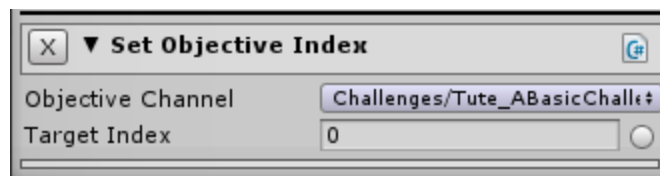
Select the node and add an **Action list** to it, demonstrated below:



You're going to click **Add Action Task** and search for **Set Objective Index**, now assign this action task. Let's now add another action task, but this time search for **Set Objective State**, now assign this action task too.

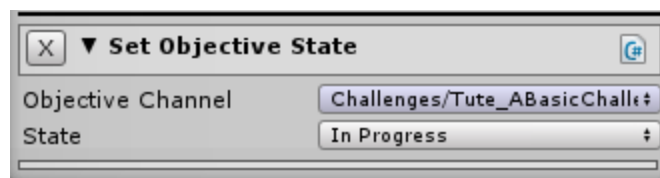
When we're looking at the **Set Objective Index** action task, we need to set the **Objective Channel**, and increment the **Target Index** from -1 (default) to 0. This means that the UI for our "Obj Gold" challenge objective will activate after the **Entrance Card** has been completed.

Let's set the **Objective Channel** to be our "Obj Gold" challenge objective. Next, let's set the **Target Index** to be 0. This is demonstrated in the below screen capture.

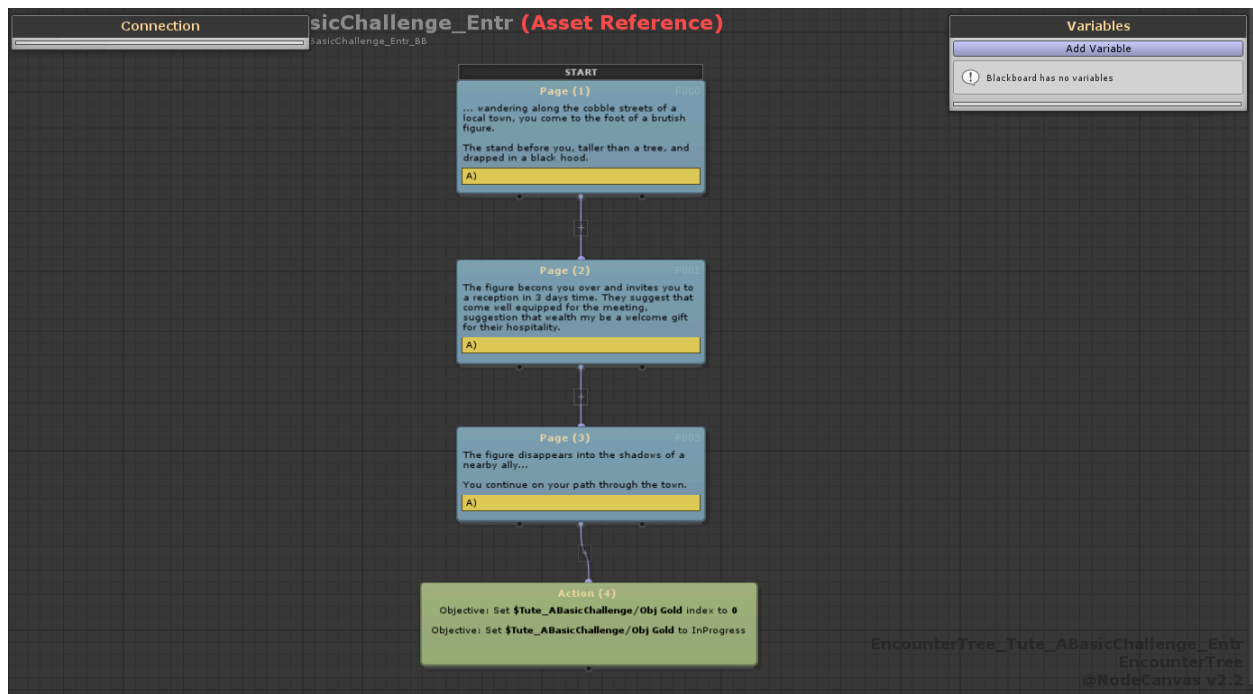


Now, let's jump to the **Set Objective State** action task and set this one up. Again this needs us to assign the **Objective Channel**, let's set this to be our "Obj Gold" challenge objective.

Next, we have to set the **State** of the challenge objective. Let's set this to **In Progress**, because after the **Entrance Card** has been completed, we want the challenge objective "Obj Gold" to be in progress. This is demonstrated in the below screen capture.



After you've set both of these action tasks up, your **Entrance Card** encounter behaviour tree should look like the screen capture demonstrated below:



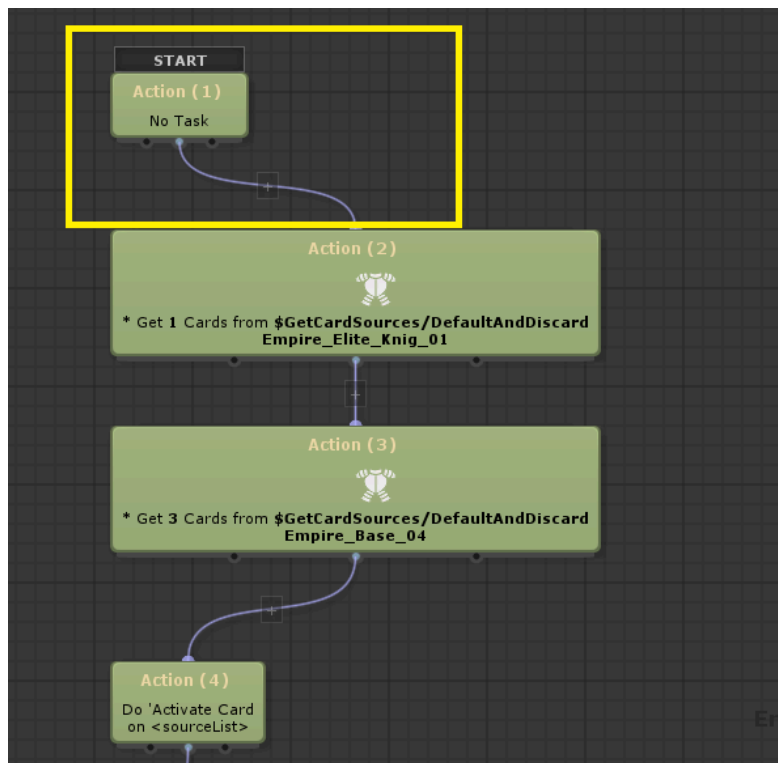
So that you can better understand what we've just had you do, here is this process demonstrated in the game.

## Setting up the data, for the ‘End’ behavior tree, so that it’s compatible with our basic challenge mod:

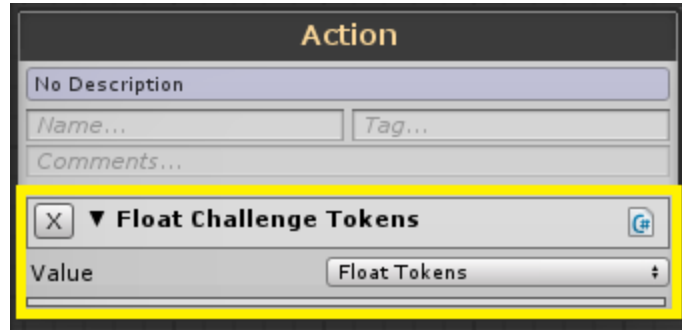
We’ll now need to jump into the **End Card** for our Basic Challenge, this card is named “*Tute\_ABasicChallenge\_End*”. We need to do the following things here:

- Set up a **Condition Check** for our “goldRequired” variable, so that the game knows if the Gold token unlock requirements have been met.
- Set up the action so that the challenge tokens float once players have entered this **End Card** encounter.
- Setting up **Challenge Status** so that the challenge actually ends and the appropriate tokens are awarded to players.

First let’s set up the easy stuff! Making the challenge tokens float is rather straightforward. Let’s start by creating a new action node at the top of the existing encounter behavior tree. Let’s assign this new node to be our **Start**. Demonstrated below:

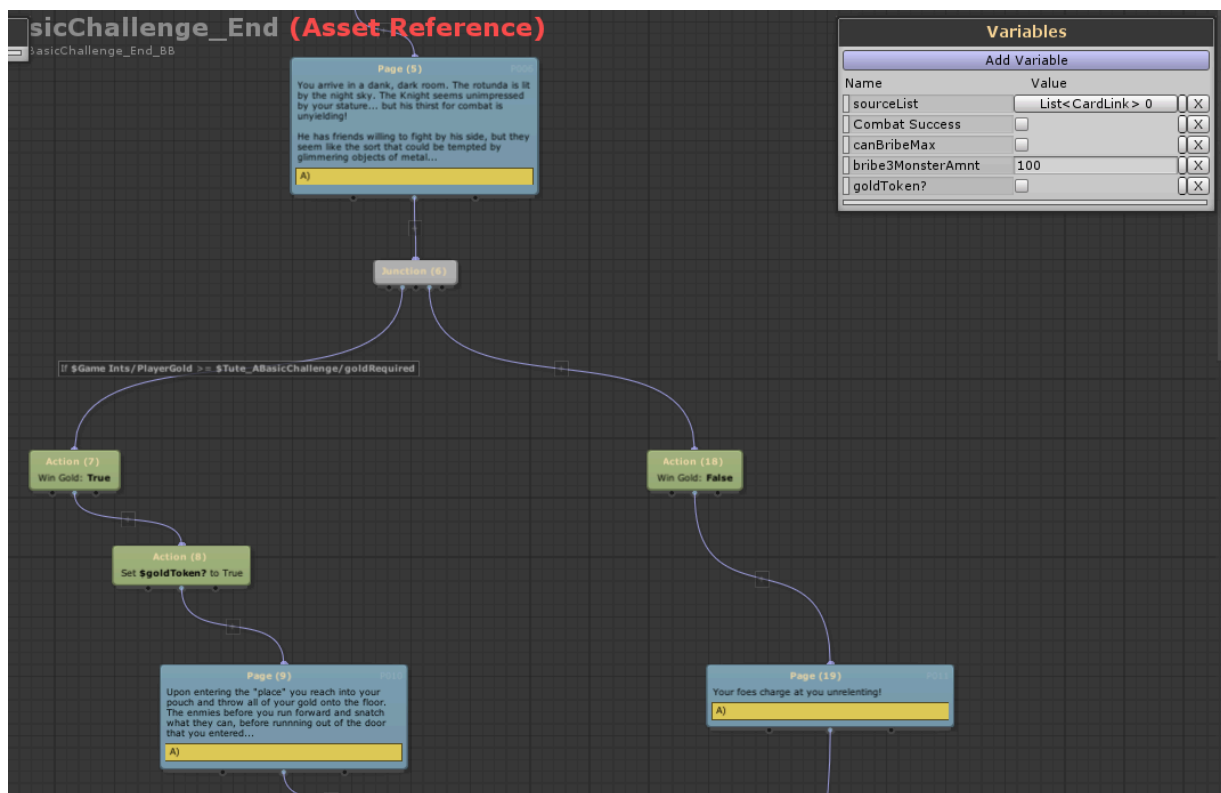


Let's select our new node and search for the action task **Float Challenge Tokens**, now assign this action task and you're done. Make sure that the **Value** is set to **Float Tokens** and not **Land Tokens**.



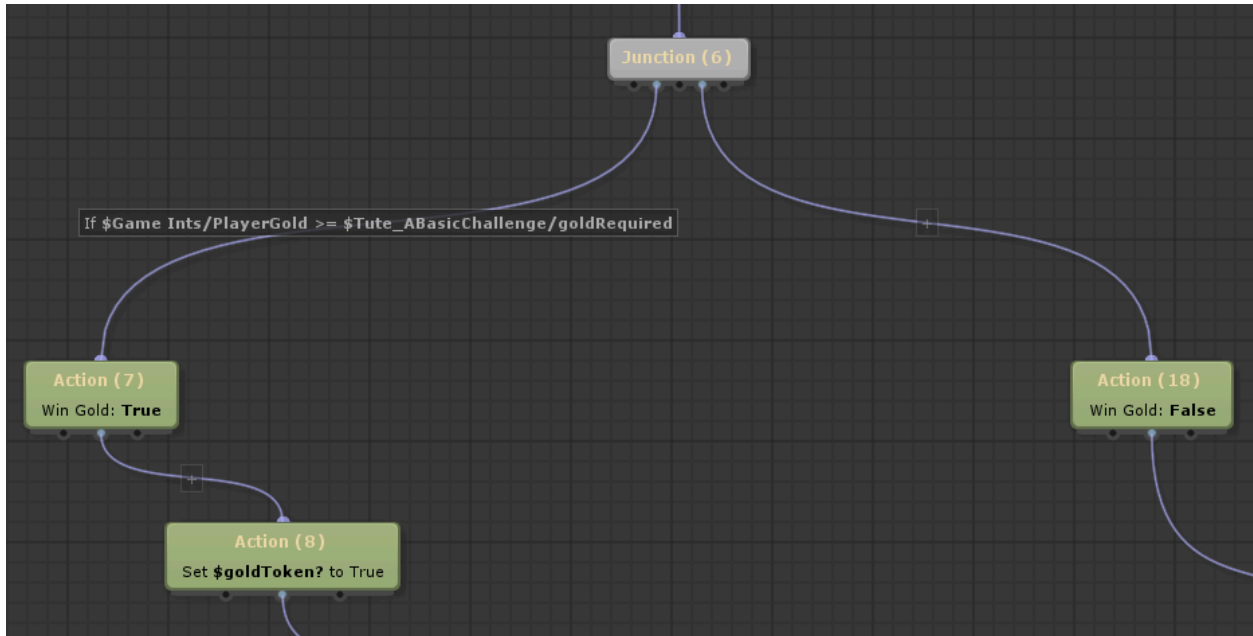
Next, let's look at setting up the **Condition Check** for the *“goldRequired”* related challenge objective. Essentially, players need to have collected 100 gold before entering the **End Card** encounter in order to unlock the Gold token for the challenge, and subsequently bribe off 3 of the possible enemies in the combat.

Let's look at what our encounter behavior tree will look like, once we've set this condition check, and then we'll look at breaking it down into steps.

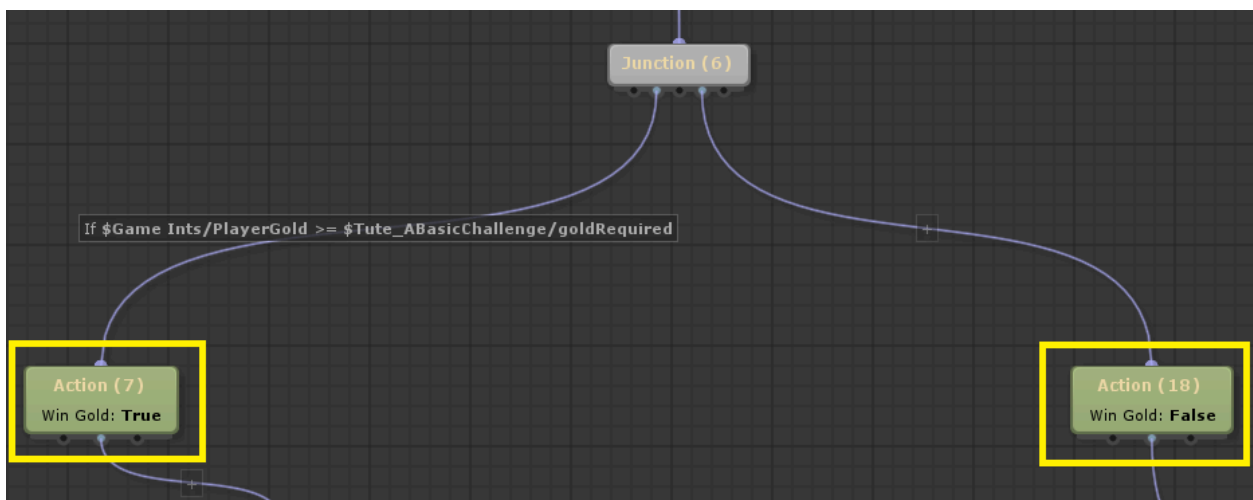




What we're going to be creating is the branch you can see in the above screen capture. To do this let's start by creating a **Junction** so that we can create a left and right branch to hold our **Win Gold True** and our **Win Gold False**.



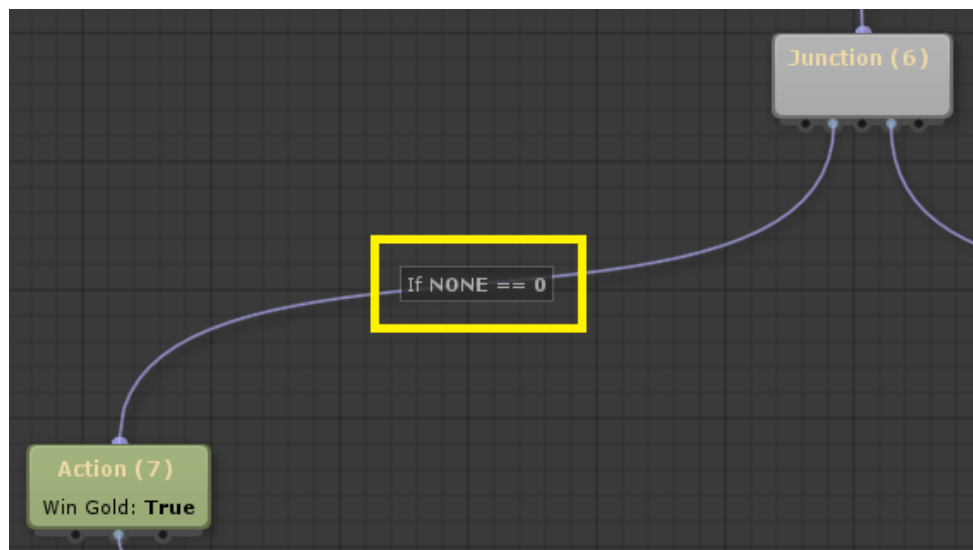
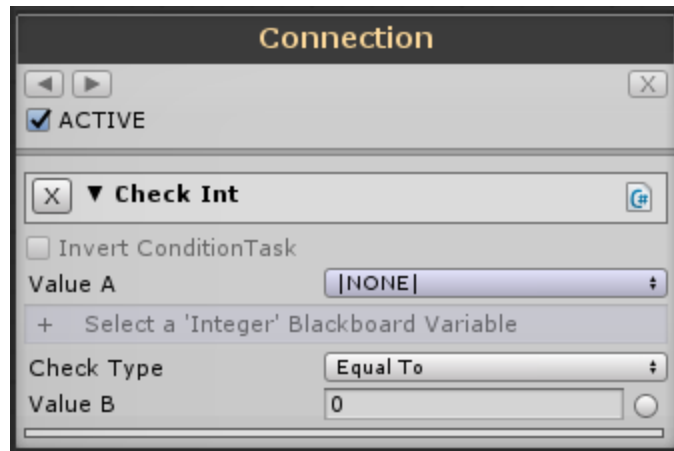
Let's start by creating the two action nodes you see highlighted below in the provided screen capture. These are our **Win Gold True** and our **Win Gold False**.



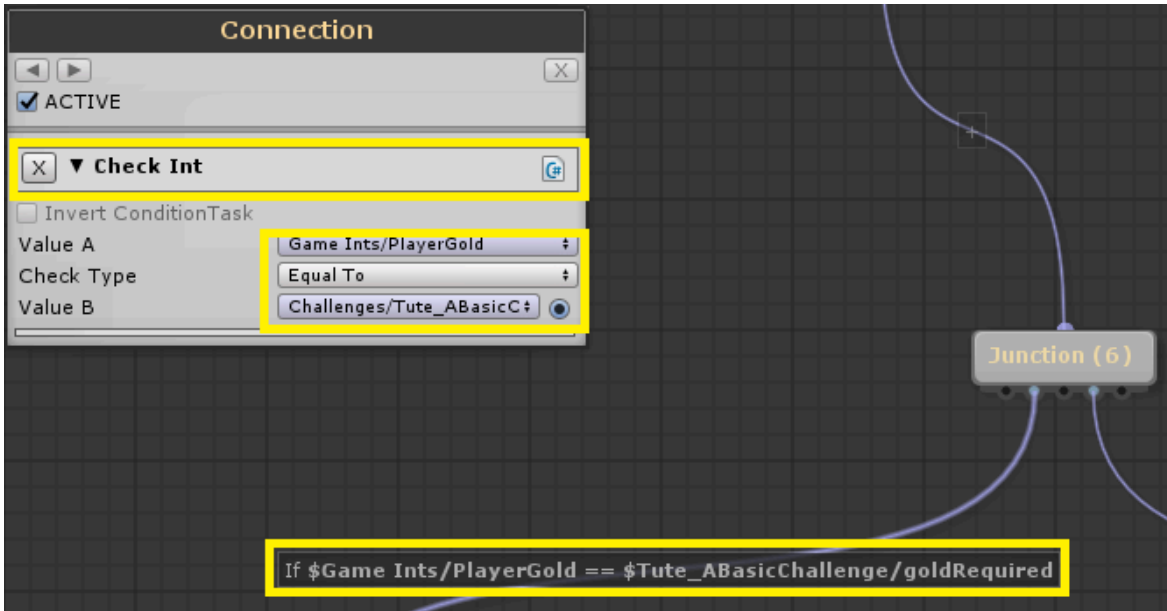
Search for the action tasks **Set Gold Challenge Token** and assign both of these action nodes with that action task. The node connected to the left branch should be **Win Gold True** and the right branches node should be **Win Gold False**.

Next we have to set the condition check for the left branch so that the game knows to award the players with the **Gold** challenge token if the “*goldRequired*” Global BB variable is greater than or equal to the current count for the players gold.

To do this, you'll spot a small addition symbol in the middle of the left branch, click this and we can add a condition task, **Check Int**. After assigning this condition task to the left branch, we can set its data.

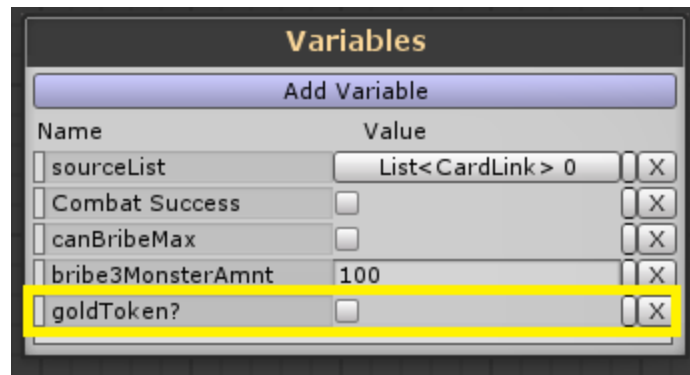


**Check Int** takes two **Values**. Let's set **Value A** to be **Game Ints > PlayerGold**. Then set the **Check Type** to be **Greater than or equal to**. Next, set **Value B** (the value that it checks against) to be **Challenges > Tute\_ABasicChallenge > goldRequired**. This is demonstrated below:

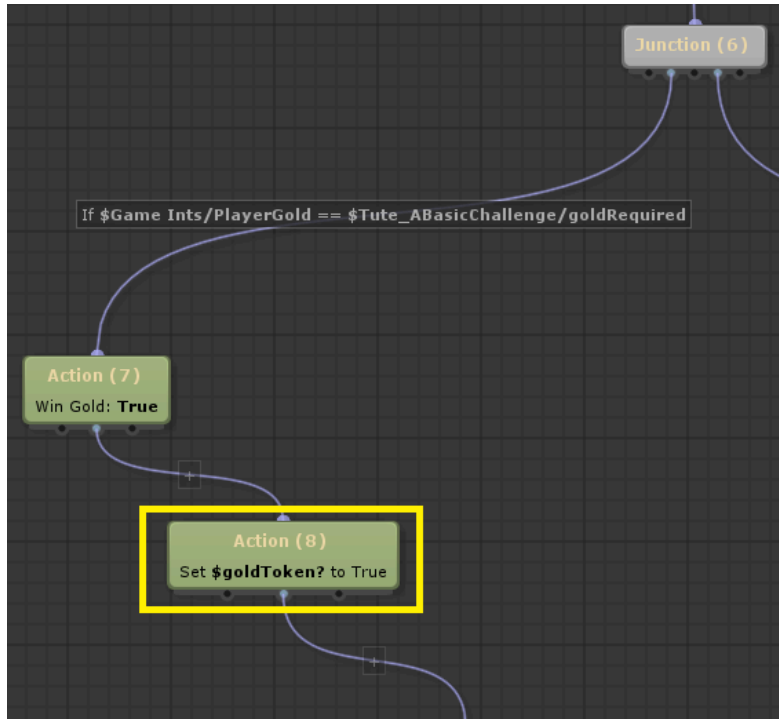


We've now said that if the players gold is greater than or equal to the amount of gold required by the challenge objective, then award players with the Gold token. Otherwise, the players should not receive the Gold token.

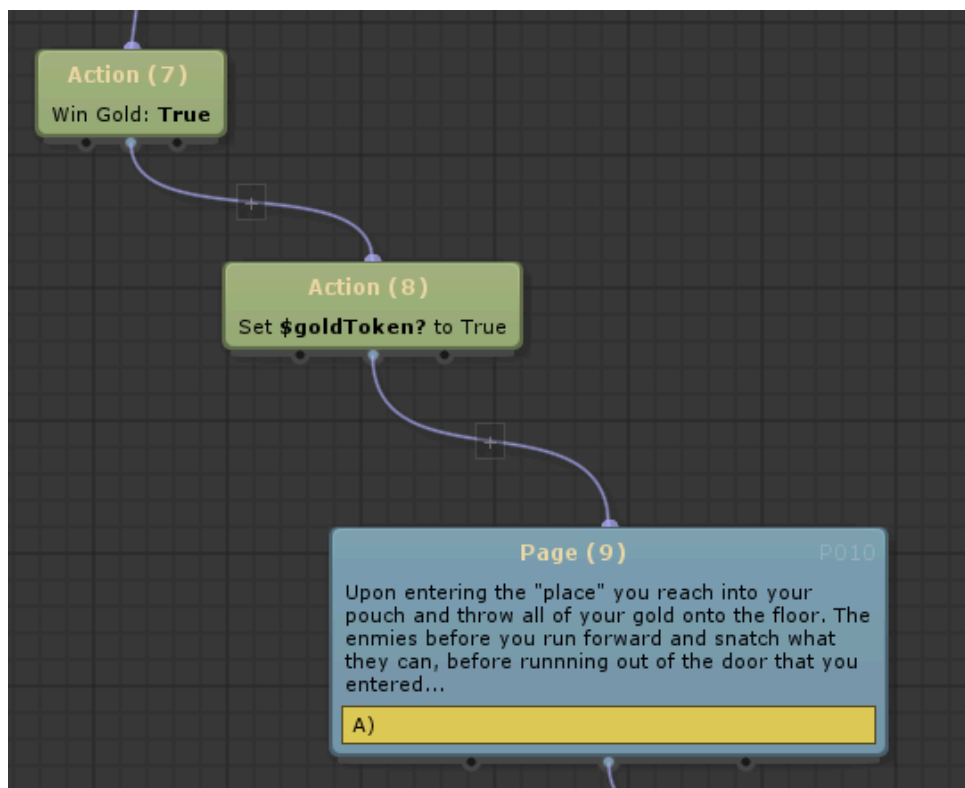
Next we have to set a variable in the Local BB of this encounter behaviour tree. Create a new boolean and name it "goldToken?". This is demonstrated below:



Now let's jump back into the left branch that we were just working in. Let's create a new action node and connect this to the bottom of the **Win Gold True** node. Let's assign the action task **Set Boolean** and set the boolean from our Local BB to true. This is demonstrated below:

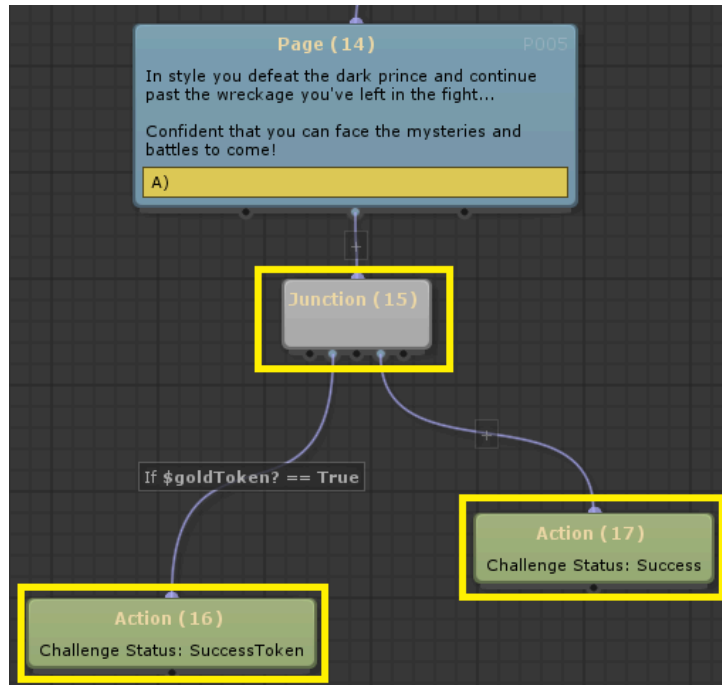


Now you can connect this action node into the top of the **Page (9)** node directly under it. Demonstrated below:

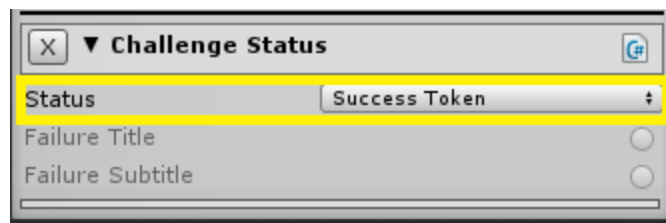


Lastly, let's set up the **Challenge Status** at the bottom of the encounter behaviour tree. If you scroll down to the bottom of the tree, you'll find the the last page of the tree, its **Page (14)**.

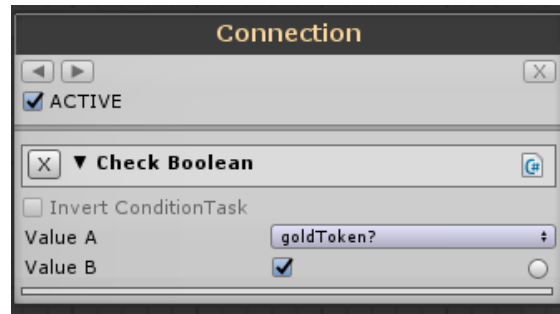
Let's create a new junction node, and 2 new action nodes. Connect the junction node to the bottom of **Page (14)**, and then connect the two action nodes under the junction node, so that you have a left and right branch. This is demonstrated below:



You'll notice that we have to set the left branch to set the **Challenge Status** to be **Success Token**, **Success Token** means that the Gold token will be unlocked, where as on the right branch, **Challenge Status** gets set to **Success** which is the Silver token unlock. Select each of the action nodes and search for the action task **Challenge Status** and set these appropriately. This is demonstrated in the below screen capture:



Lastly, we need to set the condition check for the **Challenge Status** being set to **Success Token**, this should only be true if the “*goldToken?*” boolean in our Local BB is also true. For this let’s add the condition task **Check Boolean** and check to see that “*goldToken?*” is set to true. This is demonstrated below:



Nice! You’ve just set up the **End Card** so that it functions correctly with our challenge card, it’s variables, and the collect gold challenge objective.

## Creating challenge decks and card sets:

Now, by default there are a number of challenge decks assigned for you when you create a new challenge card. These include the following decks:

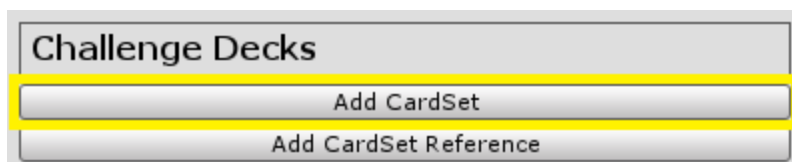
- CardSet\_Monsters\_default
- CardSet\_DefaultBlessings
- CardSet\_DefaultCurses
- CardSet\_DefaultGainPain

However, there are two **Card Sets** that you'll have to create yourself for this challenge, so let's look at creating them now. The two sets that we'll be creating are the *"DealerMarkCards"* and the *"ShoppeCards"*.

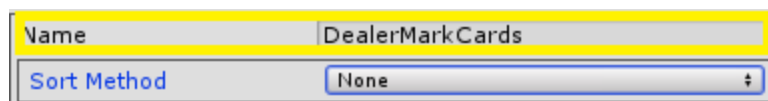
*"DealerMarkCards"* are the cards that the dealer distributes into the challenge. These are used to fill out the dungeon levels alongside the cards that are brought into the challenge by the player through the deckbuilding phase of the game.

Typically the *"DealerMarkCards"* are unique to each challenge. For this challenge we'll just be assigning 2x *"Ambush"* encounters, to the *"DealerMarkCards"* card set we're creating.

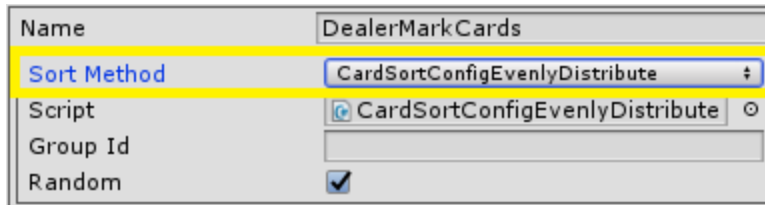
To start, let's create a new card set, to do this click the button labeled **Add CardSet**. This will create a new card set for you. Demonstrated below:



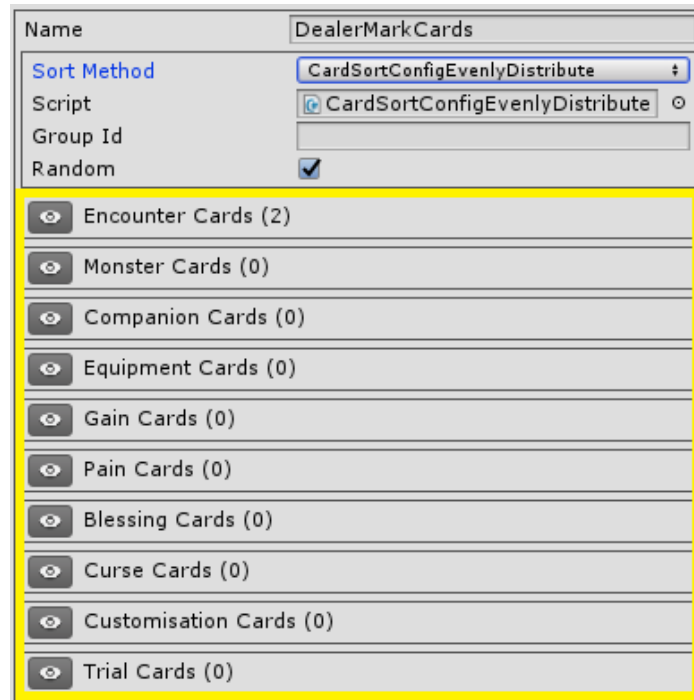
Let's expand the new card set by clicking the eye icon. Rename the card set to *"DealerMarkCards"*.



Let's also set the sort method to not be none, we'll want to set it to *"CardSortConfigEvenlyDistribute"*. What the sort method does is it evenly distributes the cards in the set throughout the dungeon levels of the challenge, when we've set it to do so.



Let's look at adding the "Ambush" encounter card to the "DealerMarkCards" now. You'll notice that within the expanded view of the "DealerMarkCards" card set, you'll have a bunch of different eye icons with specific names. It should look like the screen capture below.

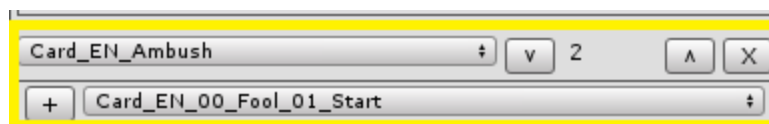


These are the different types of cards that we can add to the "DealerMarkCards" card set. We only really care about the "Encounter Cards" list, so let's expand that and look at adding our "Ambush" encounter cards to the list of encounter cards.

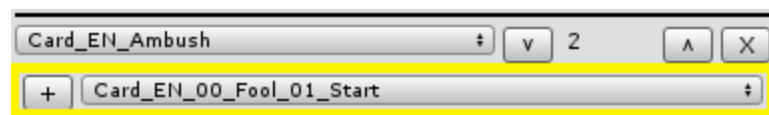


Encounter Cards (2)
Monster Cards (0)
Companion Cards (0)
Equipment Cards (0)
Gain Cards (0)
Pain Cards (0)
Blessing Cards (0)
Curse Cards (0)
Customisation Cards (0)
Trial Cards (0)

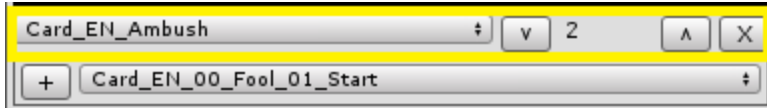
You'll notice that when we expand the *“Encounter Cards”* list that it will turn green, and a new section will appear at the bottom of the card set (highlighted in the below screen capture).



At the bottom you'll have a small addition symbol attached to a button, and next to that a drop down list where you can search for cards. Let's click the drop down list and search for the card *“Card\_EN\_Ambush”* and once we have that selected, click the addition button.



The *“Card\_EN\_Ambush”* will now be added to the *“Encounter Cards”* list of our *“DealerMarkCards”* card set.



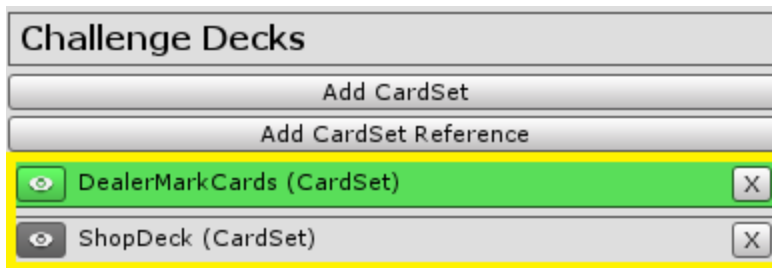
By default the amount of “*Card\_EN\_Ambush*” encounter cards in the list will be 1, but we want 2 of them in our list. By clicking the up arrow to the far right of this component you’ll increase the amount of this encounter card, that their are in the list.



When you have 2 of the “*Card\_EN\_Ambush*” encounter cards in the list, you can minimize the whole card set, and that’s the “*DealerMarkCards*” card set created.

But you’ll need to also create another card set named “*ShopDeck*”, and within card sets “*Encounter Cards*”, you will need to add the encounter card “*Card\_EN\_Shoppe*”, but make sure you only add one of these this time. To do this follow the same steps as outlined above.

After you’ve created the two card sets for this challenge they should appear at the top of the challenge decks like in the screen capture below.



## What is a dungeon:

A dungeon is used to set up how the challenges encounters are laid out across a map, which card they start on, which cards progress them to the next map layout of the challenge, and which encounter is the finale encounter for the challenge.

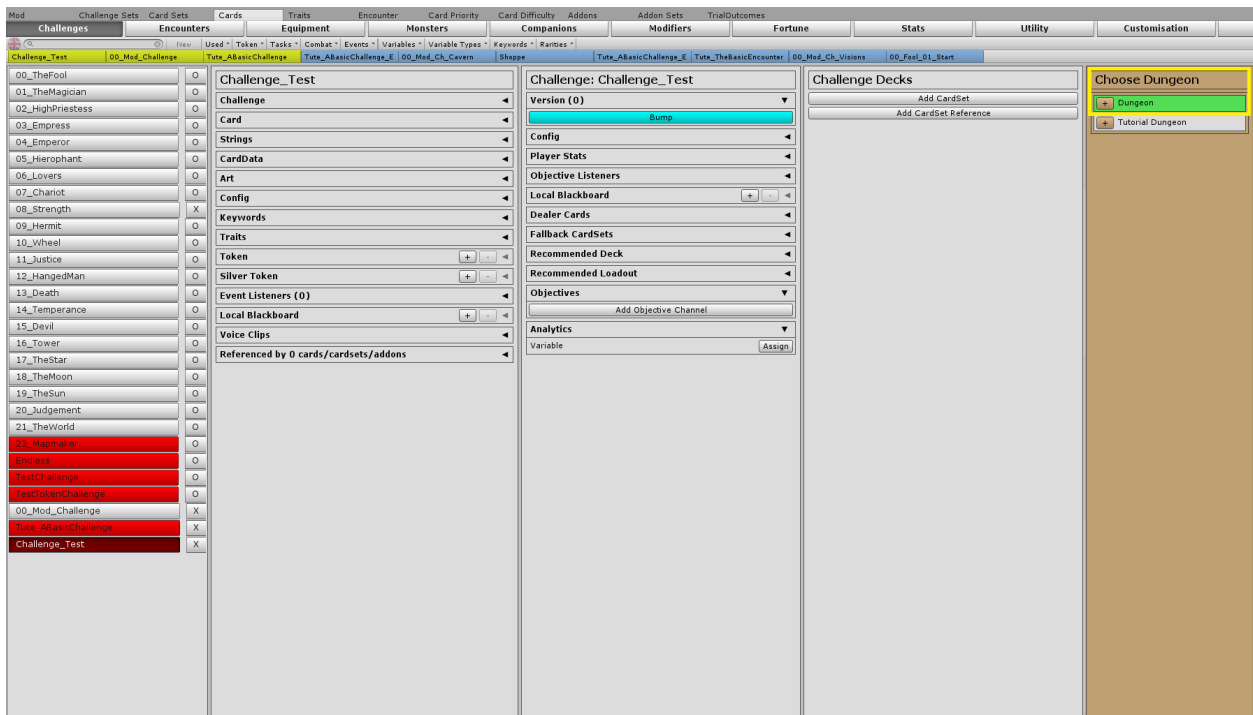
Additionally, this is where the number of Encounter and Equipment cards the player can bring on the challenge as well as any restrictions to Platinum and Brimstone cards.

Gambit difficulties can also be set globally here and affect all encounters (note that these settings were not used in any challenge at the release of HoF2).

On the right side of the screen when we first created our challenge card, you may have noticed a large section highlighted in orange, called **Choose Dungeon**. Let's set this up next.

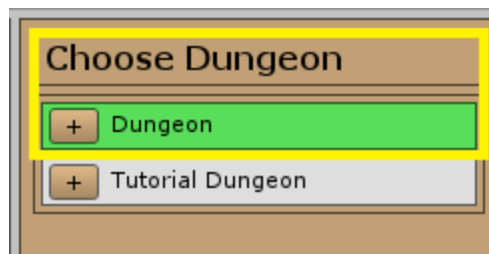
## Creating the dungeon for the challenge:

Now that we're up to creating the dungeon of the challenge, you should have noticed the big orange section of your screen to the right of the card editor when viewing the challenge cards data.



There will be a couple of options for you here. You'll be able to create a **Dungeon**, and something called a **Tutorial Dungeon**. Tutorial Dungeons are a special kind of dungeon created specifically for the fool challenge in HoF2, when you play the fool challenge for the first time it's a different dungeon than the subsequent times you've played it, this is what **Tutorial Dungeon** does, so you don't need to worry about it.

Let's start by selecting the small addition button next to **Dungeon**:



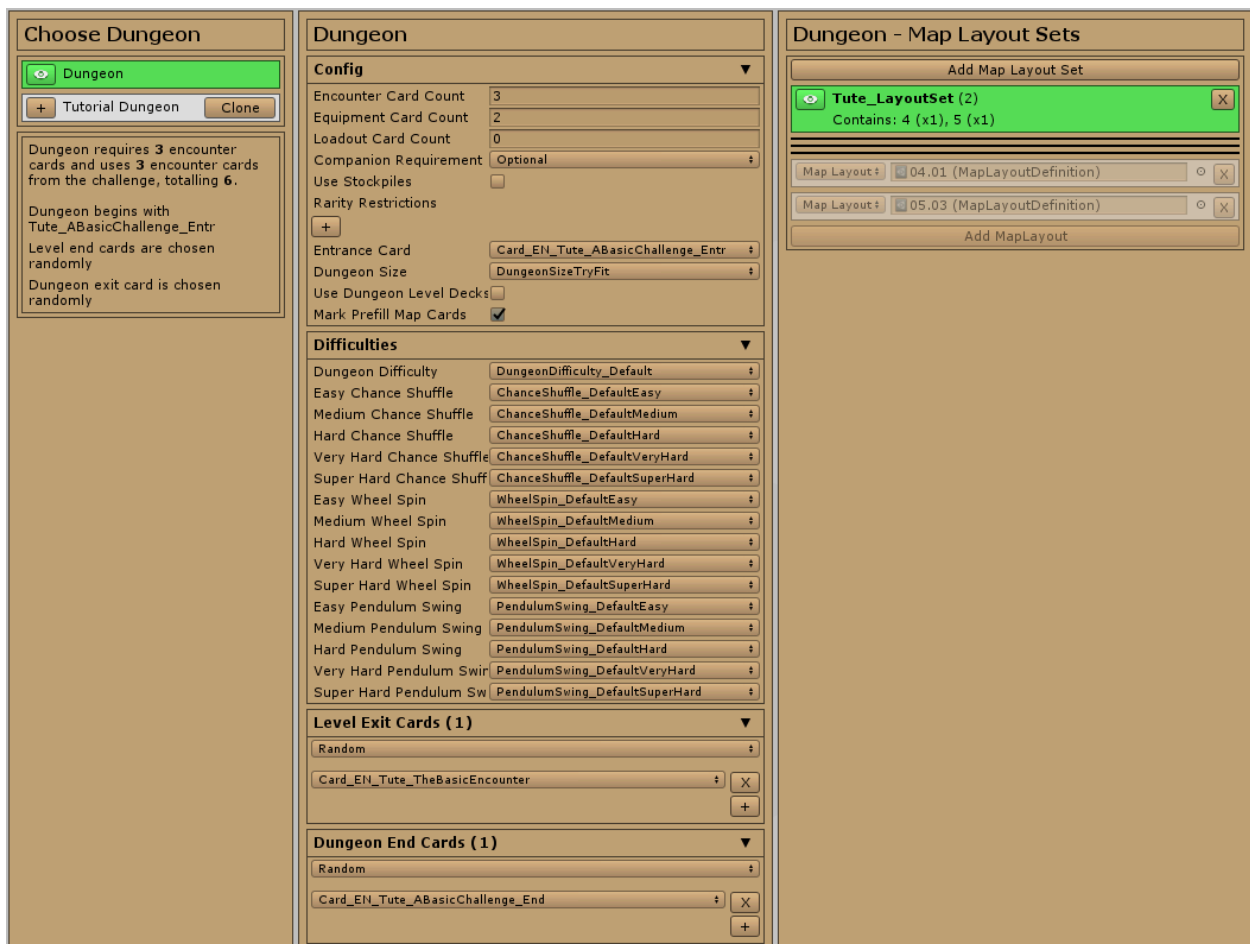
After selecting this you will have just created your challenges dungeon, and with it a whole lot of new stuff will have appeared in front of you, this is all of the new data that needs to be entered for the dungeon of your challenge, so let's look at this next.

# Dungeon Data:

In front of you there will be some labels, these will be as follows:

- *Config*
- *Difficulties*
- *Level Exit Cards*
- *Dungeon End Cards*
- *Dungeon - Map Layout Sets*

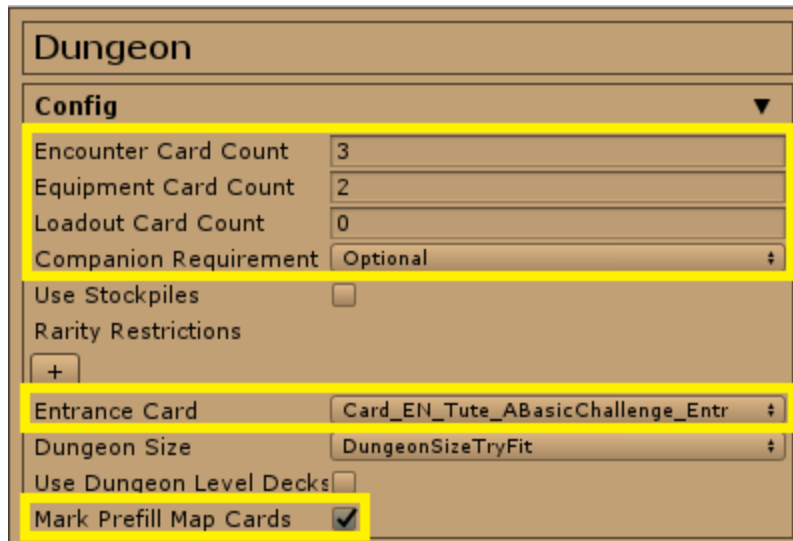
Let's start to unpick these areas a little more and explain what they all do for your dungeon. You should have something similar to the screen capture below, in front of you.



## Assigning the config data for the dungeon:

The **Config** data for the dungeon is where we can set a lot of the unique properties of our challenge, it's also where we set the **Entrance Card** for the challenge. The entrance card is the first card of the challenge, and should be introduction in its nature and presentation.

Let's take a closer look at this **Config** data, below is a screen capture highlighting the parts we'll be looking at:



The screenshot shows a 'Dungeon' configuration panel. The 'Config' section is expanded, showing several settings. A yellow highlight box encompasses the 'Encounter Card Count' (3), 'Equipment Card Count' (2), 'Loadout Card Count' (0), and 'Companion Requirement' (Optional) fields. Another yellow highlight box encompasses the 'Entrance Card' field, which is set to 'Card\_EN\_Tute\_ABasicChallenge\_Entr'. A third yellow highlight box is around the 'Mark Prefill Map Cards' checkbox, which is checked. Other visible settings include 'Use Stockpiles' (unchecked), 'Rarity Restrictions' (with a '+' button), 'Dungeon Size' (DungeonSizeTryFit), and 'Use Dungeon Level Decks' (unchecked).

Firstly, let's set the **Encounter Card Count** for our challenge. This count is the amount of encounter cards that we want players to be adding to the deck in the deck building phase, prior to a challenge's start. Let's set this count to be a value of 3, as we only want 3 player selected encounters to be presented over the challenge's levels.

Next, let's set the **Equipment Card Count** for our challenge. This count is the amount of equipment cards that we want players to be adding to the deck in the deck building phase, prior to a challenge's start. Let's set this count to be a value of 2, as we want players to be more selective of the types of gear that they will take into the challenge.

Now we'll look at setting the **Loadout Card Count** for our challenge. This count is the amount of loadout cards that we want players to be adding to the deck in the deck building phase. Loadout cards give bonuses to the player, these include additional food, gold, and health for example.

This value can be left at a count of 0. These cards are usually used only if the challenge needs it or you want your challenge to have loadout available to players. Let's set this count to be a value of 0, as we don't need players to be taking in any additional loadout cards into our challenge.

We'll look at setting the **Companion Requirement** for our challenge next. This variable is if you want the players to have the option of bringing a companion with them on a challenge run. For example, your challenge might require players to have a companion with them, and in this instance you would want to set this variable to be **Required**, instead of **Optional** or **Prohibited**. Let's set this variable to **Optional**, as we don't really mind if players want to bring along a companion.

The **Entrance Card** is the first encounter of the challenge. This encounter is unavoidable, and typically serves as the introduction to the entire challenge narrative. We've created the **Entrance Card** for you already, so you'll only need to assign the card here. Let's set the **Entrance Card** for the challenge, search for "*Card\_EN\_Tute\_ABasicChallenge\_Entr*" and select it.

**Mark Prefill Map Cards** applies a dealer mark on the corner of all of the encounters that are pre filled into a given challenge. These are the encounters that are not brought into the challenge via the player, through deck building. These encounters instead are set into the challenge, and this will never change unless you alter these decks. An example of this mark is demonstrated in the below screen capture.





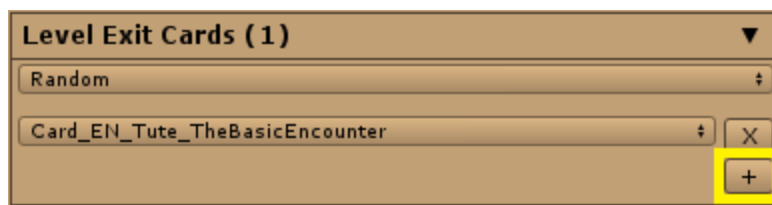
## Assigning the level exit card for the challenge:

The **Level Exit Card** is the first encounter of the challenge. This encounter is again unavoidable, and it serves as the progression encounter for transitioning into a new level of the challenge. For our challenge tutorial, you'll only need to be assigning the **Level Exit Card** that we have created for you already.

**For example**, in our challenge, players are going to be navigating across 2 dungeon levels. In order to transition from the first level into the second level, we need to add a transition encounter card, this encounter card is what we call the **Level Exit Cards**.

Our **Level Exit Cards** will typically consist of the limited text pages (but this is highly variable depending on what you want). These are typically linked into the challenge narrative and are designed so that they help players achieve the overall challenge objective. For your challenge you'll need to assign the **Level Exit Card** as your "Basic Encounter" that you created earlier in this documentation.

Firstly let's start by adding a new item to the data of **Level Exit Cards**. To do this you'll need to navigate to this data, and select the small addition button, located in the bottom left.



Doing this will create a new drop down list for you to select your **Level Exit Cards**. Enter the list and search for the card "*Card\_EN\_Tute\_ABasicEncounter*" and assign this card as your Level Exit Card.



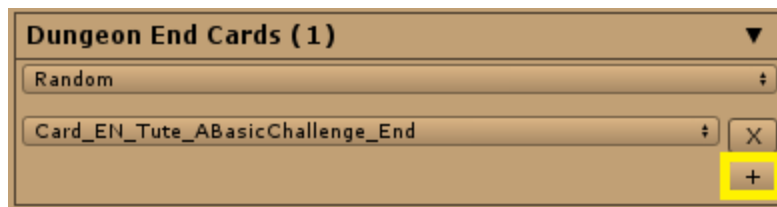
## Assigning the dungeon end card for the challenge:

The **Dungeon End Card** is the final encounter of the challenge. This is the encounter that ends the challenge, and returns players back to the challenge map. This encounter is again unavoidable. This card is the encounter that you've been working towards over the entire challenge narrative.

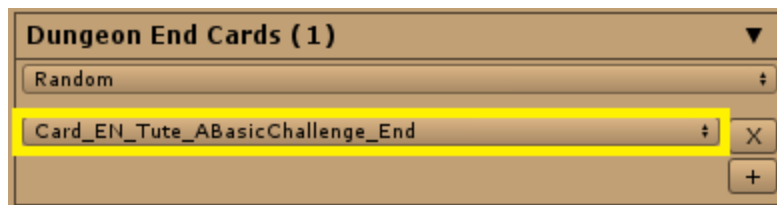
**For example**, in our challenge objective players are attempting to collect 100 gold so that they can bribe some enemies out of combat.

Our **Dungeon End Card** will consist of the concluding test pages, the bribing system, and the combat scenario. For the challenge tutorial you'll only need to be assigning the **Dungeon End Card** that we have created for you already.

Firstly let's start by adding a new item to the data of **Dungeon End Cards**. To do this you'll need to navigate to this section of data, and select the small addition button in the bottom right.



This will create a new drop down list for you to select your **Dungeon End Cards**. Enter the list and search for the card "Card\_EN\_Tute\_ABasicChallenge\_End" and assign this card as your Dungeon End Card.



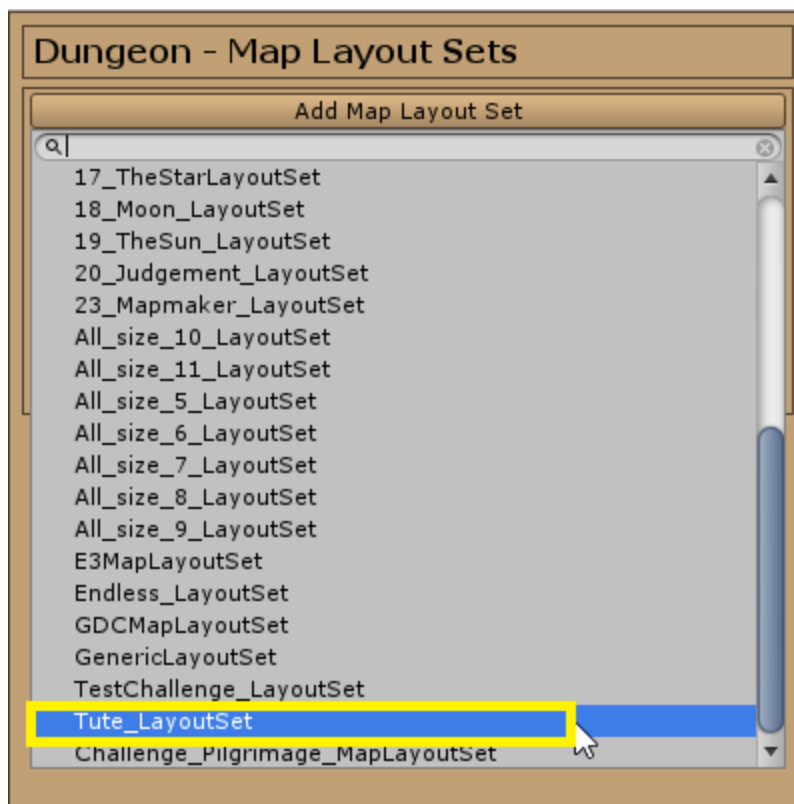
## Assigning the dungeon map layout:

A **Dungeon - Map Layout Set** is a collection of dungeon maps that have been created for a specified challenge, these are also known as the levels. These sets are created as an easier way of assigning large amounts of maps for a challenge, instead of individually assigning them into the challenge card data.

Firstly, let's navigate to the **Dungeon - Map Layout Set** data. Let's select the button **Add Map Layout Set**.



After selecting this button, you'll be presented with a list of the available Map Layout Sets currently in the tools, these are the HoF2 challenges and there are some test sets located here as well. Let's search for a set that we've created for you already, it's name "*Tute\_LayoutSet*".



Select and assign this set to your challenge cards dungeon data.

# How can I easily test a challenge in the Unity editor?

It's likely that you'll have to start a new game if your testing your content in the Unity editor. The save files that you have available in Unity are different to your save files when you play HoF2 through steam. Starting a new game means that you'll have to go through the tutorial challenge before accessing any of your modded content, this is at least the case with challenge mods, Encounter mods work a little differently and are easier to test in most cases.

So there are two ways that you can test challenge mods through the Unity editor. One way is to load the 'DevDungeon' and set your challenge mod as the challenge to be played. The other is to create a 'Challenge Addon' and activate the content as you would through steam.

**Note:** It's good practice to be testing your content in Unity through addons. Addons are how users are going to be accessing your modded content in their game, so you should test this for yourself.

There are two types of Addons that you can create for your modded content, these are the 'Challenge Addon' & 'Token Addon'. These are what you're actually enabling on a save file when you're loading into the game.

**Note:** Enabling any encounter mods on a new save will take you straight to unlocking the 'Token Addon' and then to the challenge map, this will automatically skip the tutorial challenge (Fool) if it's a fresh save file. 'Challenge Addons' do not do this because they're stored on the challenge map and are not linked to a token unlock.

If you're wanting to test your challenge mods through the Unity editor and not through the steam. You'll want to be able to skip some of the early game flow so you can get straight to testing... To do this you'll have to enter the debug menu and turn on some cheats:

If you're using your keyboard you can press the `` key found under the 'Esc' key to open the debug menu.

Alternatively if you're using a gamepad you can press in both of the joysticks which will display the debug menu.

Once you've opened the debug menu let's navigate to the 'cheats' tab and let's turn the following cheats on:

1. **Skip Early Game Flow**
2. **Skip Introduction**

You'll probably have to run the game again for these changes to take effect if you're already in the tutorial sequence.

Your challenge mods will be located under a tab in the top left of the challenge map, this tab is named 'Mods'. **Note:** that these will only appear here if you've created and are accessing your mods through a 'Challenge Addon'.

## Section 6: “*Intermediate Encounter Design*”

In this section of the documentation we'll be looking at some more intermediate encounter design, that you can use when creating your own encounters in the card editor.

### What are Token Chains?

If you've played a great deal of HoF2 you may have come across some encounters that have a token on them, and if not! Now you know.

**For example**, all companions in HoF2 have their own unique quest lines that you can play through, these quest lines are played over multiple encounters. Each encounter in the questline has a token attached to it where you would receive the cards token after completing the encounter. This is what we call a **Token Chain**.

Comparatively the **Old Maiden** encounter also uses a **Token Chain**. This one is much shorter to complete in comparison to a companions quest line and is not as difficult.

**Token Chains** are used when you want to string together a set of encounters. They typically don't impact the main narrative of the challenge that they're played in as they're completed and played over numerous runs.

### How can I design my Token Chains & Encounter Sets?

If you want to use **Token Chains** in your modded content then you'll first need to design the Encounter Set (however this is not a prerequisite). When designing a set of encounters that are interrelated and play out in a specific chronology (a campaign or adventure) you'll want to try and follow a checklist of some sort, personally I always try to have the following in my designs:

- Introduction to the narrative that's going to be told over the Encounter Set (**Beginning**)
- An objective is always good! (**Tasked Objective - Acquire this item**)
- A complication can be integrated, but it may not be needed. (**Complication; Combat?**)
- The return with objective complete (**Objective Complete - Deliver that item**)
- Conclusion of the narrative being told, occasionally alongside a greater reveal (**End**)

In comparison, let's look at the **Old Maiden** encounter:

- Old Maiden introductory and asks if you'll help (**Beginning**)
- Acquire the Potion of Youth and bring it to Old Maiden (**Tasked Objective**)
- Have acquired the Potion of Youth and bring it to Old Maiden (**Objective Complete**)

- Greater reveal of the now transformed Maiden + get a new card now (**Conclusion**)

With these examples you would have an encounter for each of the dot points listed, then each of those encounters would have their own token to be won where the reward of that token would be the next encounter in the set.

**For example**, your **Encounter Set** has 5 encounters in it. Encounter 1 = beginning, and Encounter 5 = conclusion. The player is currently playing Encounter 2 which have them receiving the objective within our encounter sets narrative.

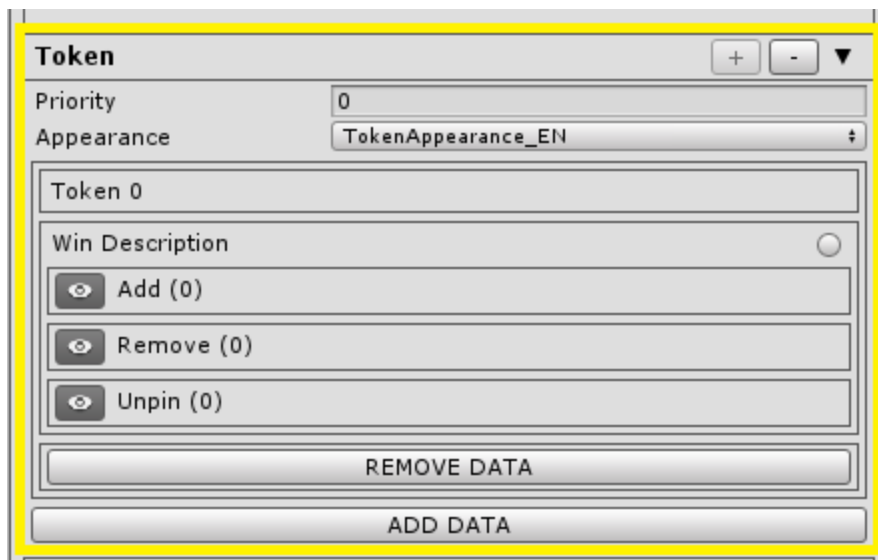
The player receives the objective after accepting the offer. The token that's present on the Encounter 2 card has now been won by the player and is held in the token dish until the end of the current challenge run. When the player finishes their current run they proceed to unlock their tokens and the token won from the Encounter 2 card will now reward the player with the Encounter 3 card.

## Adding tokens to your encounter cards:

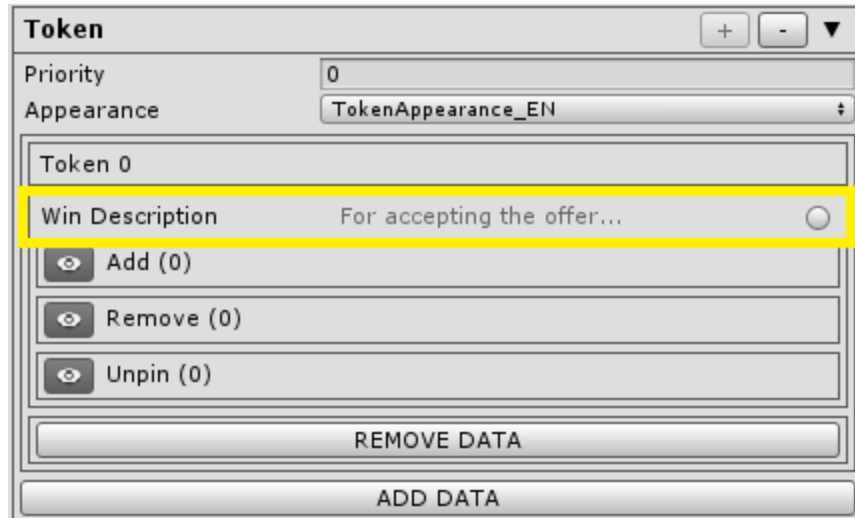
When you're navigating the data of an encounter card, you will spot a label named **Token**. If you completed the tutorial for creating the Basic Challenge, then adding tokens onto your encounters, is the exact same process as creating Silver Tokens for your challenges.

The process of adding tokens to encounter cards will be demonstrated in the below screenshots.

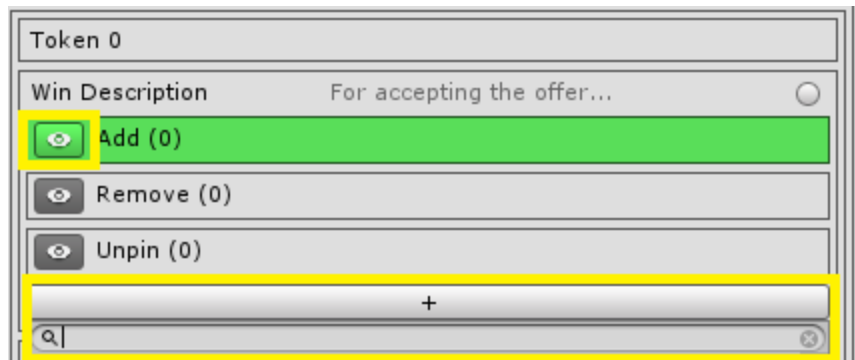
When you expand the **Token** label, you'll be presented with this data.



This data, as you'll notice, needs a **Win Description**. A **Win Description** is the text that is displayed on the players screen, when they unlock a won token at the end of a challenge run. Still using our above example of the player completing the Encounter 2 card, and then being rewarded with the Encounter 3 card, from the token unlock. We would want to add a win description that is relevant to that progression. For example, we could write something like *“For accepting the offer...”* which indicates that the player has accepted the tasked objective.



The next step would be to **Add** the cards that you wish to be added to the players deck, so that the Encounter Set can continue. This is also the card that is displayed when the tokens are unlocked. For example, you would expand the **Add** label by clicking the eyeball button, where you would now search for Encounter 3 and add it to the list.



It's important to note that you would initially have the encounter that is the first in the set, available to players in the deck builder.

## Creating Basic Spices:

Spices are pieces that are present on the dungeon maps, laid out on the card table.

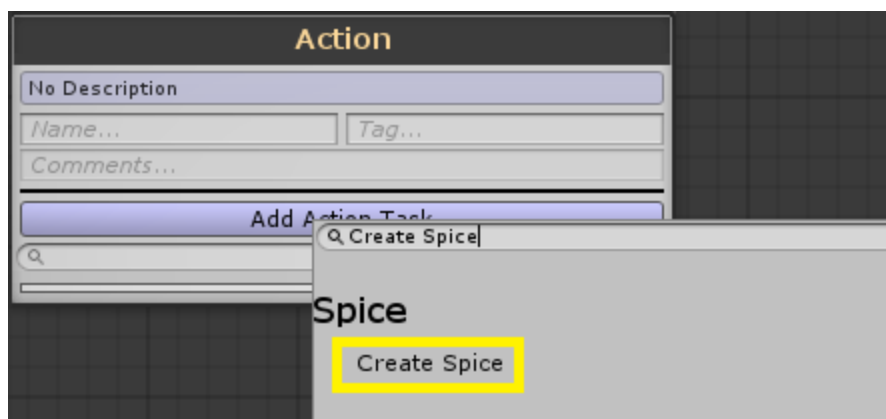


You may have noticed in the challenge **Emperor** of HoF2, that the first level of the challenge has a gain gold spice that awards you 10 gold when you land on it. If you had never noticed this, here it is demonstrated in the below screen capture.

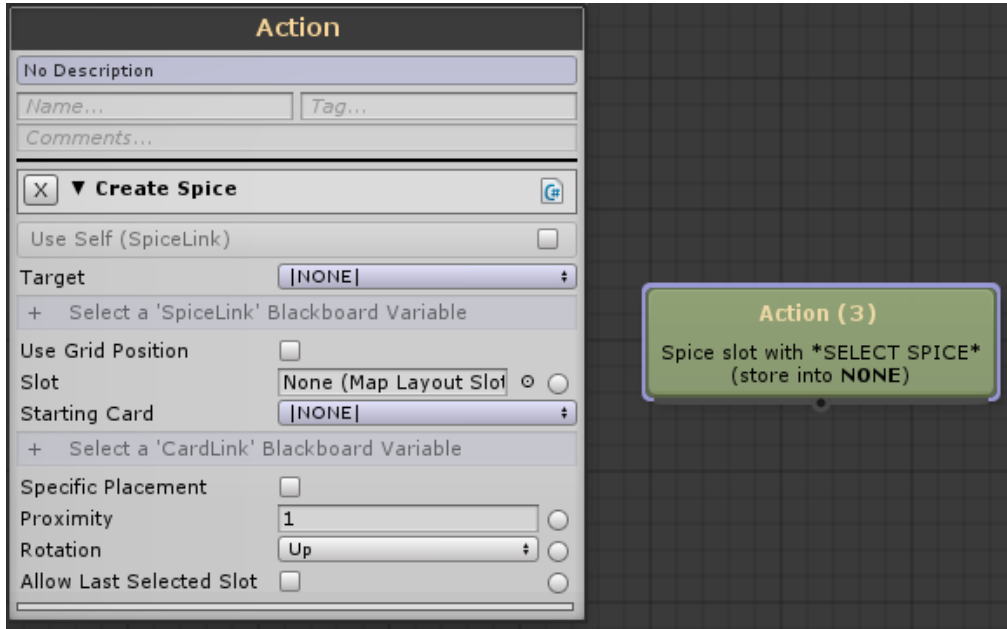


Let's run through creating a spice like this that could be plugged into our entrance card for the Basic Challenge, as this spice would work into the challenge narrative and objective rather well.

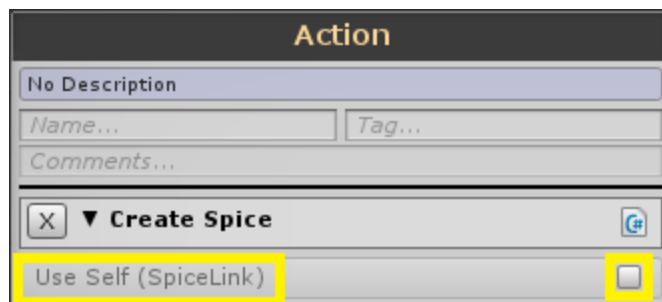
When you enter the encounter behaviour tree of an encounter card, you will want to create an action note and search for the action task **Create Spice**.



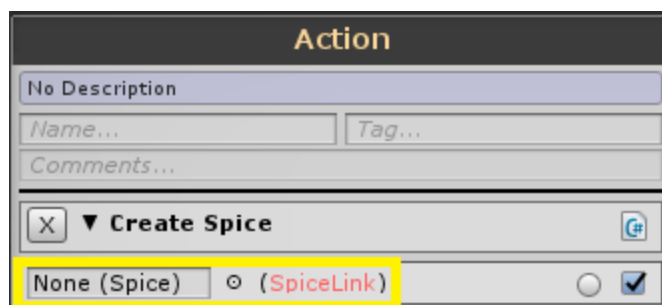
When you've created the action task **Create Spice**, you will have following data available.



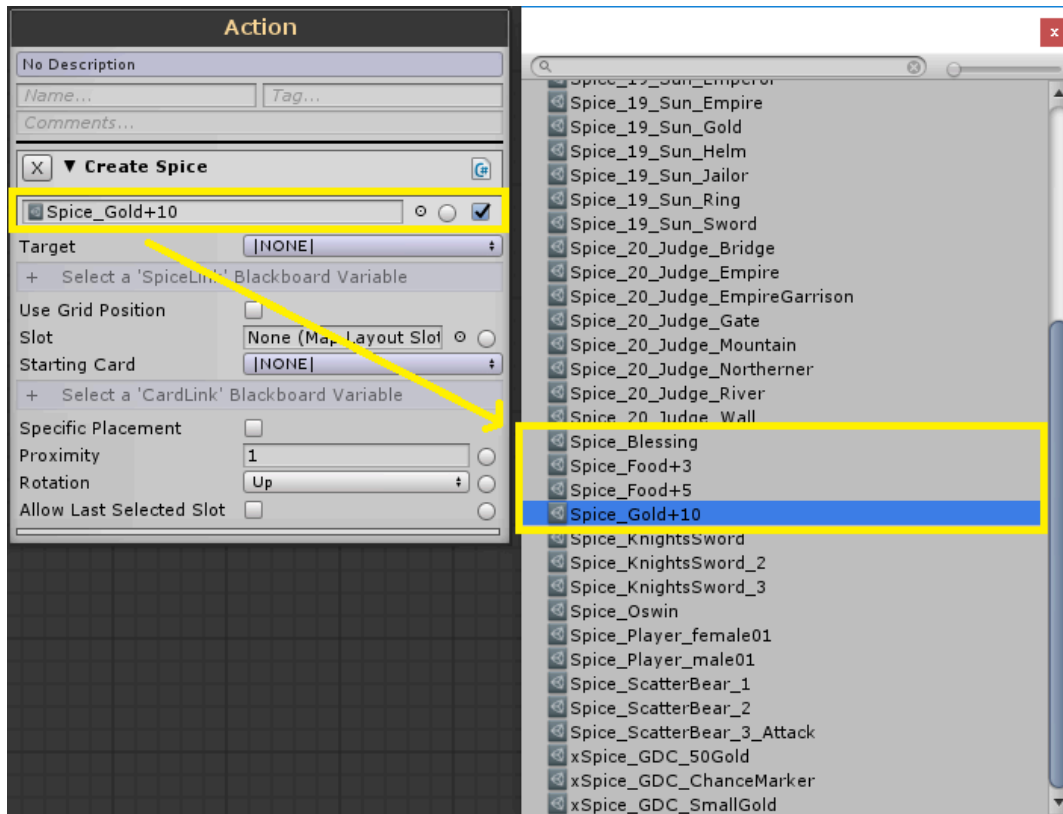
There are a few things that we want to assign in the **Create Spice** data. The first thing is selecting a **SpiceLink**, this is located at the very top. To do this you will have to check the small box to the right of this data. These steps are demonstrated in the below screen capture.



After checking this box, this data will change and will be asking you to select a **SpiceLink**.

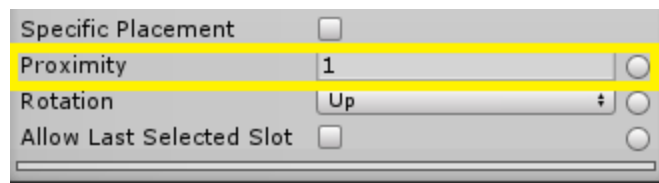


Click on the open field and you'll have access to all of the spices currently existing in HoF2 (you should only need to use these and not have to create your own ones). Using the generic spices such as gain gold and gain food, are typically the spices that you'll want to be integrating into your encounters. This is demonstrated in the below screen capture.



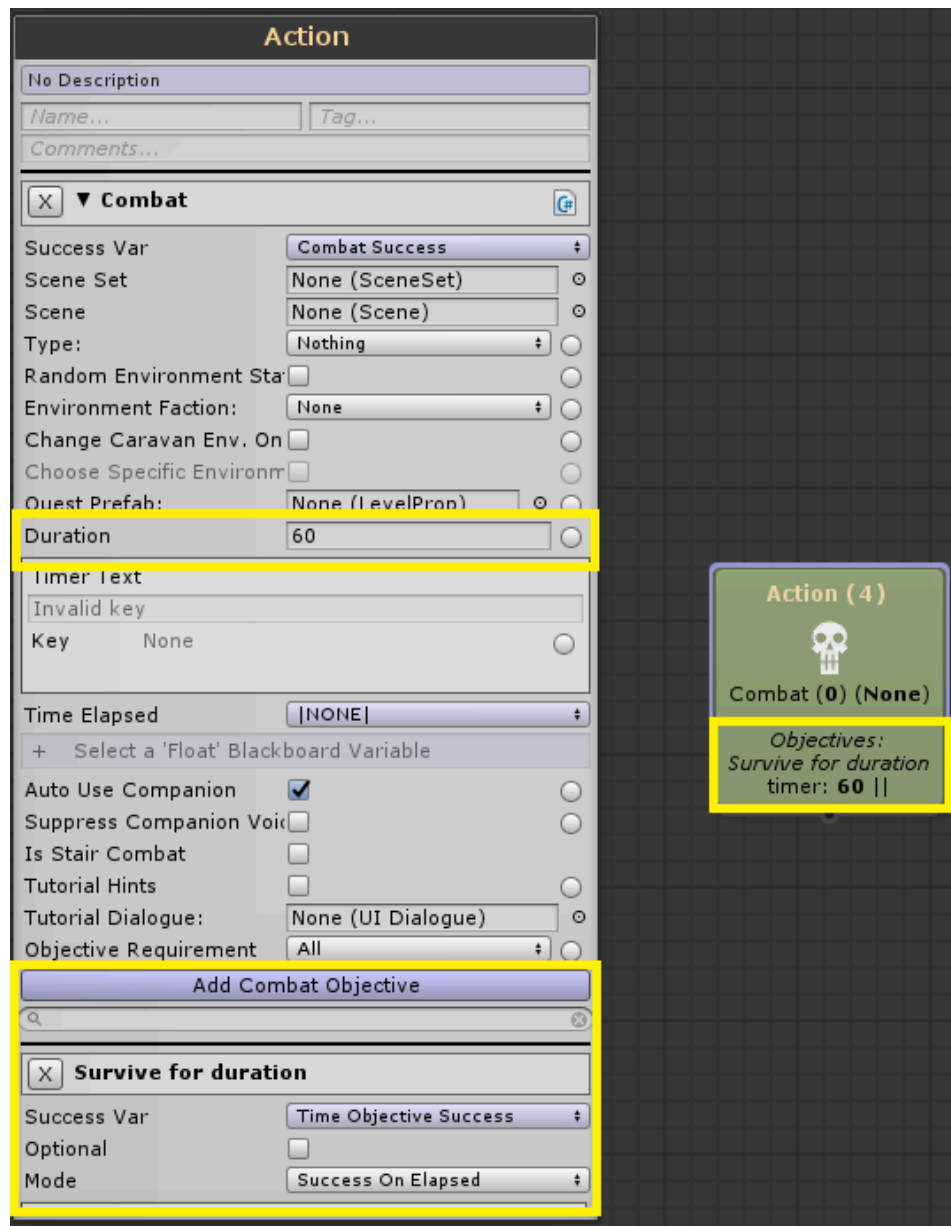
After selecting the applicable spice that you'd like to use in your encounter, we can then look at setting some data for how it should appear on the dungeon level. For this example let's select "Spice\_Gold+10" as our **SpiceLink**, this is a good example for you as its a spice that could be integrated into the Basic Challenge rather well.

You're able to set how the spice should appear in proximity to the player on the dungeon level. For example, demonstrated in the below screen capture, if the **Proximity** is set to a value of 1, then the "Spice\_Gold+10" would appear on a card within 1 movement space of the player meeple.



## Adding timed combat and combat objectives:

You can add some spice into your life with combat objectives and timed combat moments. These are a really easy way of making tense moments in your combat scenarios, and are super easy to set up! Below you'll see how to create a combat scenario with an objective **Survive for duration**, and setting that duration to be 1 minute.

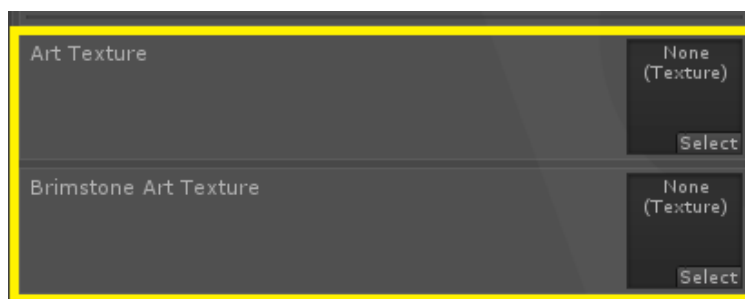
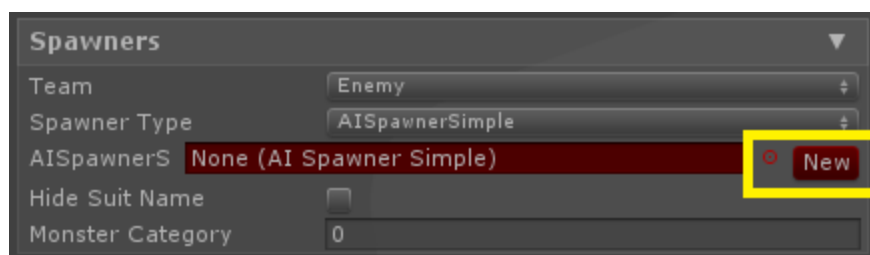
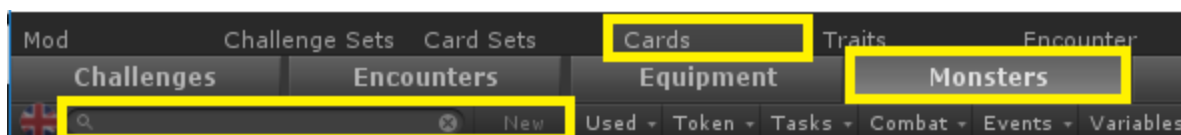


Firstly, click **Add Combat Objective** and select the objective **Survive for duration**. Next, enter the value of 60 into the **Duration** data. Lastly, if you look at your action node holding this combat scenario, you'll see the objective and the timer present at the footer of the node.

## Creating your own monster cards with custom values:

While you *can* create your own monster cards using the SDK toolkit, it's not the easiest thing to do... I'll run through how you can create them, but know that we're working to make this process better for you.

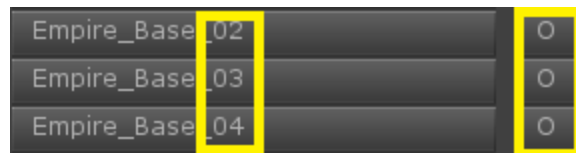
Start by navigating to **Cards > Monsters** and let's create a new **Monster** card. Name it whatever, and let's set some data. Click **New** next to the (AI Spawner Simple) section, and add some card art relevant to the type of enemies you want for your monster card.



One thing that you need to setup with custom **Monster** cards are the **Spawners** that actually bring the enemies into the combat phases of the game.

**This is where things get a little tricky...** I'm going to suggest that you duplicate an existing monster card with predefined spawners so that the step of setting up your spawners is handled for you, and is a little less painful. The reason that I suggest this step is because we've done the work to set these spawners up appropriately to the combat scenes in HoF2. If you go in and make your own ones you may find that some of your enemies will spawn outside of the map.

**How can you duplicate an existing monster card?** Well if you search the list on the left side of the card editor you'll find all of the different monster factions in HoF2. Let's expand the **Empire** group for example, and look at the same cards demonstrated below:

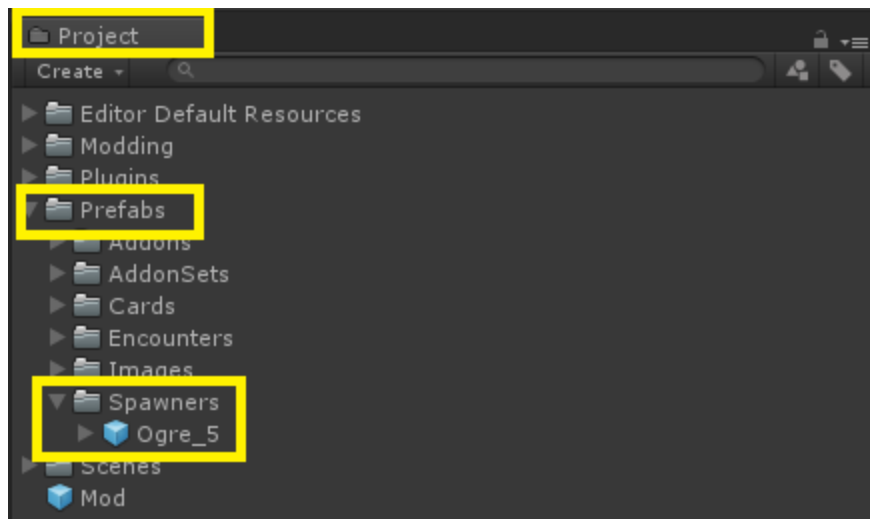


Typically we name the monster cards according to the amount of monsters to be spawned and the enemy type. **For example**, we can see 3 monster cards which are the **Empire\_Base** enemy type (above), we then have a number that follows this name. This number is the amount of enemies in that monster card, and that translates to the amount of spawners that this monster card has set up on it.

So decide on the amount of enemies that you'd like your custom monster card to spawn in combat and select an appropriate card with that amount of spawners and edit it.

You can edit cards by clicking the 'O' button to the right of the card in the list (highlighted in the above example). After you've clicked edit on an existing card, jump into the Unity editor.

Navigate to your Project window and let's look at what we've just created by creating our own monster card and also by selecting an existing card as editable.



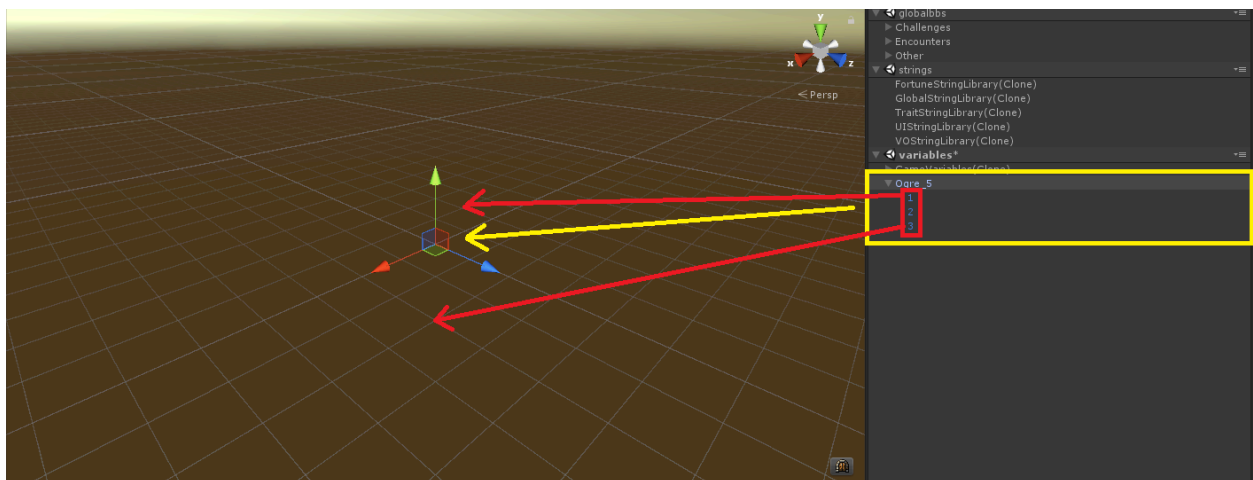
Navigate to **Project Window > Prefabs > Spawners > "whatever you named your card" & "the selected card that you made editable"** should appear here as prefabs (example above).

**Note:** The example above shows **Ogre\_5** as an available spawner, this is one of my custom monster cards that I'm using as an example. Yours will display as whatever you named your custom monster card when you created it in the card editor.

Let's select the prefab spawner for the existing monster card that we made editable (**Project Window > Prefabs > Spawners**) and drag it into the **Hierarchy**. You can also drag your custom monster cards spawner into the hierarchy too.

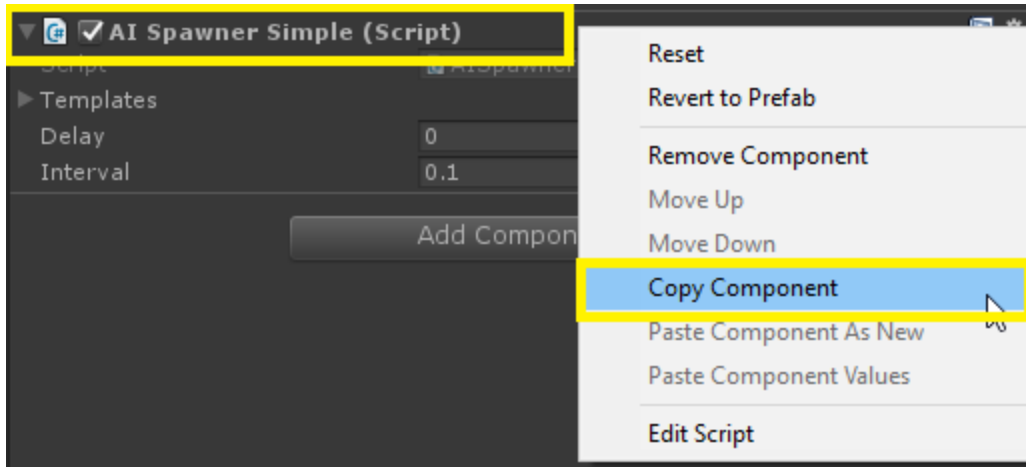
If you expand the prefab for the existing monster card once it's in the hierarchy, you will see that there are a number of children to the main prefab, these children are the spawners that produce the enemy. Each child is where 1 enemy will spawn in a combat scene in game.

**For example**, the yellow arrow (below) is identifying where the main prefab is in scene, this is set at coordinates (0, 0, 0) and is nothing more than an empty gameObject with a spawner script on it. The children are the objects highlighted in red (below) where the red arrow is pointing to where these spawn locations *could* be in relation to each other and the main prefab.

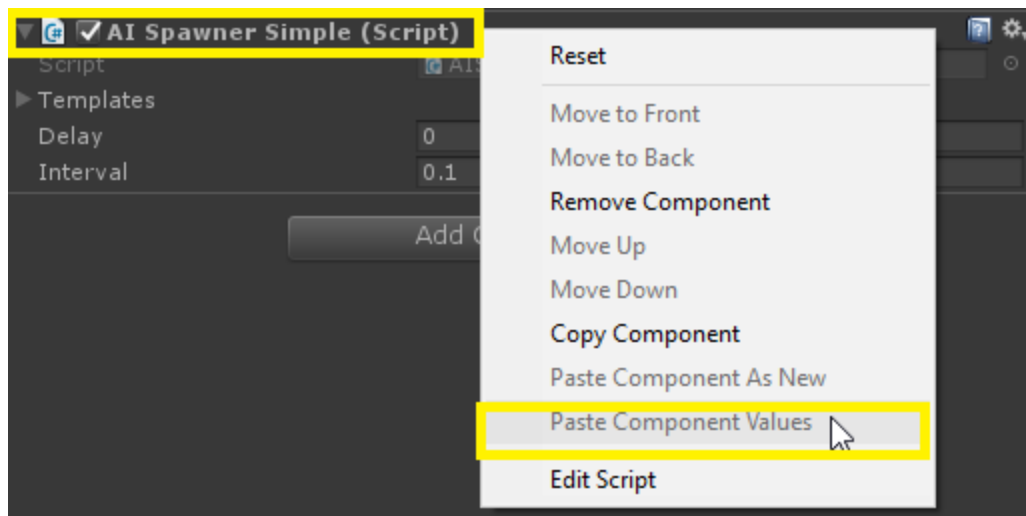


Now let's look at what you need to do for setting up the spawner for your custom monster card.

Select the prefab for the existing monster card that you've made editable and view it in the inspector. There will be a script named "*AI Spawner Simple (Script)*", right-click this script and select **Copy Component**.

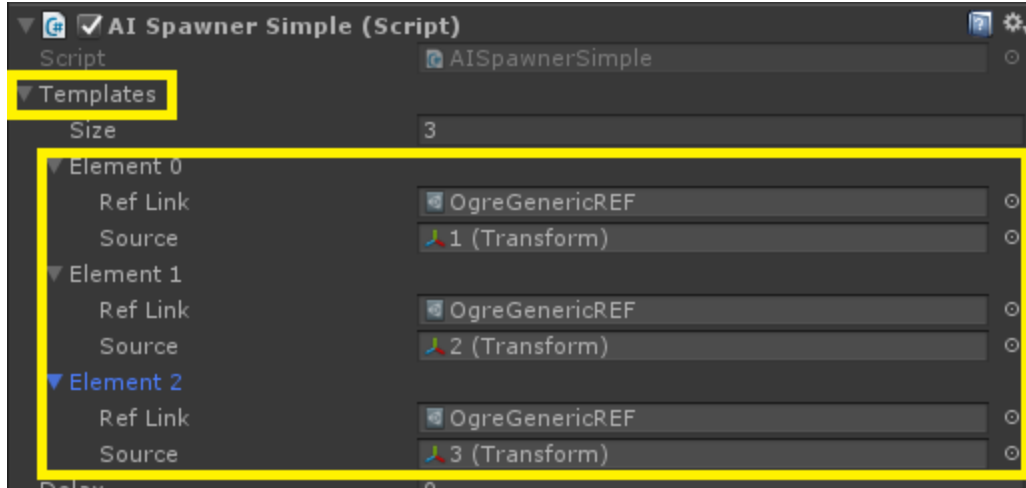


Now select your custom monster cards spawner which should also be in the hierarchy. View the gameObject in the inspector and you'll see the same spawner script but with no values set... right-click the **Scripts** name in the inspector and select **Paste Component Values**. This will add that spawner script values from the existing monster cards spawner and add them to your custom monster cards spawner.



If you expand the **Templates** of the "AI Spawner Simple (Script)" you'll see that the templates are the same as those found on the existing monster cards spawner script and this is what we want!





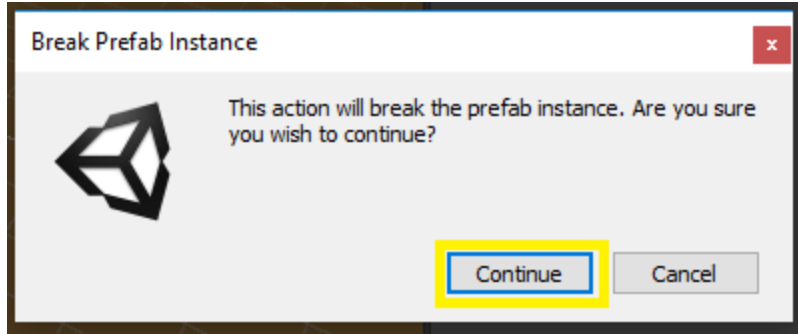
**Note:** For your own monster card you may not want the enemy type that is set (whatever enemy type the existing monster card was). You can change this in **Templates > Element > Ref Link**. Above you can see that I've set mine to be the **Ogre** enemy type instead of another enemy type.

You'll also notice that the **Templates > Element > Source** is set to each of the child gameObjects of the existing monster cards spawner. The spawners have been set up in this case and we just have to move the spawners from the existing monster card spawner, to our custom monster card spawner.

Jump back to the hierarchy and let's move the spawners to our spawner (custom monster spawner). Select the child spawner gameObjects and drag them to your custom monster card spawner (example below).



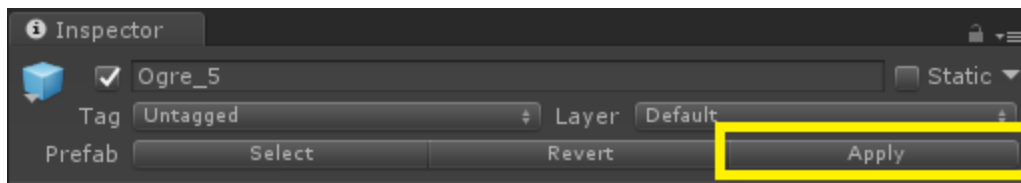
Unity will warn you that this action will break the prefab instance for the existing monster card spawner, say continue (we'll delete the edits we've made to this existing monster card anyway).



Delete the prefab of the existing monster card spawner from the hierarchy so that you can just see your custom monster card spawner (example below).



Select your custom monster card spawner and click **Apply** to the prefab, this will be located at the top of the inspector after selecting the custom monster card spawner in the hierarchy.



You won't be able to cycle this custom monster card into the HoF2 monster deck, but you can use the **Get Card(s) By Prefab** action task to retrieve the monster card and then activate it in one of your encounter mods.

## Awarding Fame after Encounters:

These are the guidelines that we follow for awarding fame after encounters:

**No Fame** - player takes no risks / no combats, i.e. General Store, Trading House, Healer.

**smlFame** - player takes some risks i.e. chance of drawing pain cards, or has an easy combat, e.g. *"Fallen Treasure"* encounter.

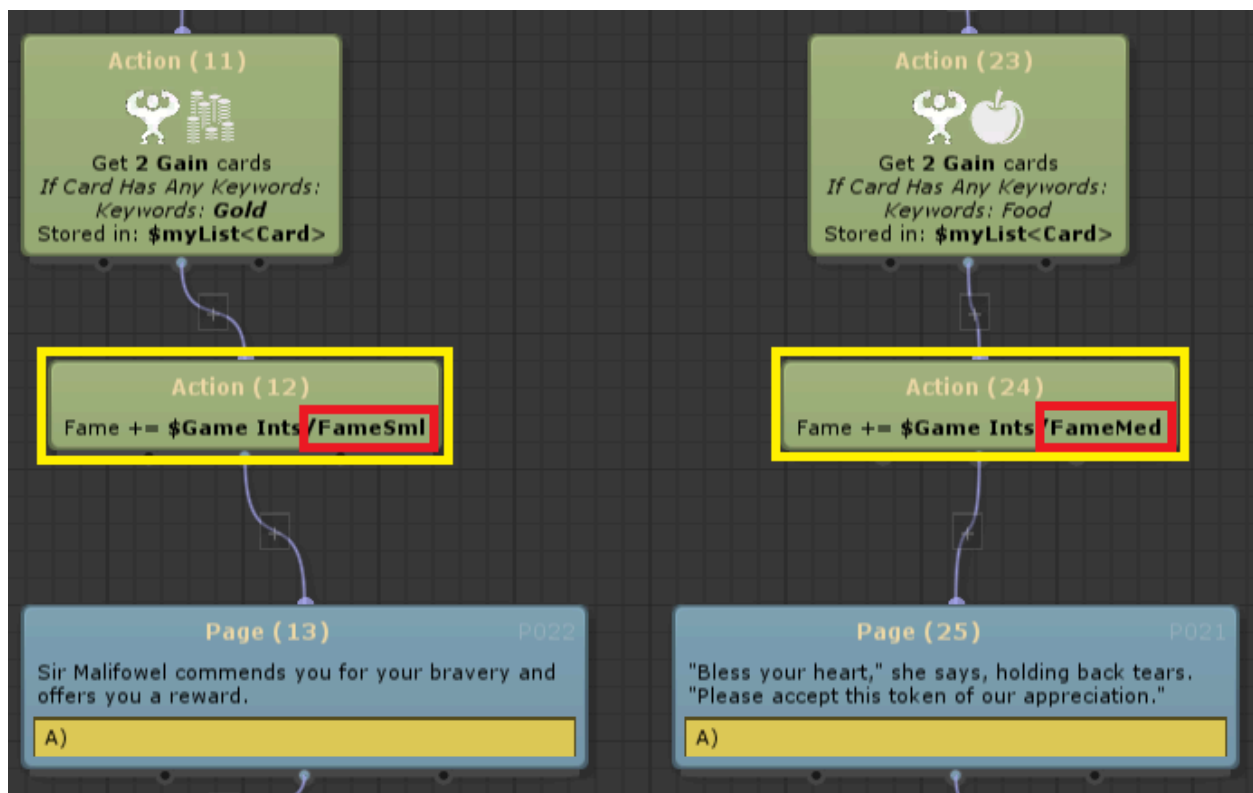
**medFame** - player has some risks, and or a standard combat, i.e. *"Highway Rescue"* encounter.

**lrgFame+** - did something to specifically gain Fame and that is the purpose of the encounter from a deck building perspective, i.e. *"Fame and Shame"* encounter.

**smlFame**, **medFame**, **lrgFame** are all variables in the game, and can be found in **GameInts**.

When you're awarding **Fame** after your encounters, in some instances, depending on what choices/outcomes the player makes/gets, you may want to award different amounts of **Fame** from the same encounter but based on the above rules.

For example, there is an encounter in HoF2 named *"Pauper Plague"*; if the player chooses to fight the paupers they receive the **smlFame** amount, while choosing refuse to fight = the **medFame** amount. This is demonstrated in the below screen capture.



It's important to note that whenever you are awarding fame after or during an encounter, you should always have a page of text that follows the action. This ensures that the auto announcement for awarding fame is displayed to players, and that they know that they have received the appropriate amount of fame, based on their choices.

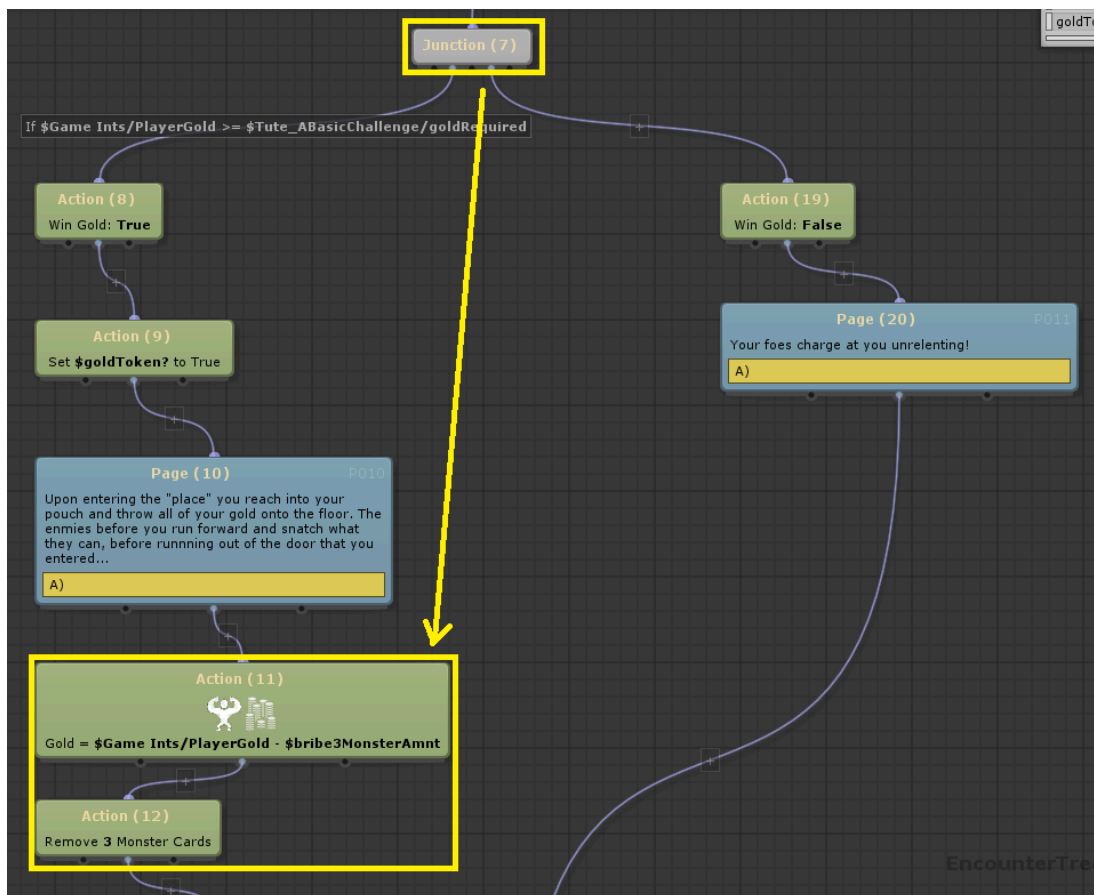
## Setting player stats directly:

If you'll recall, we created pain and gain card results for our chance card mini game back in the basic encounter tutorial. The intention behind this was to award gold for a success and then pain the player of health on a Failure result. However, there is another method that we can use to directly impact the player stats, aside from using pain and gain cards.

For example, the end encounter for the basic challenge has a scenario where the player can bribe enemies out of the final combat, so let's look at how that is done now...

Let's look at the bribing setup for this final encounter to demonstrate how you can directly change player stats, such as gold, because that's what's happening in this encounter behavior tree!

Open the encounter "*Tute\_ABasicChallenge\_End*" encounter card, and view the behaviour tree. Scroll down till you find the first junction in the node graph (this is where we check to see if the player has met the requirement of the gold token unlock, or not), it should look like the the example below.

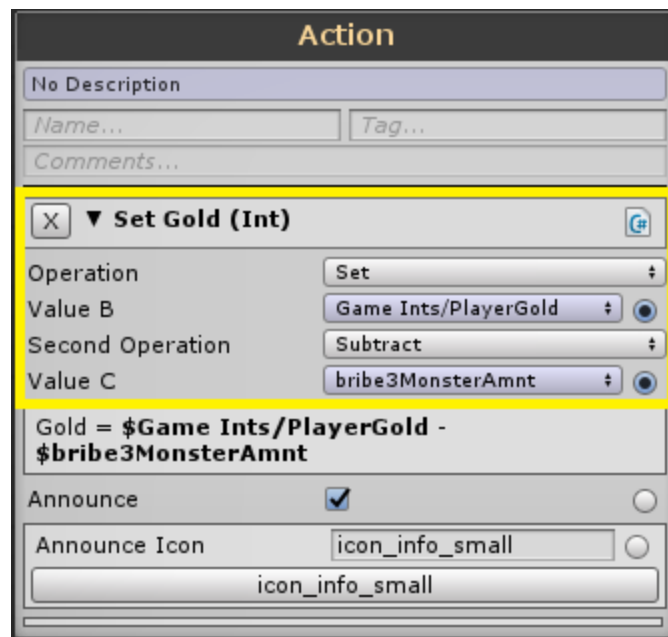


As you can see in the above screen capture, I've highlighted the junction node, and 2 action notes that are attached to the left branch of the node graph. These 2 action nodes are where we are directly changing the players stats.

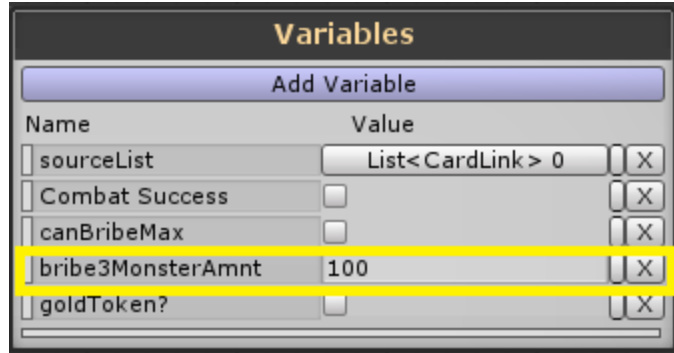
Let's take a closer look at these 2 action nodes highlighted above, so that we can demonstrate to you, how you can directly change the players stats instead of using pain and gain cards. Now, if we select the 1st action node in the left branch, we can view its data, demonstrated below.

Essentially what is happening here, is that we're directly setting the players gold (int) based on the challenge objective having being met by them. If the players collected 100 gold throughout the challenge and brought it into the final encounter of the challenge, then the gold objective has been met and the gold token should be awarded, this is done so by following the left branch of the node graph (demonstrated above).

Since the player had completing the challenge objective, they need to now pay that 100 gld which will bribe off 3 of the possible enemies that would be in the combat phases of this encounter. This is what we're doing in the example below... note, that you can not use the action task **Set Int** because the game is listening for Set Gold (Int) which is in direct reference to the players stats for gold, it's weird by that's how it is.



As shown in the above example, we're setting the players stats for gold directly, by subtracting a value which is being held in the Local BB for the encounter, as an int variable. This variable is name the "bribe3MonsterAmnt", and it demonstrated below.



As you can see the variable “*bribe3MonsterAmnt*” is set to 100, as this is the amount of gold associated with the challenge objective and is the amount of gold that should be subtracted from the players stats for gold.

The 2nd action node is where we actually remove the monster cards from the game, this is using the action task **Remove Monster Cards**, this is also demonstrated below.



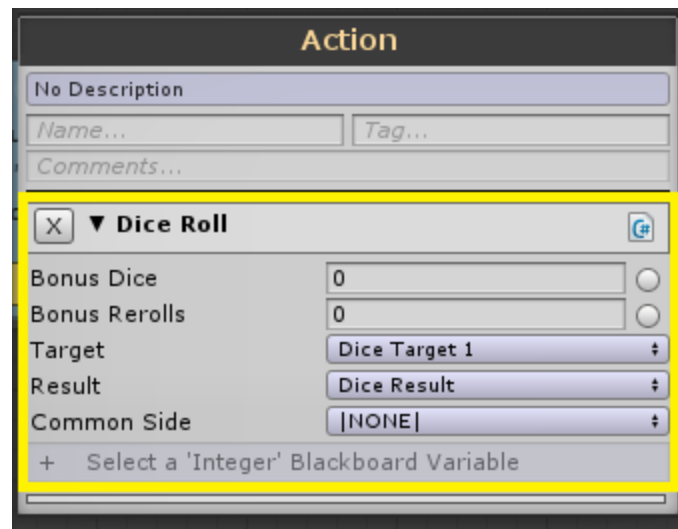
## How to create the other types of mini games in HoF2:

There are other mini games and gambits that you can use in your encounters for HoF2. These include the **Dice Gambit**, **Pendulum**, and **Wheel**. If you need a refresher on what these different mini games are, click [here](#).

## How to create a dice gambit:

Creating a **Dice Gambit** within an encounter behavior tree, is relatively straightforward. Let's run through how to create one now!

The first thing that you'll want to do is create a new action node within your chosen encounter behaviour tree. You'll want to add the action task **Dice Roll** to this action node. Once you've done this you'll have new data to work with, demonstrated below.



**Bonus Dice** is the amount of extra dice that you want players to have in any given dice gambit you create.

**Bonus Rerolls** is the amount of times that you allows the player the ability to reroll their roll.

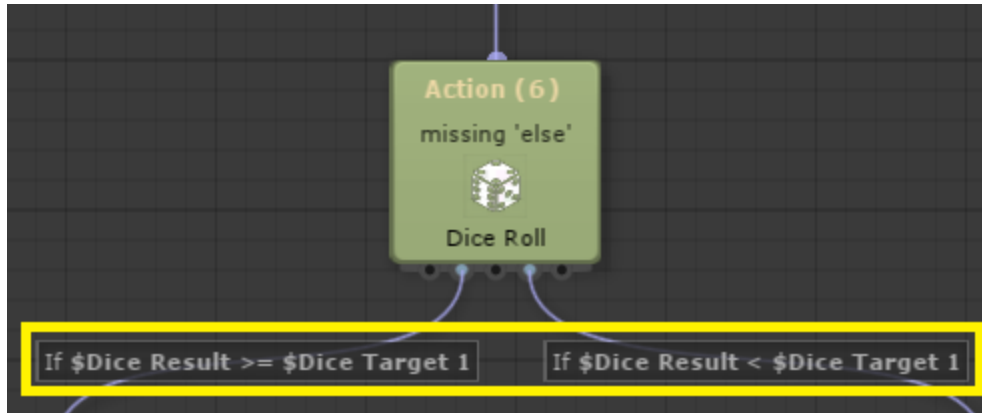
**Target** is the players target to beat in the dice gambit.

**Result** is the result of the players roll in the gambit.

**Common Side** is tracking how many 'common sides' you get (or need) during a dice gambit. For example, you wanted to know if players rolled 3 5's, and that did a specific thing in your gambit.



When you've set all of this up, you will typically have two branches coming out of the bottom of this action node holding your **Dice Roll**. Demonstrated below are two branches which check for the condition **Dice Result**. If the result is greater or equal than the target then it follows the left branch, otherwise if it's less than the target it follows the right branch.

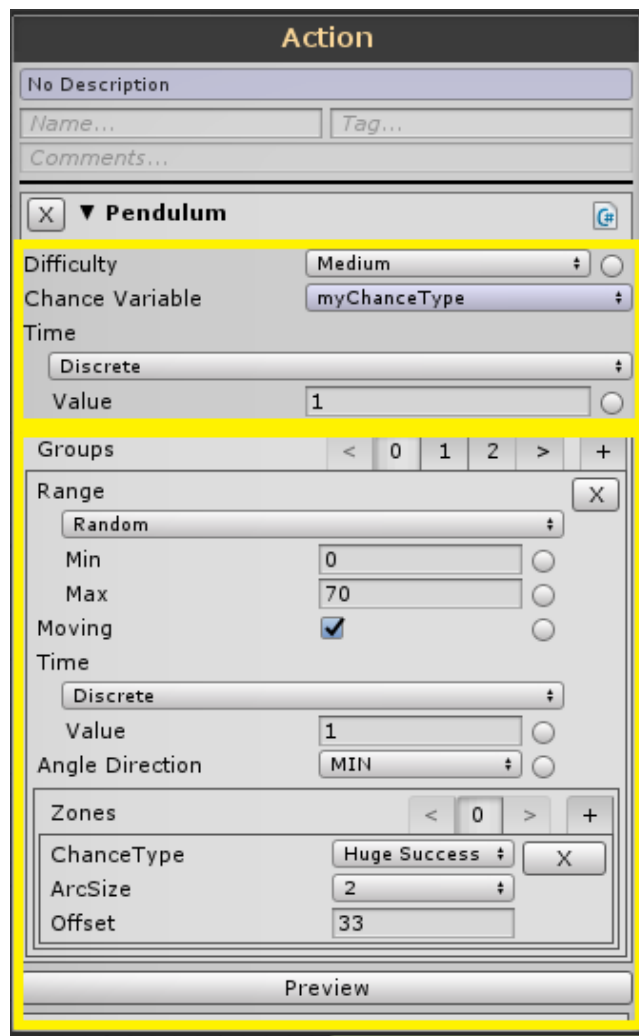


## How to create a pendulum mini game:

Creating a **Pendulum** mini game within an encounter behavior tree can be challenging. Let's run through what it takes to create one.

**Note:** We recommend that you use the data from existing pendulum games by copying their data in to your own encounter. This will provide a good starting point for a working pendulum.

The first thing that you'll want to do is create a new action node within your chosen encounter behaviour tree. You'll want to add the action task **Pendulum** to this action node. Once you've done this you'll have a lot of new data to work with, demonstrated below.



I'll outline in brief, what all of these different data fields correspond to, on the mini game.

**Difficulty** is the rate at which the pendulum swings, speed wise. Remember that **Easy** should only be used for testing purposes.

The **Chance Variable** is set by default, this is common across all of the mini games in HoF2.

**Time** is the time it takes for a pendulum mini game to complete a single swing. Setting the time to **Discrete** means that you only have to set one value for time, which is what you always want. If you set it to **Random** then there are two values that need to be set for time. A random value between these values will be selected each time you try the pendulum. Random was never used in *Pendulums* in HoF2.

**Groups** are the arcing segments that players need to stop the pendulum swing on. Segments can be assigned S, HS, or HF results. Note no Failure segment should ever be set as this is the result for missing the segments entirely.

**Range** is the start and end angle through which the pendulums swing. This takes a min and max value. For example, 0 degrees is where the swing starts, and 70 degrees is where the swing will end, before swinging back in the opposite direction.

**Moving** determines if this *Group* should be stationary or also move as the pendulum swings.

**Time** is the duration that the *Group* will move from it's min to max range.

**Angle Direction** is the starting direction for the *Group* will move, i.e. if it starts moving to the left or right.

**Zones** are where you create your actual segments. These take a **ChanceType**, an **ArcSize**, and an **Offset** value.

The **ChanceType** is the result that you want to be associated with the brick. For example, S, HS, F, or HF. Remember, do not set a *Zone* (segment) to *Failure*.

The **ArcSize** is the size in degrees of the segment.

The **Offset** value is angle in degrees that the *Zones* leftmost edge will be placed from the *Groups Min Range* OR the previous *Zones* rightmost edge. For example, the above image has a value of 33, so that *Zone* would start at 33 degrees from 0 degrees.

## How to create a wheel mini game:

Creating a **Wheel Spin** mini game within an encounter behavior tree. Let's run through the 'Field of Fae' encounter to demonstrate how **Wheel Spin** mini games are put together.

The screenshot shows an encounter behavior tree editor. On the left, there are two action nodes. Action (8) is a large green box containing five 'Get' actions for different card types, each with a specific keyword and a local variable to store the results. Action (9) is a smaller grey box representing the 'Wheel Spin' mini game itself, which takes a source list and a remaining list as input and performs 'No Action' on the chosen and remaining cards. On the right, a 'Variables' panel lists various variables, with several numerical counts (painCount, healthCount, goldCount, blessCount, foodCount) highlighted in yellow, corresponding to the variables used in Action (8).

**Action (8)**

Get **\$painCount Pain** card  
If Card Has Any Keywords:  
Keywords: Pain  
Stored in: **\$wheelPain**

Get **\$healthCount Gain** cards  
If Card Has Any Keywords:  
Keywords: Health  
Stored in: **\$wheelHealth**

Get **\$goldCount Gain** cards  
If Card Has Any Keywords:  
Keywords: Gold  
Stored in: **\$wheelGold**

Get **\$blessCount Blessing** card  
Stored in: **\$wheelBless**

Get **\$foodCount Gain** cards  
If Card Has Any Keywords:  
Keywords: Food  
Stored in: **\$foodList**

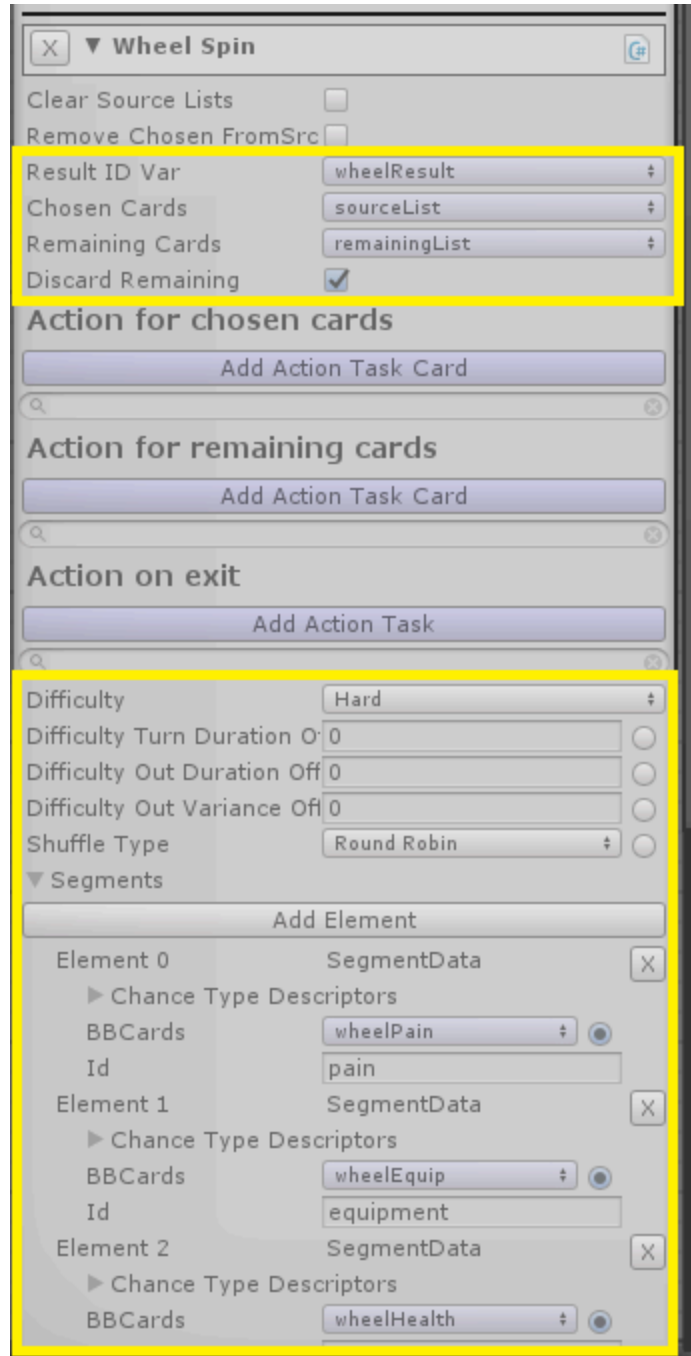
**Variables**

Name	Value
Chance Result	Huge Success
sourceList	List<CardLink> 0
wheelResult	
wheelPain	List<CardLink> 0
wheelEquip	List<CardLink> 0
wheelHealth	List<CardLink> 0
wheelGold	List<CardLink> 0
wheelBless	List<CardLink> 0
painCount	1
healthCount	3
goldCount	2
blessCount	1
remainingList	List<CardLink> 0
XXXfailCountXXX	1
curseList	List<CardLink> 0
foodCount	2
foodList	List<CardLink> 0
PlatCount	3
PlatList	List<CardLink> 0
numHS	1

**Action (9)**

Wheel Spin Task  
into <**\$sourceList**>  
remaining into <**\$remainingList**>  
Do 'No Action'  
on chosen cards  
Do 'No Action'  
on remaining cards  
Do 'No Action'  
on exit

First things first, let's explain the above example to you, a little more clearly. **Action (8)** is getting the cards that will be added into the **Wheel Spin** itself. We can also see the amount of each card that is retrieved, evident via the Local BB variables that have been highlighted to the right. **Action (9)** is the **Wheel Spin** mini game itself, so let's look at how these cards are actually added into the mini game.



**Clear Source Lists** clears any cards that are currently being held in the source list.

**Remove Chosen FromScr** only operates if 'clear source' isn't checked. It will remove the chosen card from its original list, for example, if you have 3 gold cards and you don't have clear source checked, it means that after the minigame your 'goldList' is still going to have the original 3 cards in it. However, if you have remove from source checked, and you choose one of those gold cards, the original 'goldList' will not longer have the chosen card in it but will have the remaining 2. Its useful in cases where you're doing something with the original lists of cards

**Result ID Var** this stores the result of the Wheel Spin mini game, this is created by default when you create the **Wheel Spin** action task.

**Chosen Cards** are the list that the cards chosen by players during a wheel spin are stored into.

**Remaining Cards** are the list that the cards remaining after a wheel spin are stored into.

**Discard Remaining** if this is checked, then the cards that are remaining after a wheel spin are discarded.

**Action for chosen cards** allows you to tie action tasks to those cards that are chosen from a wheel spin.

**Action for remaining cards** allows you to tie action tasks to those cards that are remaining after a wheel spin.

**Action on exit** allows you to tie action tasks after the conclusion of a wheel spin.

**Difficulty** is the rate at which the wheel spins, and the time it takes to slow down after the button press.

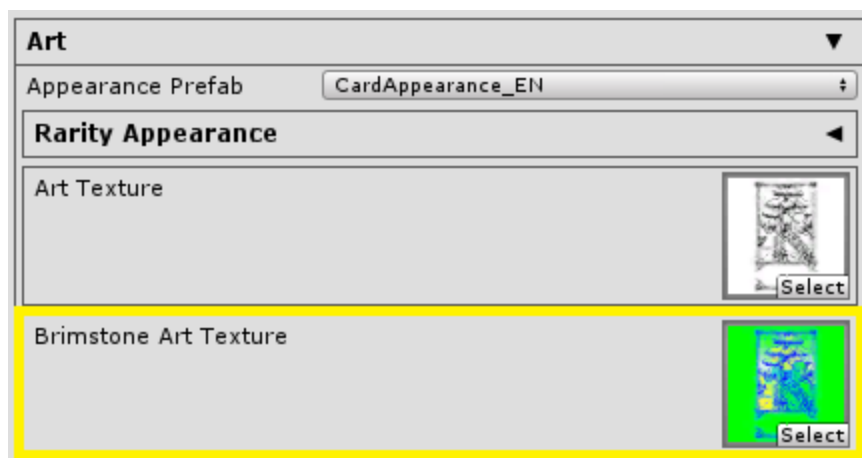
**Shuffle Type** this selects the type of distribution for the different lists of cards used in the wheel spin.

**Segments** are where you create elements that store the card lists that you're drawing from for the wheel spin.

## Creating Brimstone & Platinum Rarity Cards:

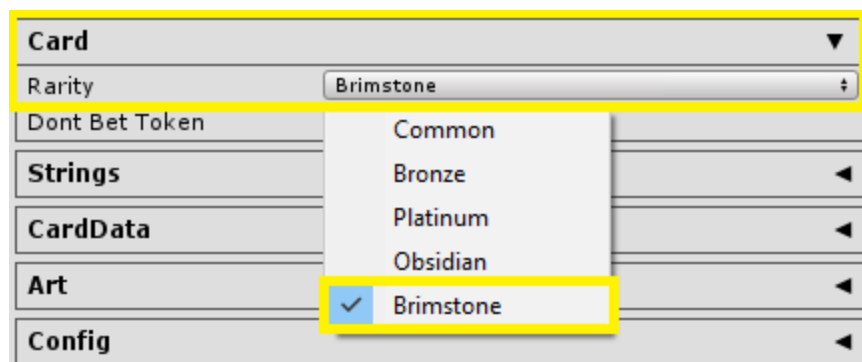
Brimstone & Platinum rarity cards are another feature of HoF2. Let's quickly run through how you can create these, for yourself. Let's start by showing you how to setup the brimstone rarity.

Firstly, let's select the encounter card that you wish to make into your brimstone rarity. Once you've done this enter the **Art** data, where you are able to set your different card art textures. You'll spot a section where you can add a **Brimstone Art Texture**, enter the selector to the right and search for "brimstone" and select an appropriate texture.



**Note:** 'Platinum' art textures are set in the **Art Texture** data, whereas **Brimstone Art Texture** is for setting 'Brimstone' textures only.

Now in order to actually set this card to appear with the brimstone texture you've selected, you'll need to jump up to the **Card** label of this cards data, and set its **Rarity** to 'Brimstone'.

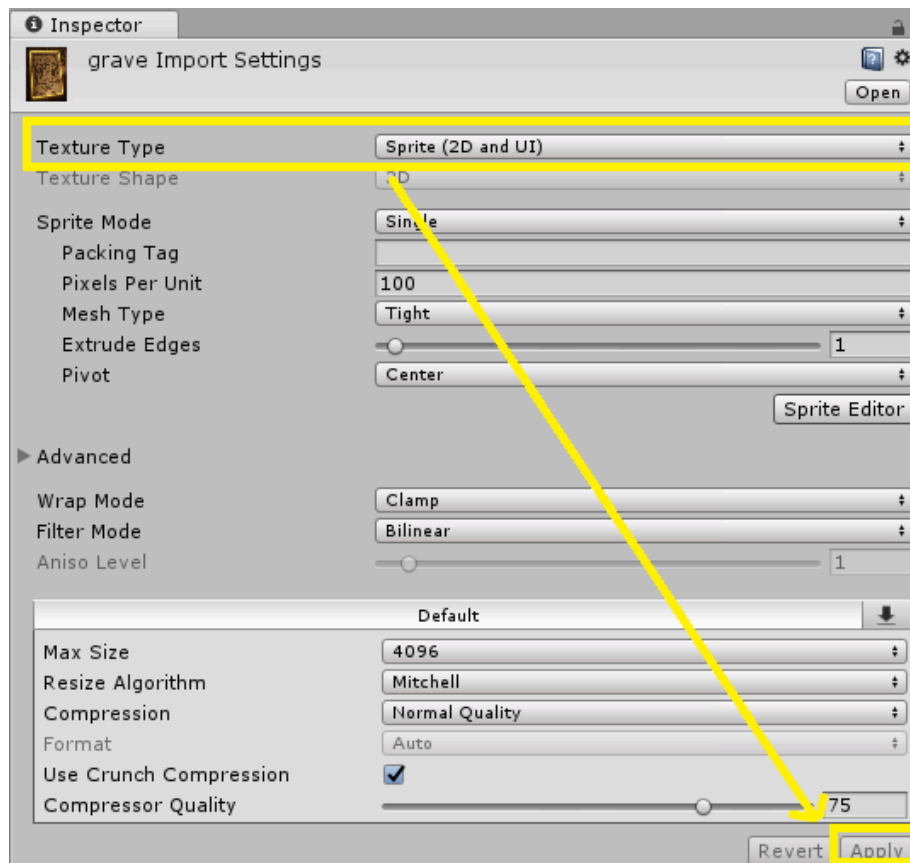


**Note:** If you were setting the **Art Texture** under the **Art** label to be a 'Platinum' texture for your card, you would set the **Rarity** under the **Card** label to be 'Platinum'.

## Adding your own art as textures for your cards:

You can add your own art textures to your card mods by adding the images into your Unity project directly.

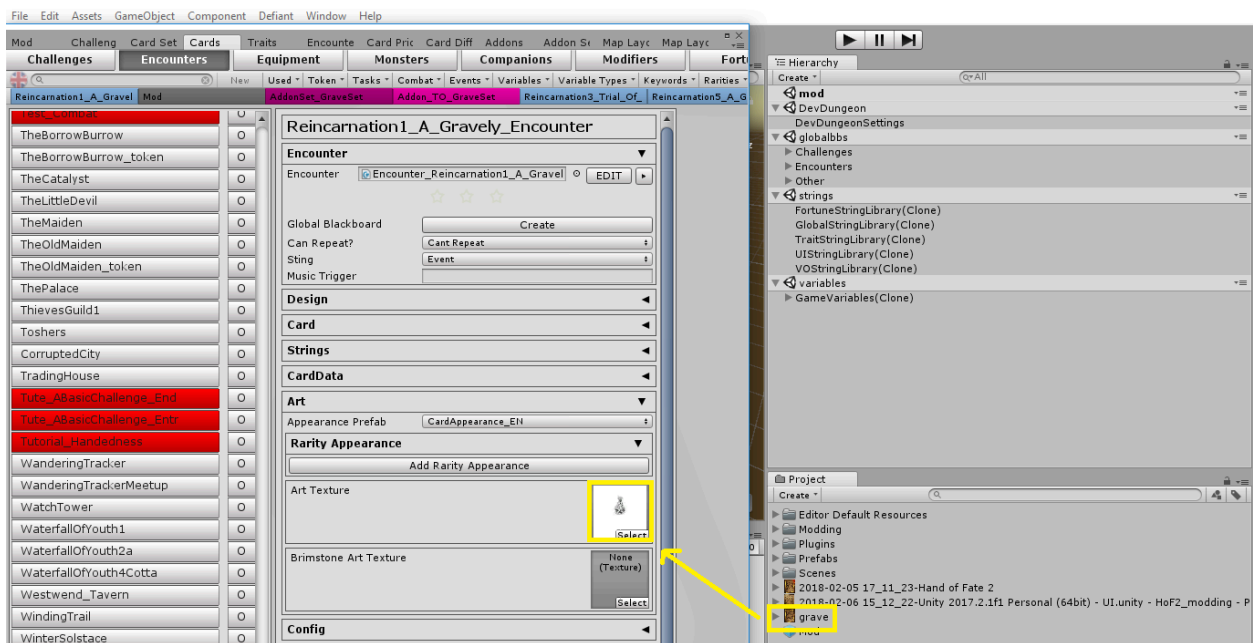
Once your images have been added to the project you have to make sure that they are set up as textures (this will not happen by default). You can do this by selecting the image within the Unity editor and then viewing it in the inspector. Demonstrated below is the process of making an image into a texture through the inspector.



Once you've done this you will be able to use this image (which is now a texture) as an **Art Texture** for your cards. To add this to a card as your **Art Texture**, you need to drag and drop the image from within your Unity project and assign it manually. This is demonstrated below.

**Note:** There are no specific size/format/transparency requirements that you need to be aware of, but know that the current art textures that we have in the game are 1024x1024. Know that anything above or below that requirement will show odd results. Also, while you may not be limited by these requirements, know that you might run into issues running out of cloud storage if your mod gets large though.





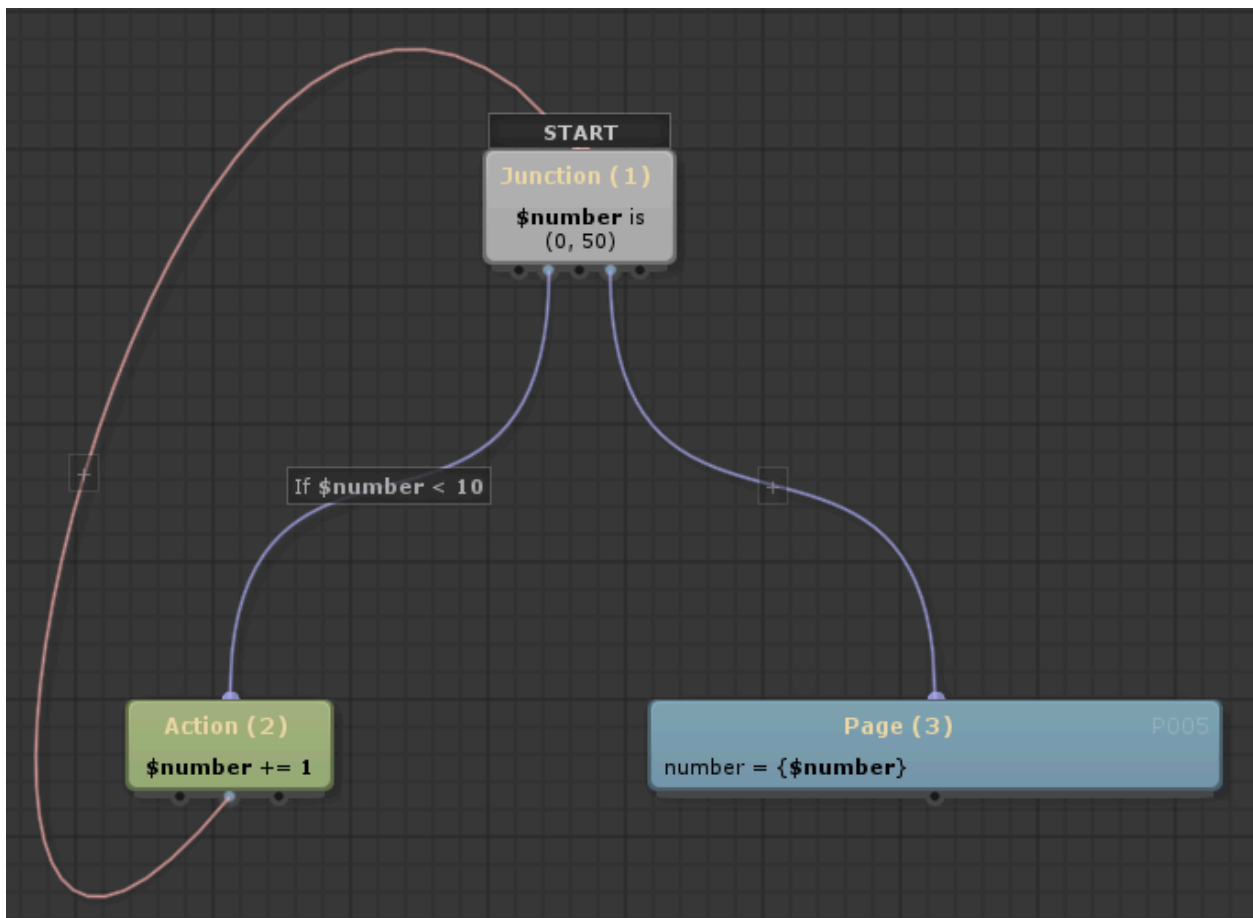
Did you know that there are templates and documentation for creating your own card art?

We've put together documentation showing you how anyone who has access to a photoshop license can use these templates + tools to make custom basic card art, brimstone and platinum card art for their own mods. You can access all of the necessary files [here!](#)

## Creating Loops in your behaviour trees:

Loops are a powerful resource in creating encounter mods for HoF2. You may have been browsing through the existing encounters within HoF2, and have spotted a number of nodes that have connections returning to nodes earlier in the graph, these are loops. Loops are used when you want to run through actions within an encounter behaviour tree more than once.

Creating loops that are made up of page and action nodes are simple enough, but when you're looping junctions and actions together, there are additional steps that you'll need take, otherwise it will break. Let's run through a basic example of a loop between a junction and an action, to demonstrate what you need to do, when you approach looping within your mods.

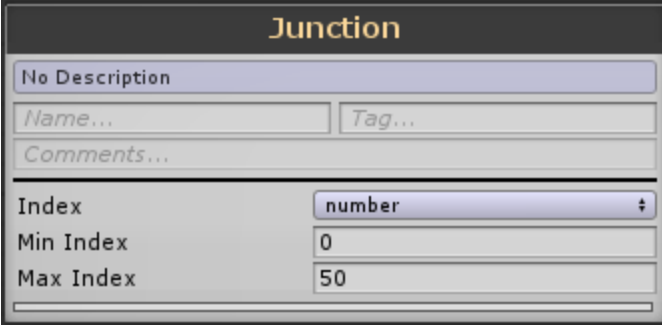


This is a simple example, to demonstrate a point. What we see above is a loop. Specifically, the left branch is where our loop is happening. If the int variable (**number**) is less than 10, then we add 1 to **number**, and then we loop and repeat the condition check and action task.

Now the whole point of this demonstration, is to show you what needs to be done within the Junction node when you're creating loops between Junction and Action nodes (no pages).

A Junction node has 3 data entries, the first is an index, this is what the node checks against. The min and the max, are the boundaries of that index. In our example the int variable in our local BB is named **number** and it's set to 0.

By setting the max index to a value of 50, this loop will not break, as the value of 10 is within the range. If we're to set the condition of (If \$number < 10) to say (If \$number < 1000), then the loop would break, as soon as the index is above the max of 50, which we have set in the junction node.



The image shows a configuration window for a 'Junction' node. The window has a title bar with the word 'Junction' in orange. Below the title bar, there is a section for 'No Description' with a text area. Below that, there are two text input fields labeled 'Name...' and 'Tag...'. Below these is a text area labeled 'Comments...'. The bottom section of the window is a table with three rows: 'Index' with a dropdown menu showing 'number', 'Min Index' with a text input field containing '0', and 'Max Index' with a text input field containing '50'.

Junction	
No Description	
Name...	Tag...
Comments...	
Index	number
Min Index	0
Max Index	50

## Section 7: “Intermediate Challenge Design”

In this section of the documentation we'll be looking at some more intermediate challenge design, that you can use when creating your own challenges in the card editor.

### Creating a Silver token for the basic challenge:

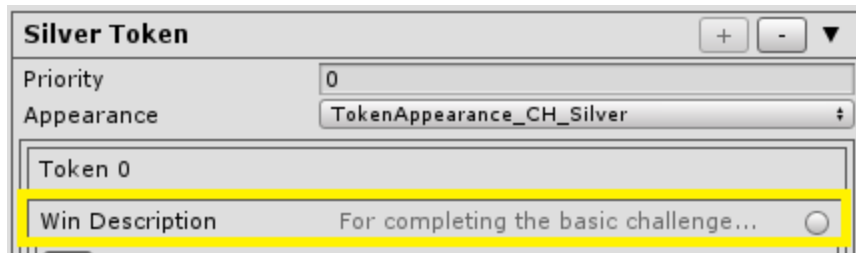
A challenge that warrants a silver token is typically more complex in it's design, and the objectives and steps of the challenge are usually greater and more expansive than that of a challenge that only has a gold token unlock.

Typically a challenge with a silver and gold token on it will have multiple challenge objectives. Our silver token will be unlocked if the player completes the basic challenge. Where as our Gold token is already set up to only unlock if the player has collected 100 gold before entering the **End Card** encounter for the basic challenge.

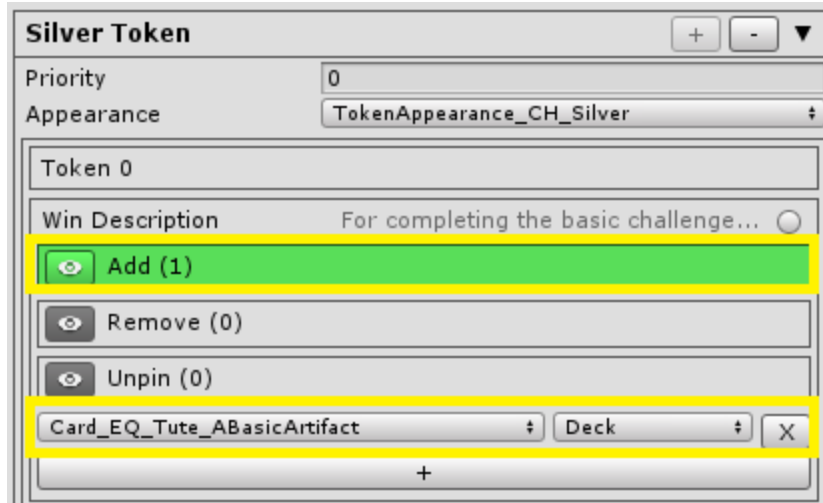
Let's start by creating our new silver challenge token, demonstrated below:



We then need to set the **Win Description** for the token. This is demonstrated below:



We then need to **Add** some cards to be unlocked and added to the players deck. This is demonstrated below:

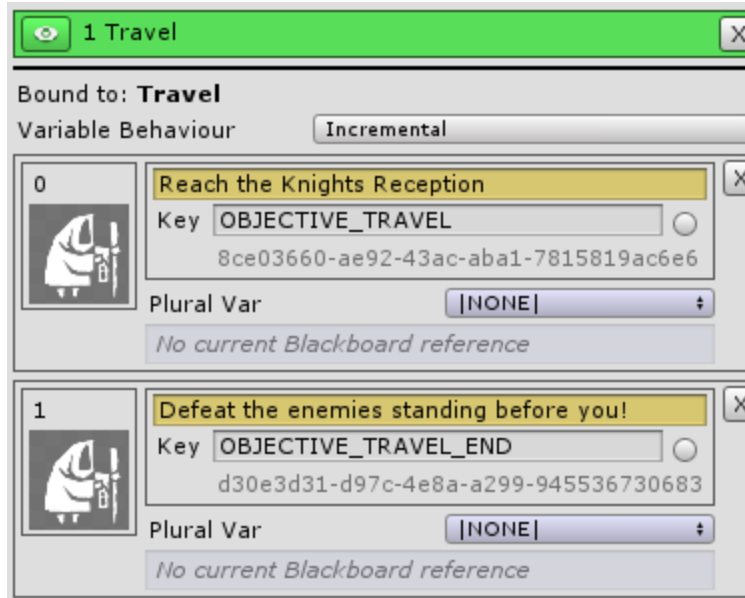


## Creating the second challenge objective, linked to the silver token unlock for the basic challenge:

As we explained above, our Silver token will be unlocked if the player completes the basic challenge (reaches the end encounter, and does not die). Let's look at setting that up now.

Jump into the challenge card for the Basic Challenge. Navigate to the **Objective** data, and we'll be creating a new objective, so go ahead and click **Add Objective Channel**. Let's name this channel "*1 Travel*". Let's now expand this channel and look at what we'll need to set.

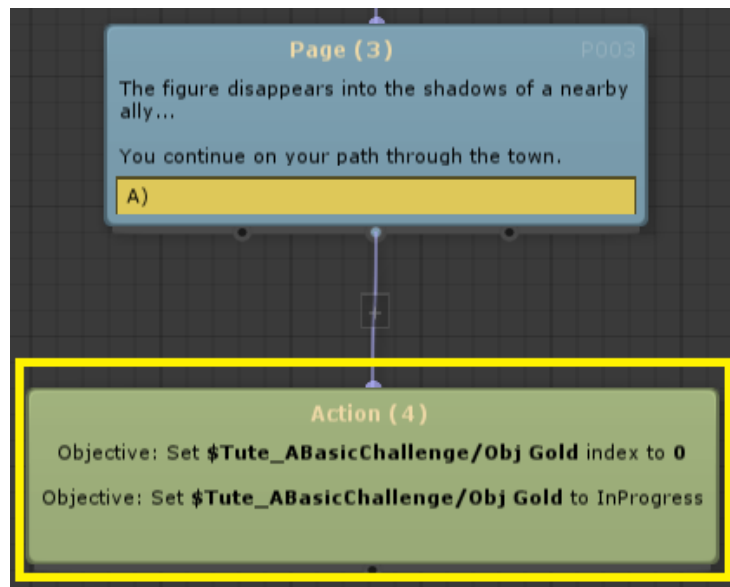
As you'll see below, we need to create 2 data sections, these will have the **Index** values **0** and **1**. **Index 1** will hold the description "*Reach the Knights Reception*". **Index 2** will hold the description "*Defeat the enemies standing before you!*". This is demonstrated in the below screen capture:



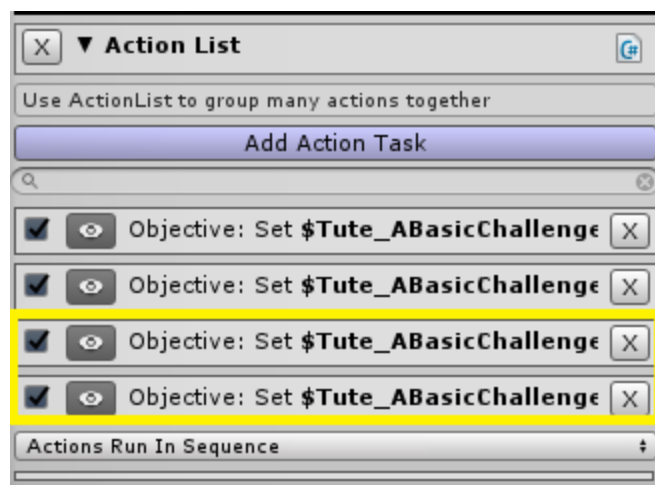
That's all our second objective is going to be, pretty simple right! Now we have to jump back into the **Entrance Card** for the Basic Challenge, this was named "*Tute\_ABasicChallenge\_Entr*", and we are going to add some new action tasks to the action node we created in the Basic Challenge Tutorial, earlier in this documentation, where we were incrementing the index value of a challenge objective from -1 to 0, and where we were setting the status of an objective to '**In Progress**'. Let's go ahead and do that next.

Adding the 'Objective Channel Index' & 'Objective State' action tasks to the end of the Entrance Card, for the second challenge objective:

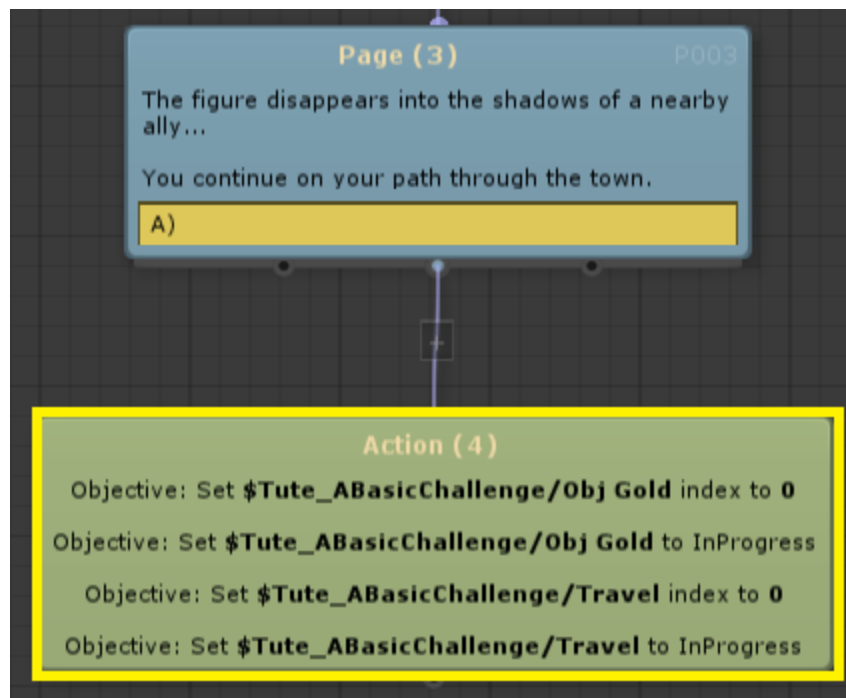
Once we're back in the encounter behaviour tree for the "Tute\_ABasicChallenge\_Entr" encounter card. Let's find that action node that we created before. It should look like the screen capture provided below:



Now let's select this node and add a new action task **Set Objective Index**, for incrementing the objective index of our **Travel** challenge objective to 0. Now add the **Set Objective Status** of our **Travel** challenge objective to '**In Progress**'. Demonstrated below:



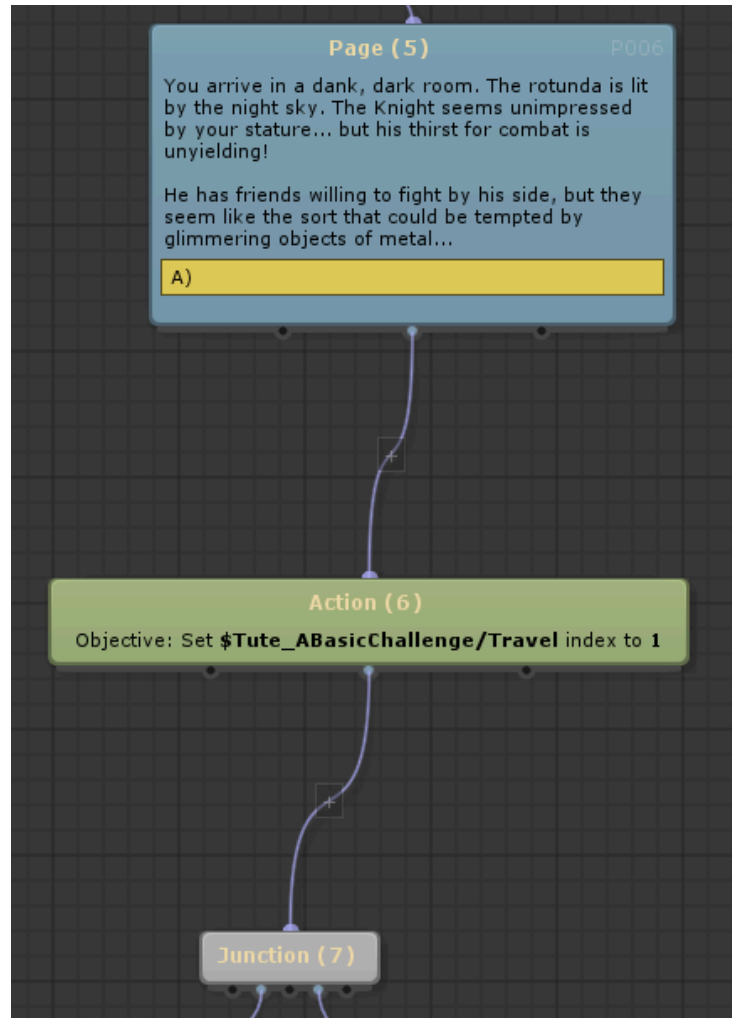
Your action node at the end of the encounter behaviour tree for the **Entrance Card** should now look like the provided screen capture.





## Adding the ‘Objective Channel Index’ action task to the End Card of the basic challenge, for the second challenge objective:

Let’s quickly jump back in the encounter behaviour tree for the “*Tute\_ABasicChallenge\_End*” encounter card. Let’s find the first page of text, that is immediately followed by a junction node. It should look like the screen capture below, minus the action node in the middle, this is what you’re going to add now.

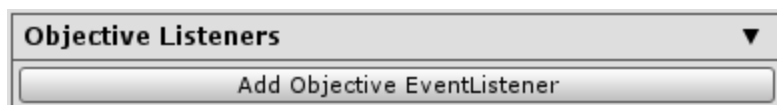


Let’s create the action node and select it. Let’s add the action task **Set Objective Index**. This time we want to set the objective index for our **Travel** challenge objective to 1, this will change the UI in the top right to read our second description string, which was “*Defeat the enemies standing before you!*”. Your graph should now look like the above screen capture.

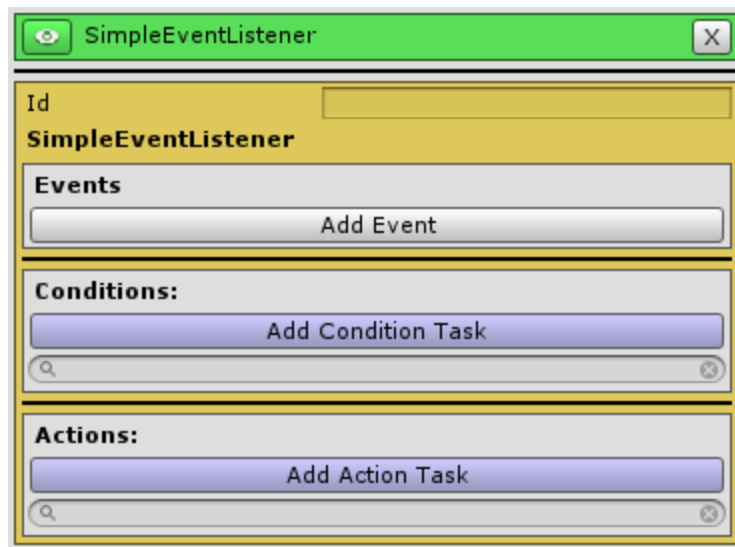
## Creating an objective listener for the gold challenge objective:

**Objective Listeners** are another piece of data that we can create on our challenge card. Let's create an objective listener for our gold acquired throughout a challenge run. This will update the the UI in the top right with a strike through, if the players gold count is greater than or equal to the "goldRequired" (100). This will also update the UI if the player drops below the "goldRequired" amount during a run. It's a nice bit of polish that we can add to our challenge.

Let's start by creating the objective listener on the challenge card. Select the data named **Objective Listener** and expand its contents. Now, create a **Simple Event Listener** by clicking the **Add Objective EventListener** button.

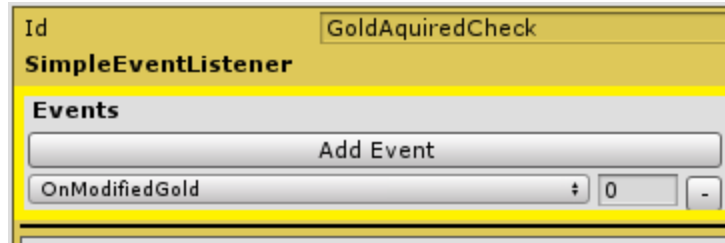


Once you've created your new **Simple Event Listener**, expand its contents and take a look inside. Demonstrated below:

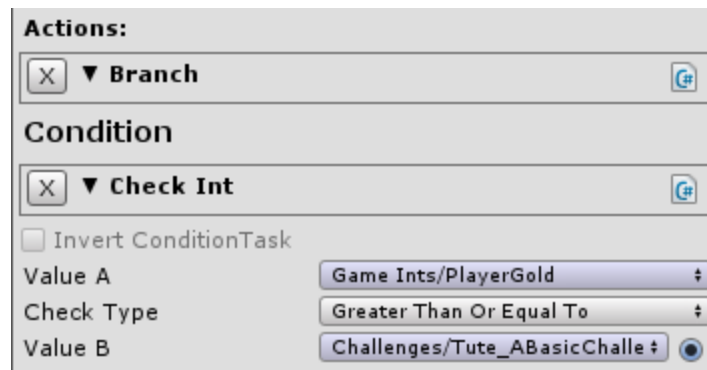


Let's start by renaming the **Simple Event Listener** to "*GoldAquiredCheck*". You can do this by entering the string into the section labelled **ID**.

Next let's click **Add Event** this is the event that the **Simple Event Listener** is going to be listening out for, and when it does, the actions we set will initiate (we'll add these next). Let's click **Add Event** and add the event named **OnModifiedGold**. Demonstrated below:




Next, let's set the actions that we want to take place whenever the players amount of gold is modified. Let's click **Add Action Task** under the section **Actions**. Next, let's add the action task **Branch**. After we've added our **Branch**, this will give us some data to fill in. We have a **Condition** that we need to set up, let's make our **Condition** a **Check Int**. **Check Int** takes two **Values**. Let's set **Value A** to be **Game Ints > PlayerGold**. Then set the **Check Type** to be **Greater than or equal to**. Next, set **Value B** (the value that it checks against) to be **Challenges > Tute\_ABasicChallenge > goldRequired**. Demonstrated below:



Next we have to set the data for if this condition is true, then do a thing. Let's add the action task **Set Objective State** under the section labelled **True**, let's set the **Objective Channel** to be **Challenges > Tute\_ABasicChallenge > Obj Gold**, now let's set the state for true to be **'Complete'**.


Next we'll do the same thing for for the section labelled **False**, but instead of the objective state being set to **'Complete'**, let's set it to **'In Progress'**. This is all demonstrated in the screen capture provided below:

**Actions:**

X ▼ **Branch** 

---

**Condition**

X ▼ **Check Int** 


Invert ConditionTask

Value A

Check Type

Value B


**True**

X ▼ **Set Objective State** 

Objective Channel

State

**False**

X ▼ **Set Objective State** 

Objective Channel

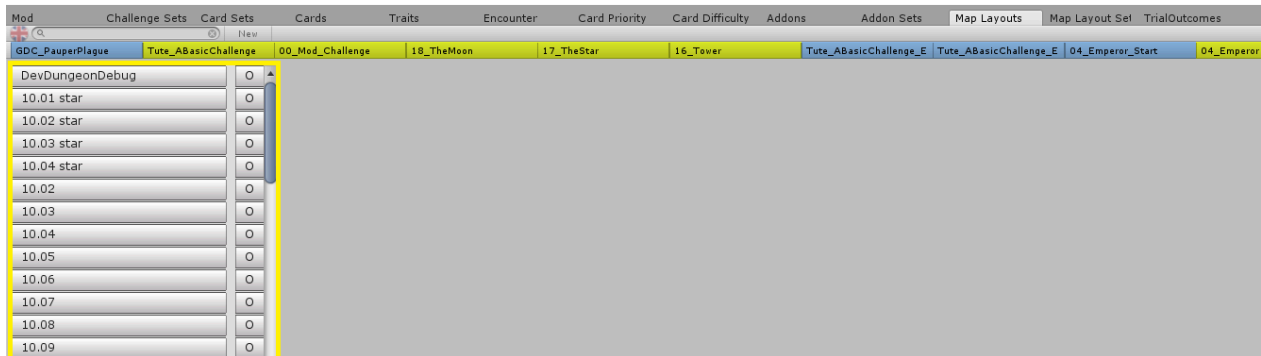
State

## Creating your own dungeon map layouts:

You can create your own dungeon map layouts using the HoF2 modding tools. When you are navigating the card editor, you will spot a tab named **Map Layouts**.

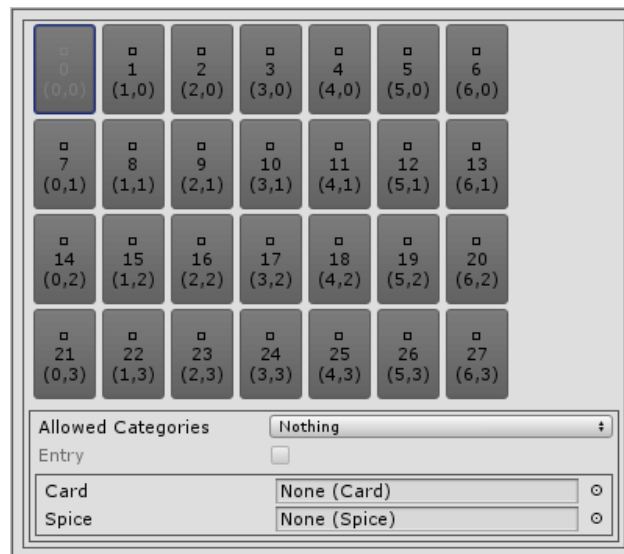


After selecting this tab, you'll have a long list of existing dungeon map layouts to the left of the card editor window.

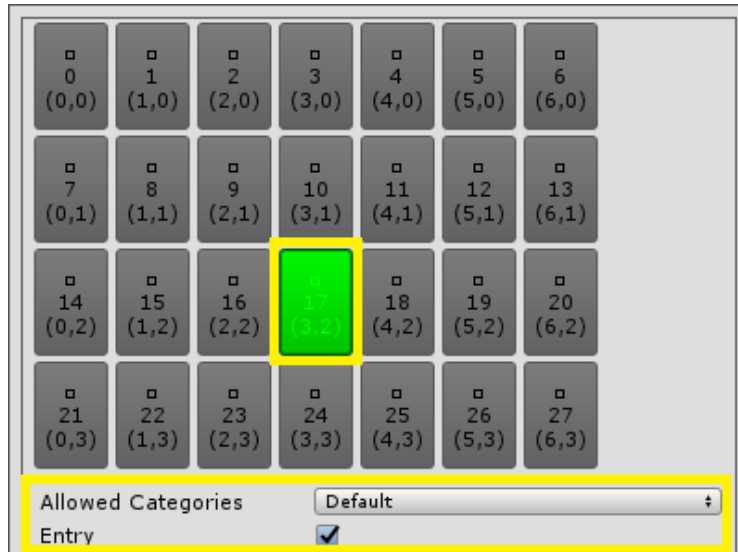


Using the same method for creating new cards for encounters, challenges, and equipment, you can create your own dungeon map layouts. Let's quickly show you how you can do this now. Enter the name of your dungeon map layout that you wish to make, and click **New**.

After you've done this, navigate the list on the left and select your newly created dungeon map layout. Your dungeon map layout should look like the screen capture below by default.

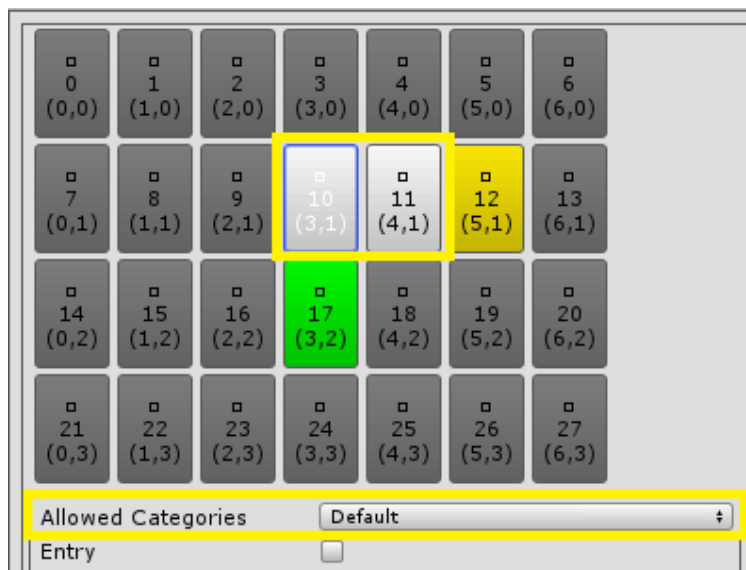


If you select any of the cards on this map layout, you can set categories to them. For example, I'll select a card in the middle of the map and set its **Allowed Categories** to be **Default**. I'll then set this card to also be my **Entry**. This is demonstrated in the below screen capture.

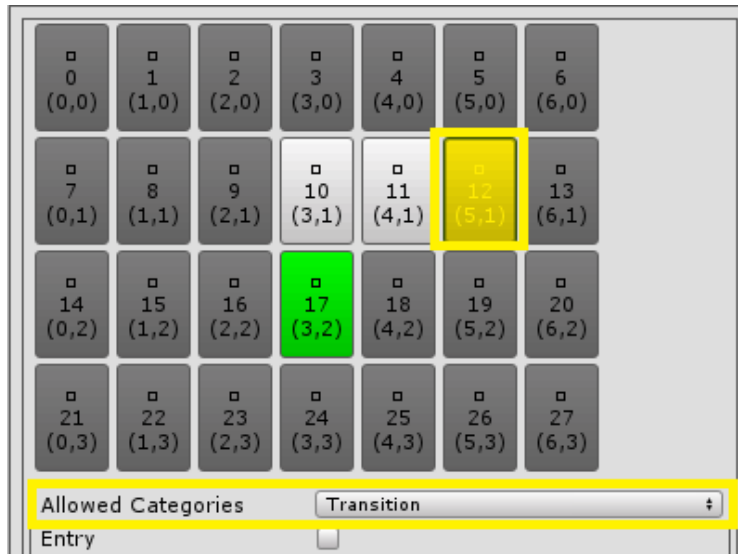


Setting a slot on your map layout to **Entry** identifies that card as the start of the level. You cannot set multiple entries on a map layout.

You'll see below that I've made a map layout with a right hand turn. The 2nd and 3rd slots of this map have the **Allowed Categories** set to **Default**. Demonstrated below.



Then we have the last card slot, which is coloured yellow, and has the **Allowed Categories** set to **Transition**. This marks that card slot as the transition point of the dungeon map layout, this is where the **Level Exit Cards** will be set. Demonstrated Below.

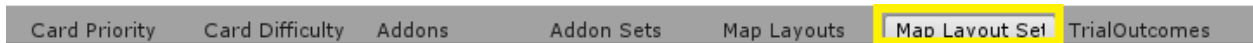


Setting a slot on your map layout to **Transition** identifies that card as a possible end for that level. It is possible to set multiple card slots on a dungeon map layout with the category **Transition**, doing so means that the system will choose one of these marked slots at random for the transition onto the next level in the map layout set.

## Creating your own dungeon map layout sets:

When you're creating your own dungeon map layouts for a challenge mod, you'll have to create what's called a map layout set. This is fairly straightforward, and its name gives you the general gist of things... its a set of map layouts that are used throughout a challenge.

You can create your own map layout sets using the HoF2 modding tools. When you are navigating the card editor, you will spot a tab named **Map Layout Sets**.



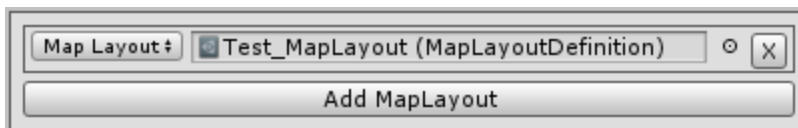
After selecting this tab, you'll have a long list of existing map layout sets to the left of the card editor window.

Using the same method for creating new cards for encounters, challenges, and equipment, you can create your own map layout sets. Let's quickly show you how you can do this now. Enter the name of the map layout set that you wish to make, and click **New**.

After you've done this, navigate the list on the left and select your newly created map layout set. Once you've selected the set, you'll have a large red button named **Add Map Layout**.



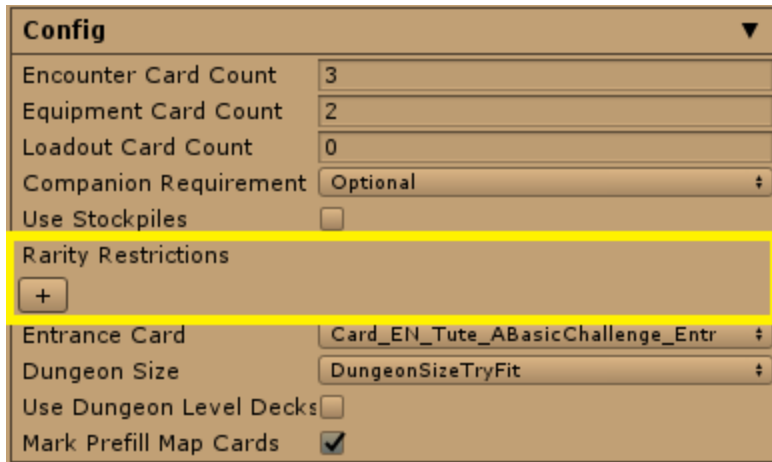
Clicking this button opens a drop down list, and you can search for the dungeon map layouts that you've created using the tools. For example, the map layout that was just demonstrated was named "*Test\_MapLayout*", so this is what that set would look like if I added that dungeon map layout.





## Restricting Cards based on Rarity:

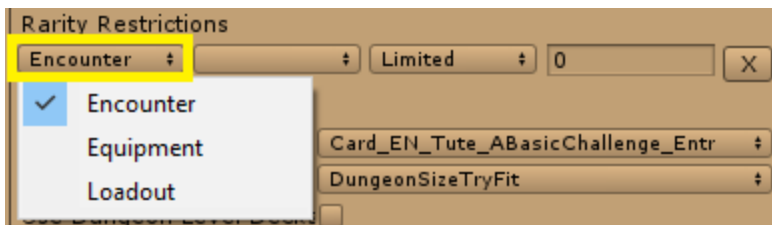
When you're creating a challenge mod, you have the ability to restrict certain cards based on their **Rarity**. We'll look at how we can restrict the amount of **Brimstone & Platinum** cards that a player bring into your challenge now.



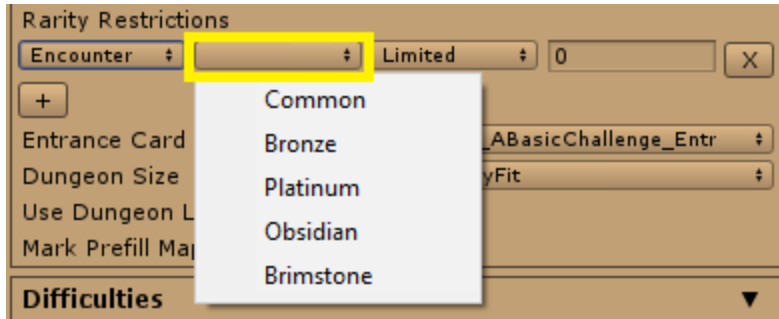
Firstly, you'll need to create some new data within **Rarity Restrictions**. To do this select the addition button highlighted in the below screen capture.



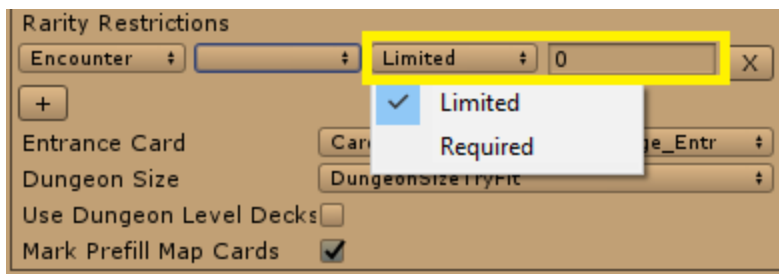
After you've created the new data, there is some data that we need to set for our **Rarity Restriction**. We can set which deck the restriction applies to. For example, below we have the **Encounter** deck selected for the restriction, this means that when the player is in the deckbuilder, the restriction that we'll be applying will be in relation to the Encounter deck.



Next, we can set the type of rarity that should be restricted from the selected deck. We can either place restrictions on **Brimstone** or **Platinum** cards.



We then have two options for our restriction type. We can state that the restriction applies to the player having to include this rarity type in their deck building phase, for this we'd set this to be **Required**. Otherwise, if we wanted to restrict the player from adding any cards from the encounter deck with our selected rarity, then we'd set this to be **Limited**. You would then set the amount of cards for which the reaction applies, but if you're wanting players to be prohibited from bringing them into the challenge, you'd leave this at 0. Demonstrated below.

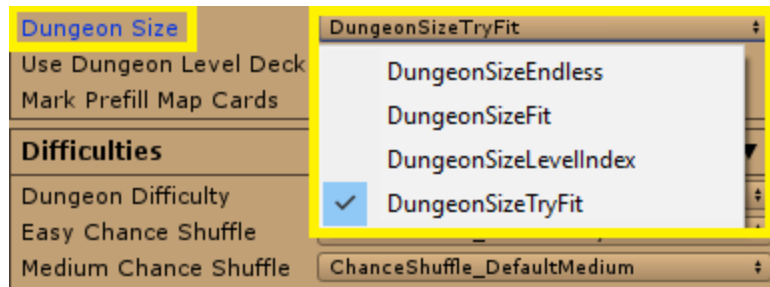


Here is an example of how all of this data translates into the game:



## Other dungeon config data:

There is data found within the **Config** of the Dungeon created for a challenge card. This data is called **Dungeon Size**. This data is demonstrated in the below screen capture.



**DungeonSizeEndless** is the option that you'd select if you wanted your challenge to be used in Endless mode, it's quite literal with its title, the dungeon's size is Endless.

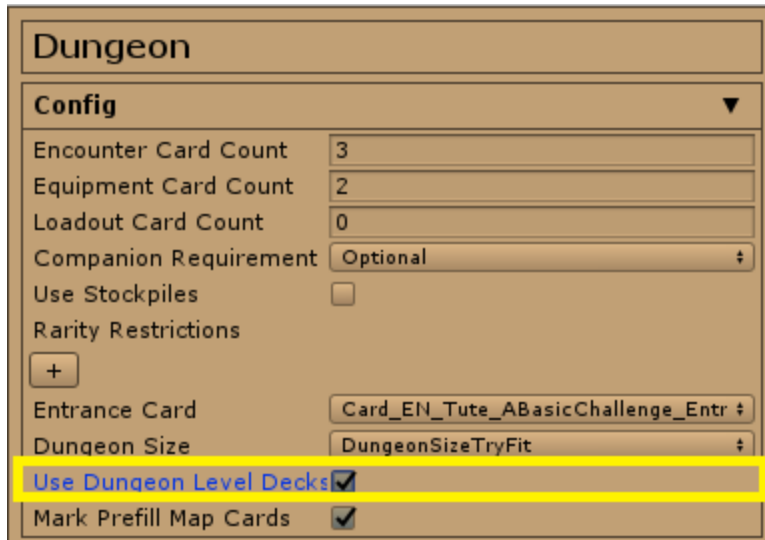
**DungeonSizeFit** is the option that you will select if you have a map layout that has the perfect amount of cards to the amount of slots available on the level.

**DungeonSizeLevelIndex** is the option you will select if you want to have levels play in a particular order for your challenge, you set each level up with an index value and then they will play through them in order of the indices.

When using the **DungeonSizeTryFit** you need enough cards to fill the slots, but you could have more. In this case, some of the cards you take along wont appear.


## Using Dungeon Level Decks:

**Use Dungeon Level Decks** is another check box that you'll spot within the **Config** data for the Dungeon you've created.



The image shows a 'Dungeon' configuration window. Under the 'Config' section, several settings are visible: Encounter Card Count (3), Equipment Card Count (2), Loadout Card Count (0), Companion Requirement (Optional), Use Stockpiles (unchecked), and Rarity Restrictions (+). Below these are dropdown menus for Entrance Card (Card\_EN\_Tute\_ABasicChallenge\_Entr) and Dungeon Size (DungeonSizeTryFit). The 'Use Dungeon Level Decks' checkbox is checked and highlighted with a yellow box. The 'Mark Prefill Map Cards' checkbox is also checked.

When you check this box, your dungeon data will expand to the right of the card editor, where you'll now find a section named **Dungeon Level Decks**.



The image shows a 'Dungeon Level Decks' panel. It has a title bar and a large empty area below it. A button labeled 'Add' is positioned at the top of the main area.

What can I do with **Dungeon Level Decks**? These allow you to assign specific cards to the floors/ levels of your dungeon map layouts.

**For example**, remember back to when we created the Challenge Decks for our basic challenge, and we created 2 card sets, one was the *“DealerMarkedCards”* and the other was the *“ShoppeCards”*. Another way that we could've distributed the *“ShoppeCards”* is via the **Dungeon Level Decks**, if we wanted for example, a shop to always be on each floor/ level of our dungeon.

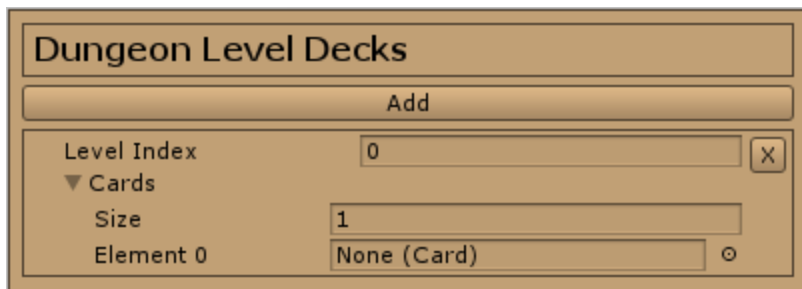
To do this we would start by creating a new **Dungeon Level Decks** by clicking the button **Add** located at the top of this section. This will create some new data, demonstrated in the screen capture below:



The screenshot shows a form titled "Dungeon Level Decks". At the top is an "Add" button. Below it, there is a "Level Index" field with the value "0" and a close button (X). Underneath is a collapsed section labeled "Cards" with a downward arrow. Below the "Cards" section is a "Size" field with the value "0".

The **Level Index** is how we identify the floor/ level that we want these **Cards** to be present on. For example, a **Level Index** of 0 indicates the first floor/ level of the dungeon, and so on and so forth.

**Cards** is where you assign the specific cards that you want to be on the given floor/ level of your dungeon. Incrementing the **Size** to a value of 1, will allow you to enter 1 card into the list, this is demonstrated below:



The screenshot shows the same "Dungeon Level Decks" form. The "Level Index" field still has "0". The "Cards" section is now expanded, showing a "Size" field with the value "1". Below the "Size" field is an "Element 0" field with the value "None (Card)" and a close button (X).

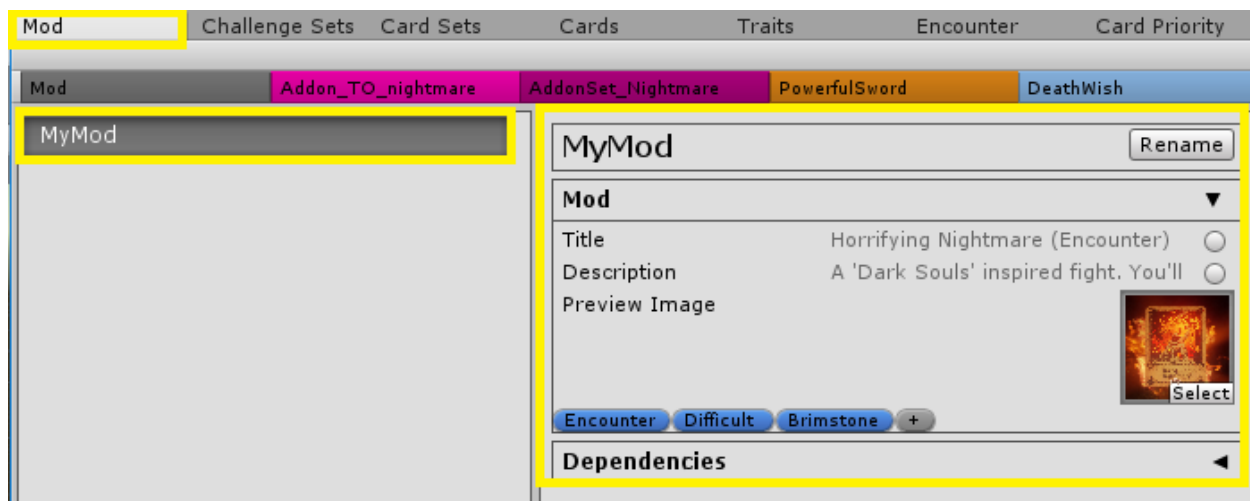
The newly created **Element 0** is where we would store our shop card, meaning that there would always be a shop on the opening floor/ level of our dungeon.

## Section 8: *Distribution of Content*

After you've finished creating the cards that are used in your mod, it's time to look at publishing your work to the Steam Workshop. Let's look at creating the **Mod** itself, the **Addon** that holds the cards you've made for your modded content, and the **Addon Set** that stores your **Addon**.

### Creating the Mod:

First things first, let's navigate to the **Mod** tab of the card editor.



After navigating to the **Mod** tab of the card editor, let's take a look at your mod (by default this is named **MyMod**). You'll be able to rename your mod here, know that renaming your mod does not change the name of the mod visible in the Steam Workshop, this is only for local use.

Let's look at how you can change the **Title** of your mod along with the **Description**, **Preview Image**, and **Tags** which are all displayed on the Steam Workshop when you **Build & Publish**.

**Title** is the name of your mod when it is searched for and viewed in the Steam Workshop.

**Description** is the description field of your mod when it is viewed in the Steam Workshop. This can be changed from within Steam too.

**Preview Image** is the image that is displayed with your mod on the Steam Workshop for HoF2. Know that this preview image has a max size of 1MB and if you try to **Build & Publish** with an image of a greater size than this limit, you will run into errors in Unity.

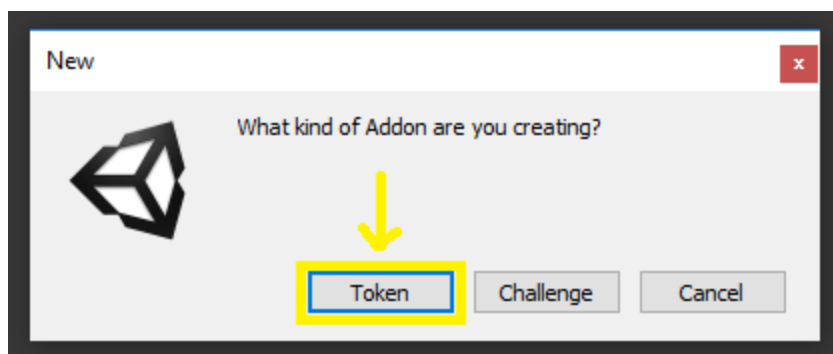
**Tags** are displayed alongside your mods in the Steam Workshop and are identifiers that users can search for so that they find content relative to the given tag ie. Encounter, Challenge, etc...

## Creating Addons & Addon Sets:

Within the card editor you can create an **Addon**. Creating an **Addon** is how you group together the card(s) used in your mod. These can include Encounters, Equipment, Modifiers, Challenges and so on.

For any mods that are just single encounters or encounter sets, you'll have to create a **Token Addon**. Let's look at creating a **Token Addon** for an Encounter/Equipment/Modifier card:

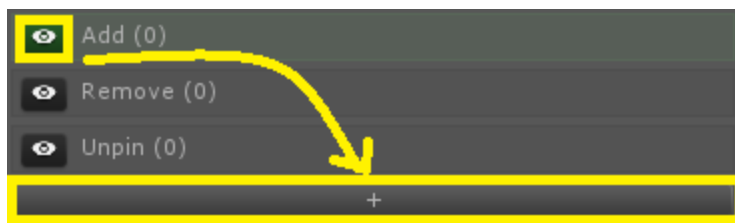
1. Go to the Addon tab of the card editor.
2. Create a new Addon and make sure you select 'Token' when prompted.



3. Add a token by pressing the [+] next to the Token section of the Addon.



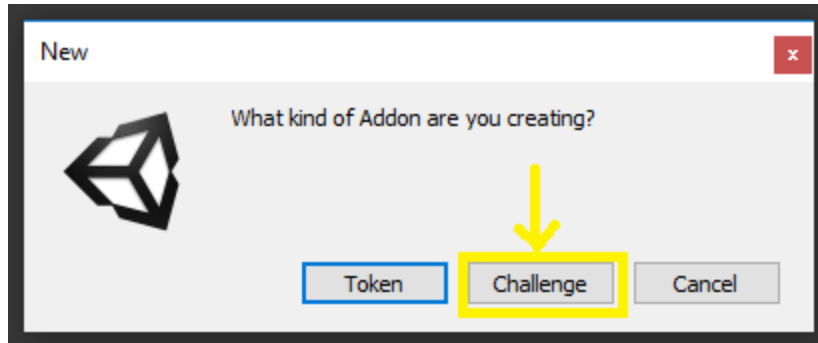
4. Select the [Eye] icon next to Add (0) within this Token section.
5. Press the [+] button to add a card.



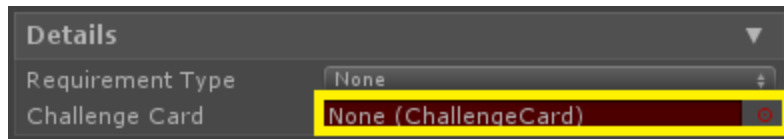
6. Select your mod card from the drop down list.
7. Assuming it is an encounter card, you want to make sure the second drop down is set to Deck (this will add the card to the player's default deck in game).

For any mods that are challenges, you'll have to create a **Challenge Addon**. Let's look at creating a **Challenge Addon** for a challenge mod:

1. Go to the Addon tab of the card editor.
2. Create a new Addon and make sure you select 'Challenge' when prompted.

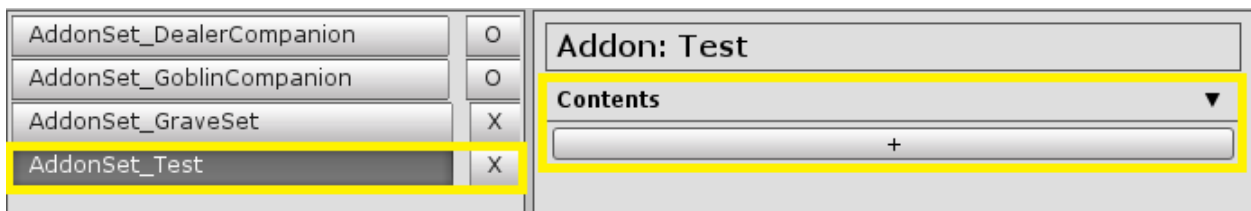


3. Expand the Details section and assign your 'Challenge Card' where it asks.



After you've created the correct Addon for you, you'll need to create the **Addon Set** which references the **Addon** you've just made:

1. Go to the Addon Sets tab of the card editor.
2. Create a new Addon Set.
3. Expand the Contents section and assign your Addon from the drop down list.



Nice! You've made your **Addon & Addon Set** for your mod and are now ready to return to the **Mod Editor** window, where you can **Build & Publish** your content!

**Note:** When you're making a Token Addon, always make sure you set the Win Description string for when players unlock your modded content in game!



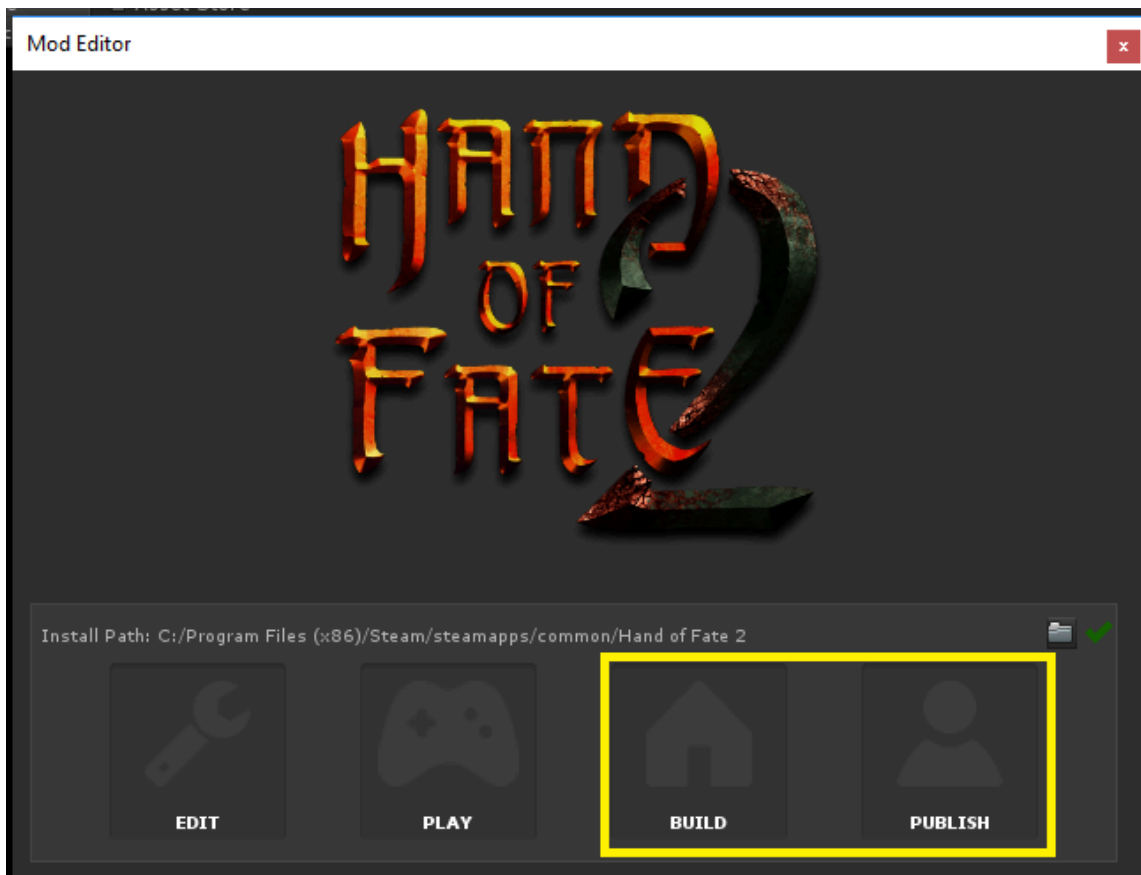
## How to Build & Publish mods to the Steam Workshop:

Now we get to publishing your content! With all of the modded content that you've created through these mod tools, you may want to actually release some of it to the public and you totally should do that! We may even give it a run!

After you've returned to the **Mod Editor** window in Unity, we'll focus our gaze on the 2 buttons we've yet to touch throughout this documentation, the **Build & Publish** buttons.

These buttons do pretty much what they say. **Build** is what you'll need to do in order to even click the **Publish** button, but what does clicking **Build** do? Clicking **Build** tells the system to build all of your content together into one package which is then the mod you'll publish to the Steam Workshop.

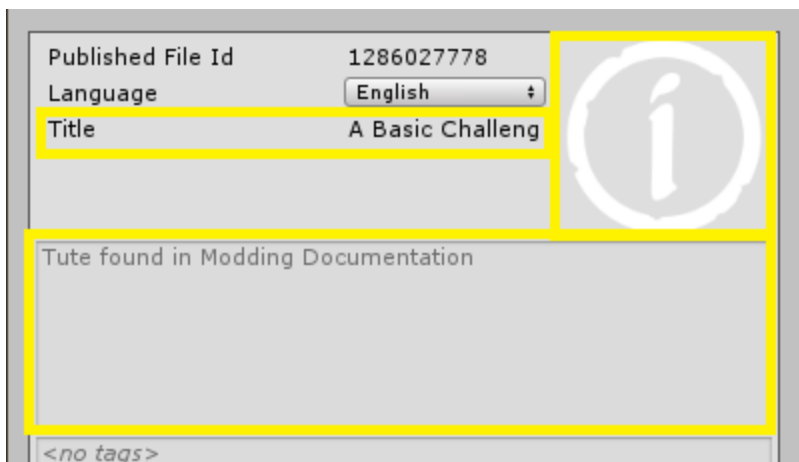
To give you a bit more context, this is packaging your Cards, Addons, Addon Sets, etc... into that **Mod** that we set a **Title**, **Description**, **Preview Image**, and **Tags** for. This final product is then the Asset Bundle which you're publishing.





What does clicking **Publish** do? It grabs that Asset Bundle that you built and it uploads it to the Steam Workshop for HoF2. After you click **Publish**, the **Publish Mod** window will appear.



The main thing that you need to do here is edit the **Visibility** of your mod being published to the workshop (by default this is set to **Private**). The areas highlighted below are filled with the data you entered for **Title**, **Description**, **Preview Image**, and **Tags**.



All you need to do now is click **Upload**. This will display a progress bar at the bottom of the **Publish Mod** window, this is the progress of your upload to the Steam Workshop.

Published File Id	1286027778	
Language	English	
Title	A Basic Challenge	
Tute found in Modding Documentation		
<no tags>		
Visibility	Public	 <b>UPLOAD</b>
Change notes...		

## How to update a mod that's already published on the Steam Workshop:

After making edits to an existing mod that you've already published to the Steam Workshop, you're going to want to update the current submission with your changes. To do this, follow these steps:

1. Enter the Card Editor, Make your Edits, Save, and return to the **Mod Editor** window.
2. Click **'Build'** from the **Mod Editor** window.
3. Once able, Click **'Publish'** from the **Mod Editor** window, and click **Upload**.
4. Now when your mod is viewed in the Steam Workshop, it should say updated and give you time stamp and date. (picture below)



# Section 9: HoF2 SDK - The Game Master's Toolkit

## FAQ

---

### How do I navigate Encounter Behaviour Trees in Unity?

This is one of the fun/frustrating idiosyncrasies of the Unity Engine UI, Holding middle mouse and drag should help you navigate the encounter tree.

### Is it possible to record voice lines and use them in a mod?

While you *could* import your own WAV File (.wav) and use it through a mod, it's probably not going to be beneficial to your mods file size and overall quality... There are a lot of steps we take when we are implementing the VO for HoF2. These include steps such as animation, subtitle considerations, and translations. We recommend that you use our existing VO clips available within the SDK toolkit, as these have been setup accordingly. Hope this helps!

### What are those multi colored tabs at the top of the card editor?

These different tabs are actually the card editors history, these are **not** your open cards. Clicking on a tab will navigate you quickly to that card, instead of you having to search through long lists.

### How do I create Addons & Addon Sets?

We've put together instructional documentation that walks you through the steps of creating the right Addons for your modded content, and also creating the Addon Sets. You can find steps for this [here](#)!

### How do I create a second mod?

So you've created your first mod and it's live on the Steam Workshop, nice one! If you want to start making a second mod you'll have to extract the project within the SDK and use that new project as your second mod. You cannot have multiple mods in the same project. You can find steps for this [here](#)!

### How do I create monster cards with custom values?

While you *can* create your own monster cards using the Hand of Fate 2 SDK (modding toolkit), it's not the easiest thing to do... There is documentation for creating these under the subheading 'Creating your own monster cards with custom values:'. This shows you how to create them. You can find it [here](#)!

## Templates and documentation for creating your own card art?

We've put together documentation showing you how anyone who has access to a photoshop license can use these templates + tools to make custom basic card art, brimstone and platinum card art for their own mods. You can access all of the necessary files [here!](#)