



**HUMANKIND™**

**OFFICIAL MODDING GUIDE**

## Table of Contents

1 Installing Modding Tool and Creating the Mod.....	6
1.1 Downloading Unity 2020.3.7f1 .....	6
1.2 Installing and launching “HUMANKIND Mod Tools” .....	7
1.3 Starting the first project .....	10
1.4 Continue working on the mod.....	12
2 Game Assets Types .....	13
2.1 UI mapper notes .....	15
3 Game Asset Modification/Replacement.....	18
4 Game Asset Creation .....	21
4.1 Duplicating the existing object.....	21
4.2 Creating objects from scratch.....	23
5 Building and Running the Mod.....	28
6 Shipping the Mod .....	31
6.1 Shipping the mod within the site.....	31
6.2 Shipping the mod within Unity editor .....	33
Modding Tool Possibilities .....	37
7 Adding a new Technology .....	37
7.1 Setting up the environment .....	37
7.2 Creating the new technology.....	39
7.3 Setting up the definition.....	39
7.4 Setting up the cost .....	40
7.5 Setting up the prerequisites .....	40
7.6 Setting up the unlock references.....	41
7.6.1 Setting up the unlock reference .....	41
7.6.2 Setting up the unlock reference UI mapper.....	41
7.7 Mapping the technology to the Tech Tree .....	42
7.8 Testing.....	44
8 Adding a new Constructible District .....	46
8.1 Setting up the environment .....	46
8.2 Creating the district definition .....	48
8.3 Setting up the “Definition” tab.....	48
8.3.1 Setting up the “Constructible” section .....	48

8.3.2	Setting up the “District” section .....	50
8.3.3	Setting up the “Extension District” section .....	51
8.4	Setting up the “Construction” tab.....	52
8.5	Setting up the “Prerequisites” tab.....	54
8.5.1	Setting up the “Constructible” section .....	54
8.5.2	Setting up the “District/Own Tile” section .....	56
8.5.3	Setting up the “District/Neighbouring Tiles” .....	58
8.5.4	Setting up the “District/Borough” .....	59
8.6	Setting up the UI and GeoLocalization .....	59
8.7	Adding the constructible district into the Tech Tree .....	60
8.8	Testing.....	61
9	Adding a new Unit .....	63
9.1	Setting up the environment .....	63
9.2	Creating the new unit .....	64
9.3	Setting up the “Definition” tab.....	65
9.3.1	Setting up the “Constructible” section .....	65
9.3.2	Setting up the “Unit” section .....	67
9.3.3	Creating a unit descriptor.....	69
9.4	Setting up the “Construction” tab.....	71
9.5	Setting up the “Prerequisites” Tab.....	72
9.6	Setting up the “AI” Tab .....	73
9.7	Adding the UI mapper for the unit .....	73
9.8	Creating the unit presentation .....	74
9.8.1	Setting up the environment.....	74
9.8.2	Creating the new unit presentation .....	75
9.9	Creating the new unit pawn presentation .....	76
9.10	Testing.....	78
10	Adding a new Culture .....	79
10.1	Setting up the environment .....	79
10.2	Creating the new culture .....	80
10.3	Setting up the “Definition” tab.....	81
10.4	Setting up the “Presentation” tab.....	82
10.5	Adding a UI mapper for civilization .....	84

10.6	Creating the new legacy trait.....	85
10.7	Creating the new legacy trait descriptor .....	86
10.8	Setting up the legacy trait descriptor .....	87
10.9	Setting up the legacy trait.....	88
10.10	Adding a UI mapper for the Legacy Trait.....	89
10.11	Testing.....	90
10.12	Adding an emblematic district .....	91
10.13	Adding an emblematic unit.....	92
10.14	Testing.....	95
11	Adding a new Narrative Event .....	96
11.1	Setting up the environment .....	96
11.2	Create a new narrative event .....	97
11.3	Setting up the “Definition” section .....	98
11.4	Event category.....	99
11.5	Setting up the “Trigger” section .....	100
11.6	Setting up the “UI” section.....	103
11.7	Setting up the “GeoLocalization” section.....	103
11.8	Setting up the “Choices” section .....	103
11.9	Testing.....	105
11.10	Adding a lasting effect.....	106
11.10.1	Adding a descriptor.....	106
11.10.2	Adding a status definition.....	109
11.10.3	Applying a lasting effect to choices.....	112
11.11	Testing.....	114
11.12	Adding narrative consequences .....	116
11.12.1	Narrative consequence.....	116
11.12.2	Testing .....	119
11.12.3	Narrative consequence tag.....	120
11.12.4	Testing .....	123
12	Adding a new Civic.....	125
12.1	Setting up the environment .....	125
12.2	Creating the new civic.....	127
12.3	Setting up the civic definition.....	127

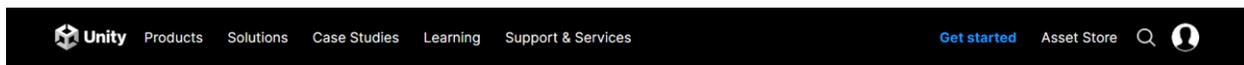
12.4	Setting up the empty choices .....	127
12.5	Setting up the civic descriptors.....	128
12.6	Setting up the civic descriptor UI mapper .....	128
12.7	Fulfilling the empty choices .....	129
12.8	Setting up the “Prerequisites” section.....	130
12.9	Mapping the new civic to the Civics screen .....	131
12.10	Setting the new civic choices UI mappers .....	132
12.11	Creating a new civic narrative event.....	134
12.12	Setting up the “Definition” section.....	134
12.13	Setting up the “Trigger” section .....	135
12.14	Setting up the “Variables” section.....	135
12.15	Setting up the “UI” and “GeoLocalization” sections .....	136
12.16	Setting up the choice.....	136
12.17	Testing .....	137
13	Balancing Battle .....	139
13.1	Changing unit’s basic stats.....	140
13.2	Testing .....	141
13.3	The unit abilities .....	142
13.4	The battle abilities .....	145
13.5	The game effect .....	149
13.6	Property effects calculations .....	150
13.7	Balancing the battle ability .....	154
13.7.1	Rebalancing the charge ability .....	154
13.7.2	Rebalancing the unit specialty .....	156
14	Useful data .....	159

# 1 Installing Modding Tool and Creating the Mod

To start the modding process of the game, the “HUMANKIND Mod Tools” has to be installed as well as Unity 2020.3.7f1.

## 1.1 Downloading Unity 2020.3.7f1

1. Go to <https://unity3d.com/get-unity/download/archive> to get the necessary Unity version.



## Unity download archive

From this page you can download the previous versions of Unity for both Unity Personal and Pro (if you have a Pro license, enter in your key when prompted after installation). Please note that we don't support downgrading a project to an older editor version. However, you can import projects into a new editor version. We advise you to back up your project before converting and check the console log for any errors or warnings after importing.

### Long Term Support releases

The LTS stream is for users who wish to continue to develop and ship their games/content and stay on a stable version for an extended period.

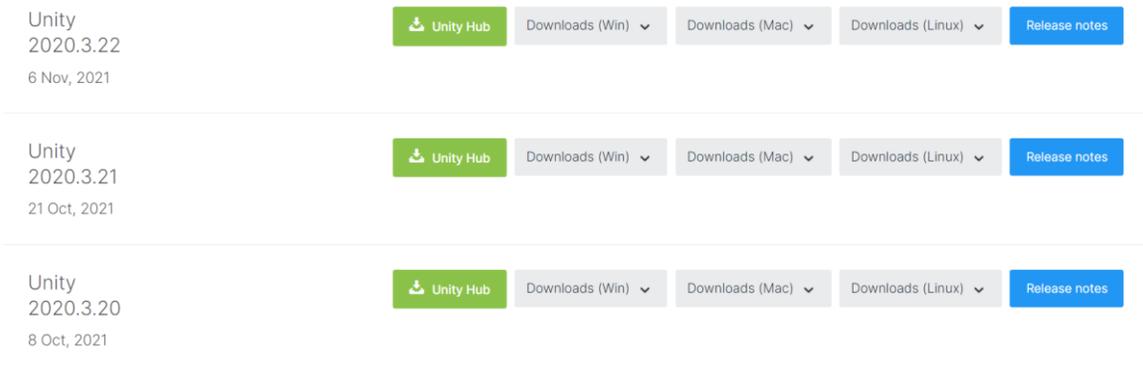
[Download LTS releases](#)

Unity 2021.x   **Unity 2020.x**   Unity 2019.x   Unity 2018.x   Unity 2017.x   Unity 5.x   Unity 4.x   Unity 3.x

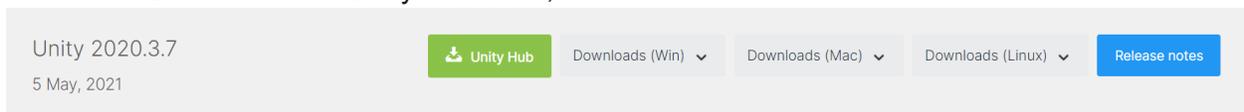


## 2. Select “Unity 2020.x”

Unity 2021.x   **Unity 2020.x**   Unity 2019.x   Unity 2018.x   Unity 2017.x   Unity 5.x   Unity 4.x   Unity 3.x

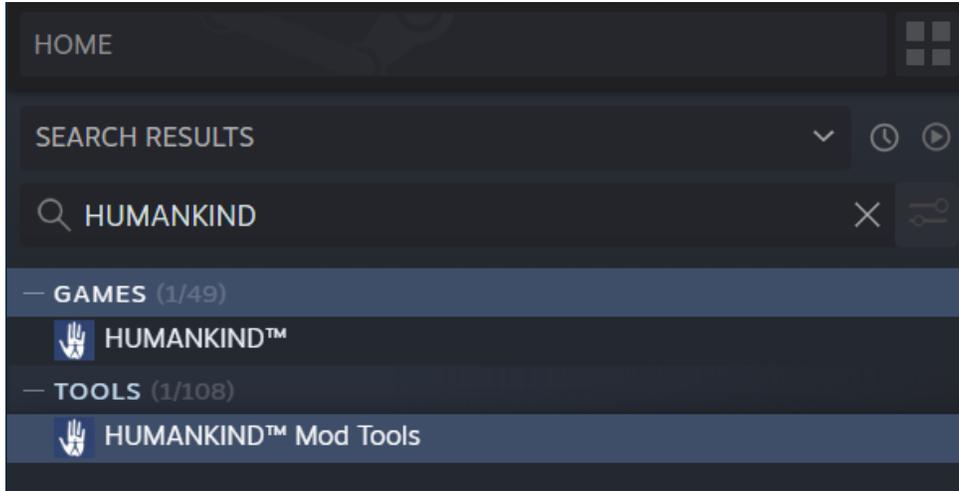


## 3. Search for “Unity 2020.3.7”, download and install it.

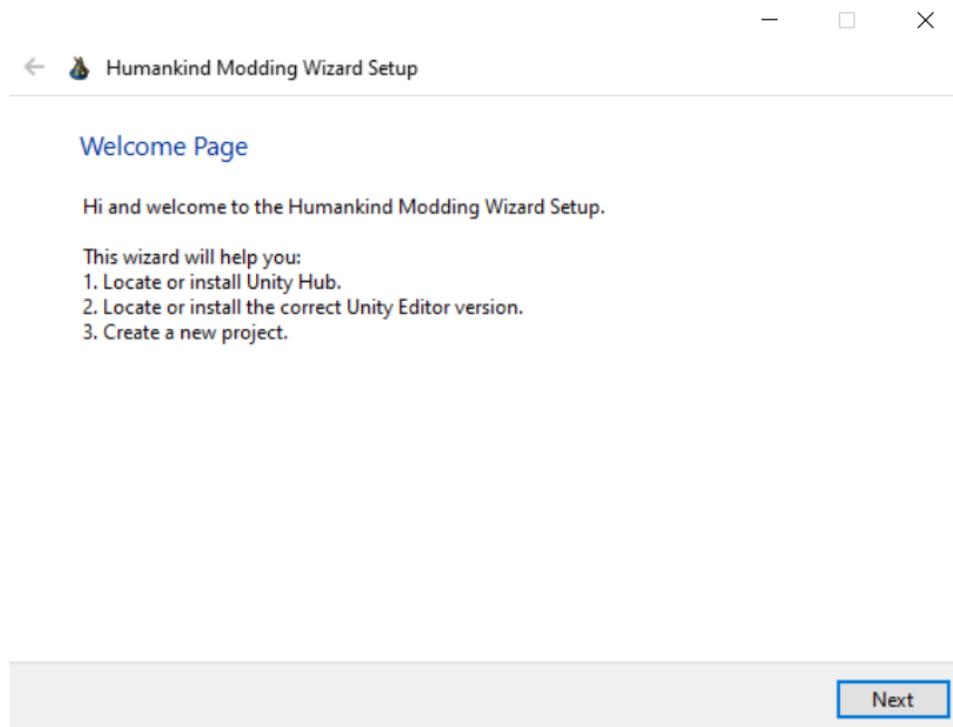


## 1.2 Installing and launching “HUMANKIND Mod Tools”

1. Search for HUMANKIND in your library in Steam and install “HUMANKIND Mod Tools” under the “TOOLS” section.



2. Launch the modding tool and go through the launcher steps.



## Unity Hub

Unity Hub is correctly installed and located.

C:\Program Files\Unity Hub\Unity Hub.exe

Locate

Install Unity Hub

Next

## Unity Account and License

### **Please read carefully!**

In order to create a modding project, you need a Unity account and an active License.

If you do not have a Unity account, you can create one by clicking on this link:

[Create Unity account](#)

Once you have a Unity account, login inside the Unity Hub and activate your license.

Open Unity Hub

[Activating a licence in the Hub](#)

Next

←  Humankind Modding Wizard Setup

---

### Unity Editor 2020.3.7f1

Unity Editor is correctly installed and located.

Choose where Unity Editor will be installed.

←  Humankind Modding Wizard Setup

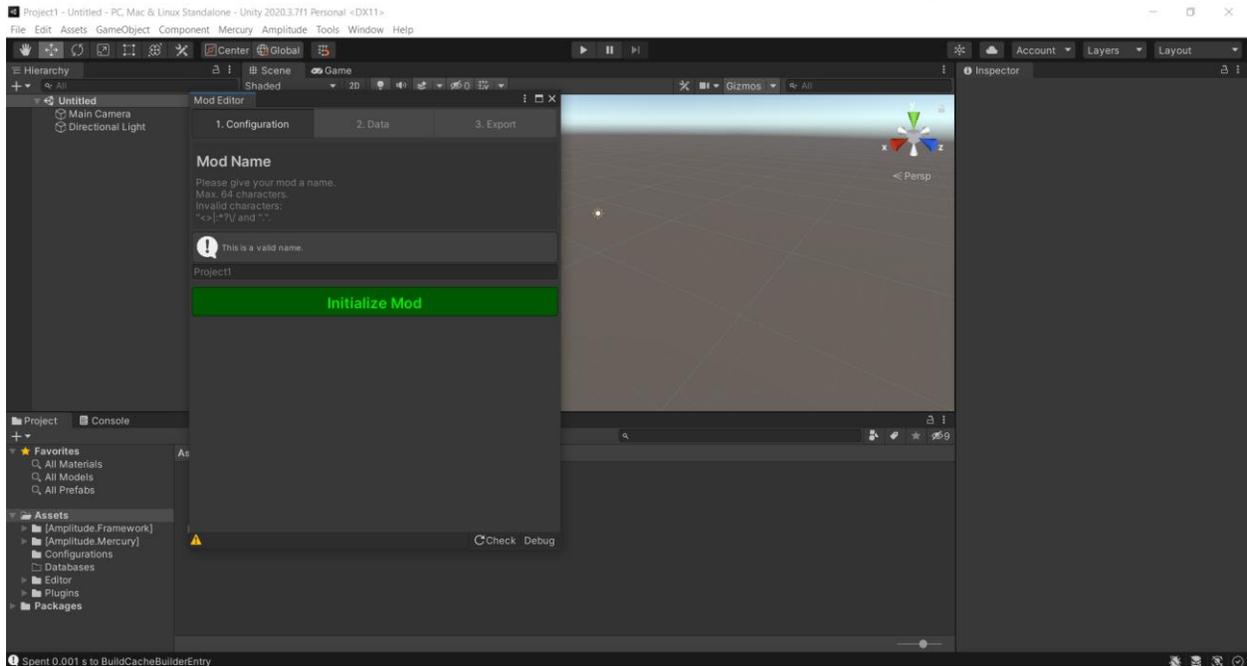
---

### Create Project

Choose the root folder where your project(s) will be created.

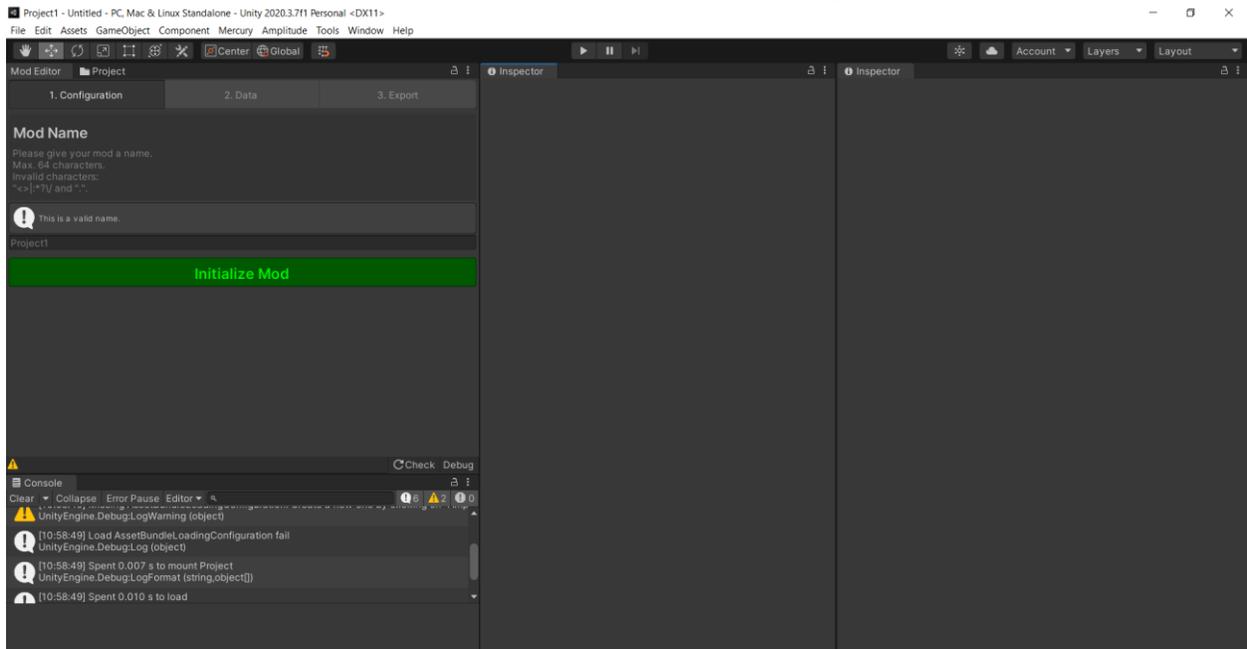
Choose your new project name.

### 3. Once Unity is fully started, you can proceed with modding.

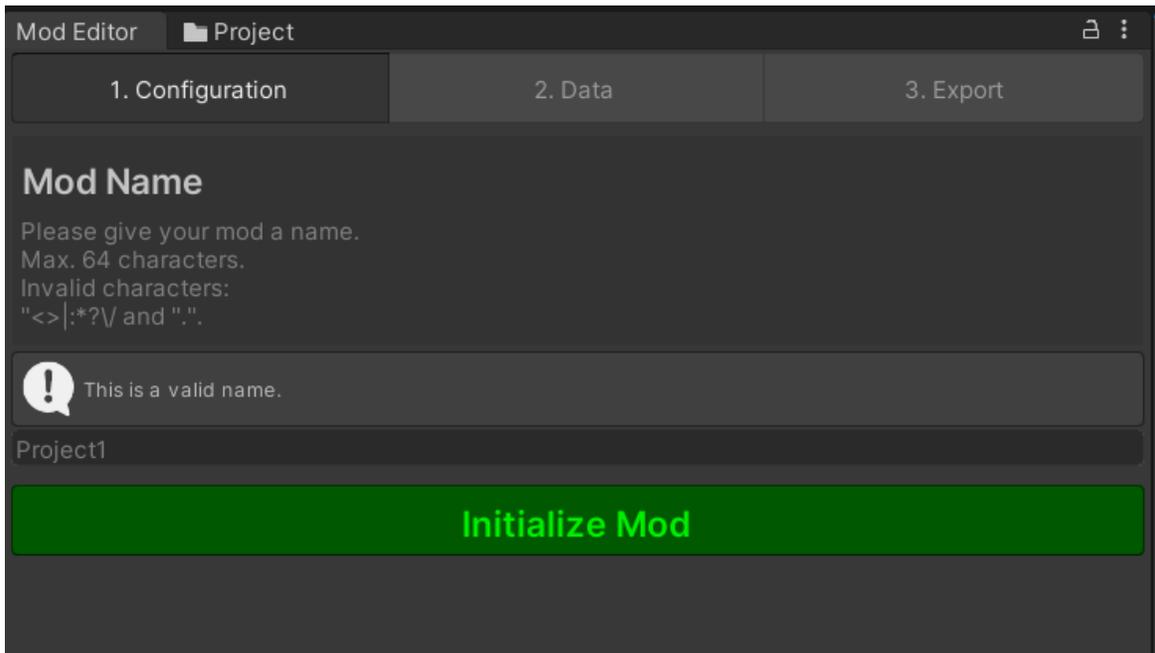


## 1.3 Starting the first project

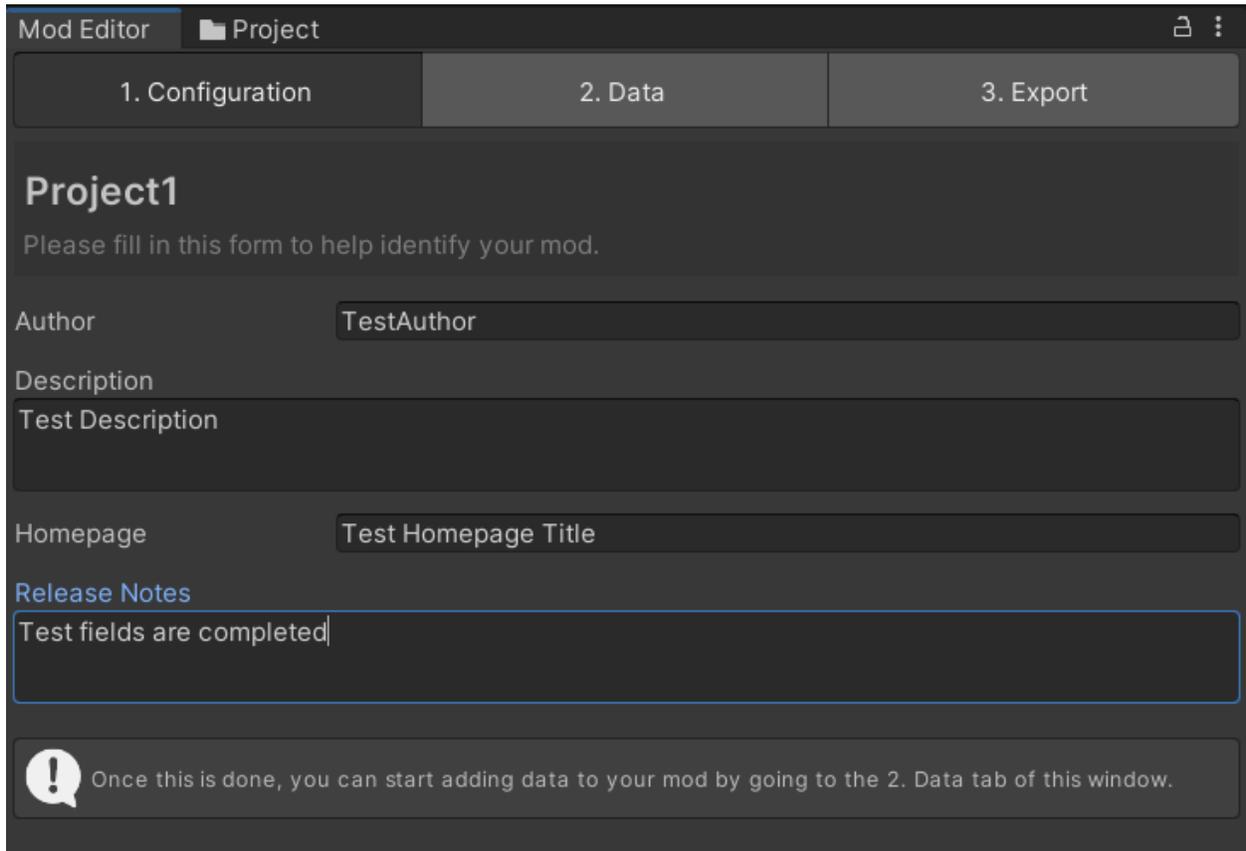
1. Most of the usual Unity tabs won't be used during the mod creation so you should rearrange the working space to the more efficient one. We recommend you to focus on "Mod Editor", "Project", "Console" and two "Inspector" tabs.



2. Once the working space is ready, create the new mod by clicking on the “Initialize Mod” button.



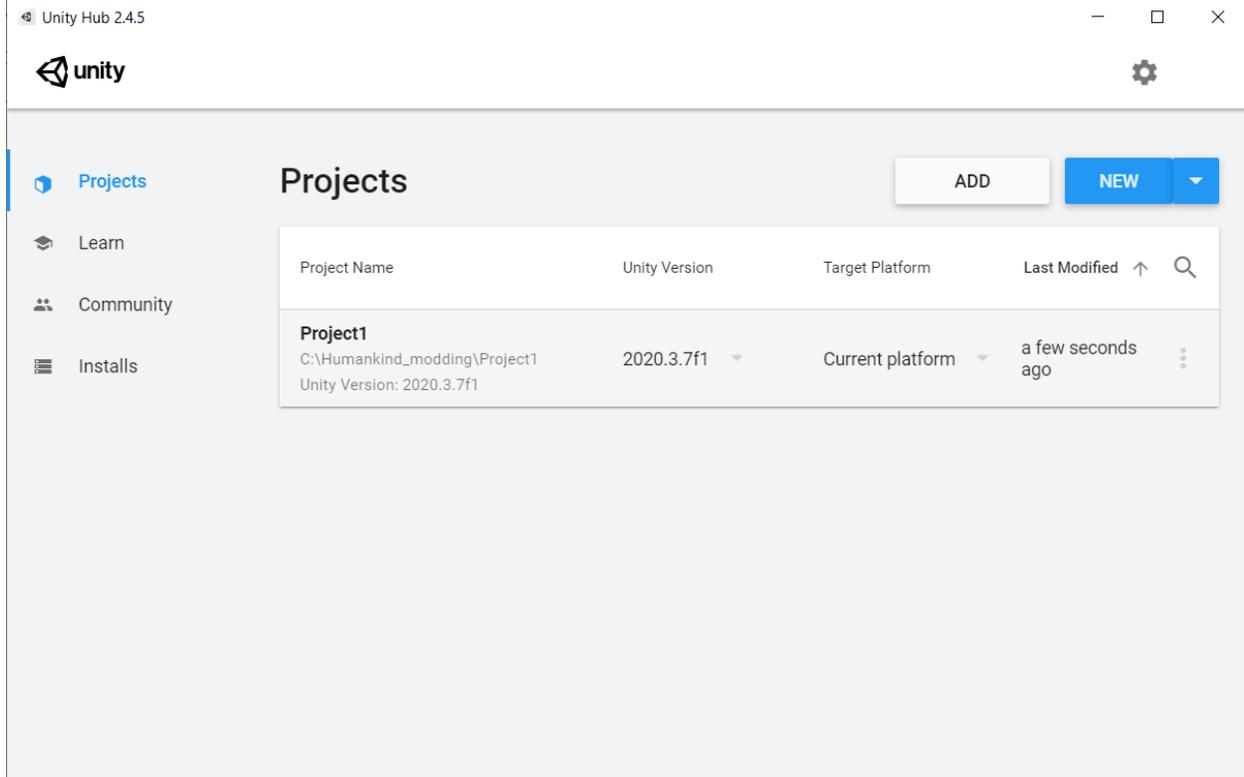
3. Complete the related fields like “Author”, “Description”, etc.



Now you are ready to export some databases or work with your own to create the first mod!

## 1.4 Continue working on the mod

If the project was closed you can reopen it from the Unity Hub to continue working on the mod.



## 2 Game Assets Types

In order to modify the game, you have to work with the related in-game databases, collections, and assets.

The game has 3 main associated types of assets:

- Definition - defines the object characteristics, settings and applies a descriptor to this object.

The screenshot shows a dark-themed software interface for editing a game asset. At the top, the title bar reads "Test 1 (Extension District Definition)". Below the title bar, there are several configuration fields:

- Script:** ExtensionDistrictDefinition
- Key:** 1
- Hidden:**
- Is Obsolete:**

Below these fields is a tabbed interface with four tabs: "Definition", "Construction", "Prerequisites", and "AI". The "Definition" tab is active. Under this tab, there is a section for "DLC Prerequisite" (expanded) and a "Constructible" section with the following settings:

- Category:** Food
- Level:** 0
- Unicity:** Any
- Constructible Visual Affinity:** DistrictVisualAffinity\_Base\_Food

Below the "Constructible" section is a "District" section with the following setting:

- Prototype:** Extension\_Base\_Food

Next is a section for "Own Descriptor References" which contains a list of 7 items, each with a search icon and a close icon:

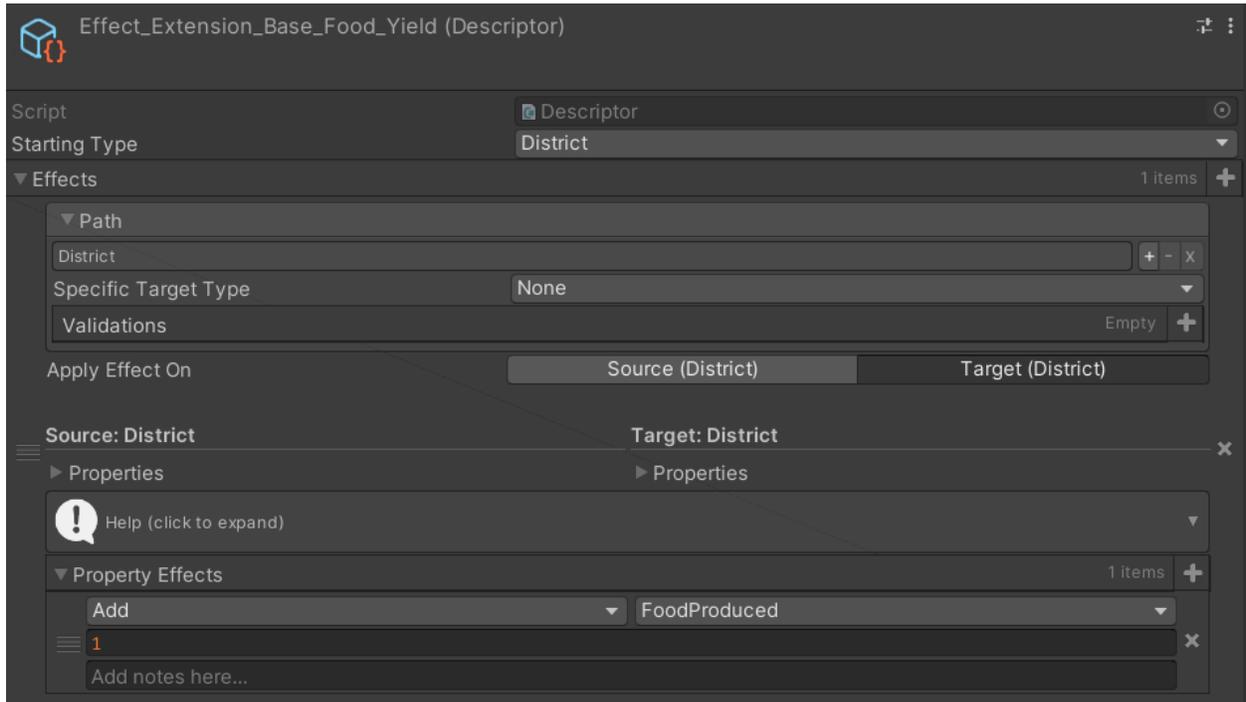
- Effect\_Extension\_Base\_Food
- Effect\_Extension\_Base\_Food\_Yield
- Effect\_Extension\_Base\_Science
- Effect\_Extension\_Base\_Science\_Yield
- Effect\_Extension\_Default\_Vision
- Effect\_Extension\_WorkplaceSlot\_Food
- Effect\_Extension\_WorkplaceSlot\_Science

At the bottom, there are two checked checkboxes:

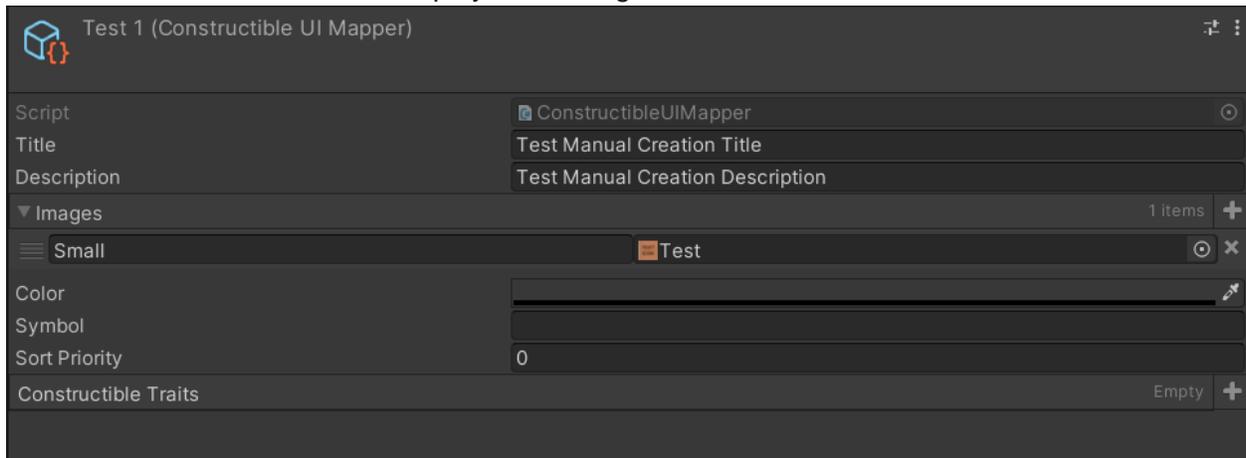
- Is Replaceable Or Destroyable:**
- Can Be Downgraded To Ruin:**

The final section is "Additional Visual Levels" with the value "Empty".

- Descriptor - defines the effect that is applied to the in-game objects, assigns tags, etc. Several descriptors can be applied to the same definition object.



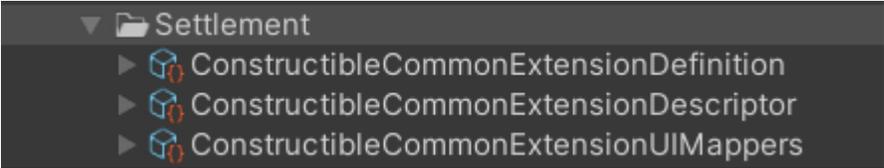
- UI Mapper - defines the interface representation of the asset including name, description, images, etc. If there is no UI mapper object for the definition, it won't be visible to other players in the game.



All these 3 objects have different types for different in-game objects/mechanics and are divided into different collections. Example of different definitions:

```
AirUnitDefinition
AirportDefinition
ArtificialDepositDistrictDefinition
ArtificialWonderDefinition
ColonialDistrictDefinition
ConstructibleActionDefinition
EmpireWideConstructionParticipationDefinition
ExploitationDistrictDefinition
ExtensionDistrictDefinition
FakeEmblematicCamp
HolySiteDefinition
LandUnitDefinition
MissileUnitDefinition
NationalProjectDefinition
NationalProject_Leveling
```

Example of different collections:



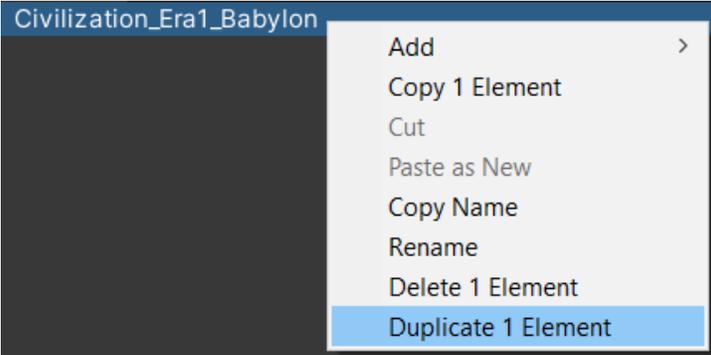
For example, city district “Harbour” consists of:

- 1 definition object “Extension\_Base\_Harbour”;
- 3 assigned descriptors:
  - Effect\_Extension\_Base\_Harbour;
  - Effect\_Extension\_Default\_Vision;
  - Tag\_Extension\_Urban.
- 1 UI mapper “Extension\_Base\_Harbour”.

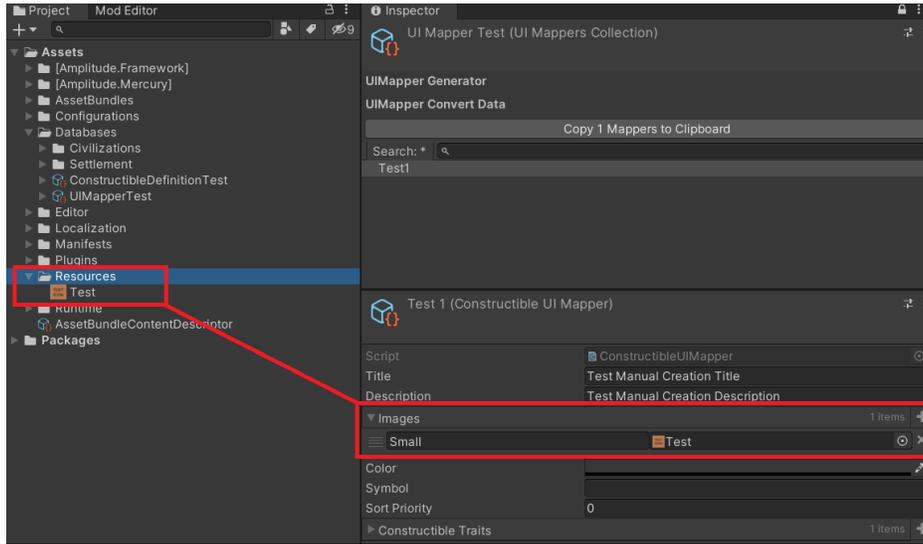
## 2.1 UI mapper notes

Few important notes regarding UI Mappers:

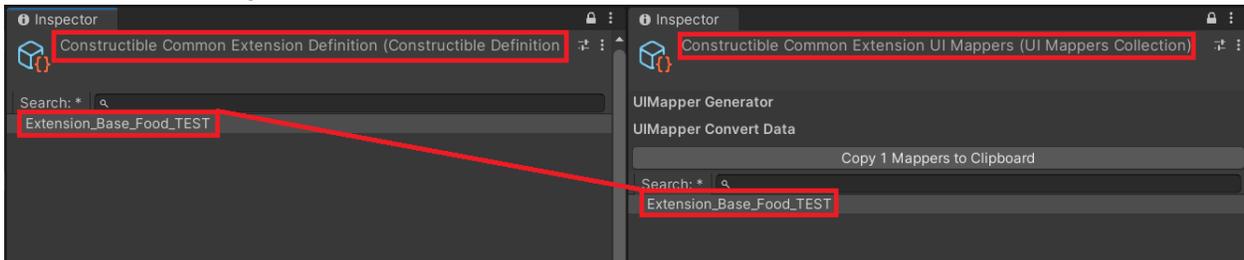
- The best way to work with UI mappers is to duplicate the existing similar UI mapper. It will eliminate the issue of having the wrong UI mapper type.



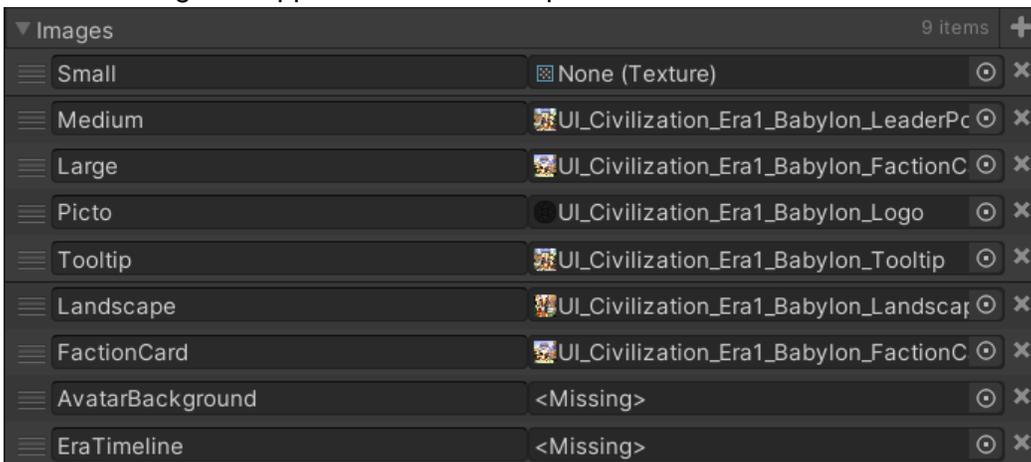
- If the new image is added to the UI mapper object, it must be stored inside the “Resources” folder of the project.



- All UI Mappers must have the same names as Definition objects to be displayed in the game.



- The “Images” field of the UI mapper works only with specific keywords. Few keywords for the example: “Small”, “Medium”, “Large”, “Picto”, “Tooltip”, “Landscape”, “FactionCard”, “AvatarBackground”, “EraTimeline”. Please refer to existing UI mappers for more examples.



- The field “Color” works as a filter to the attached images.
- The “Sort Priority” is used to sort units in the armies.
- The “Constructible Traits” is used during some special occasions.

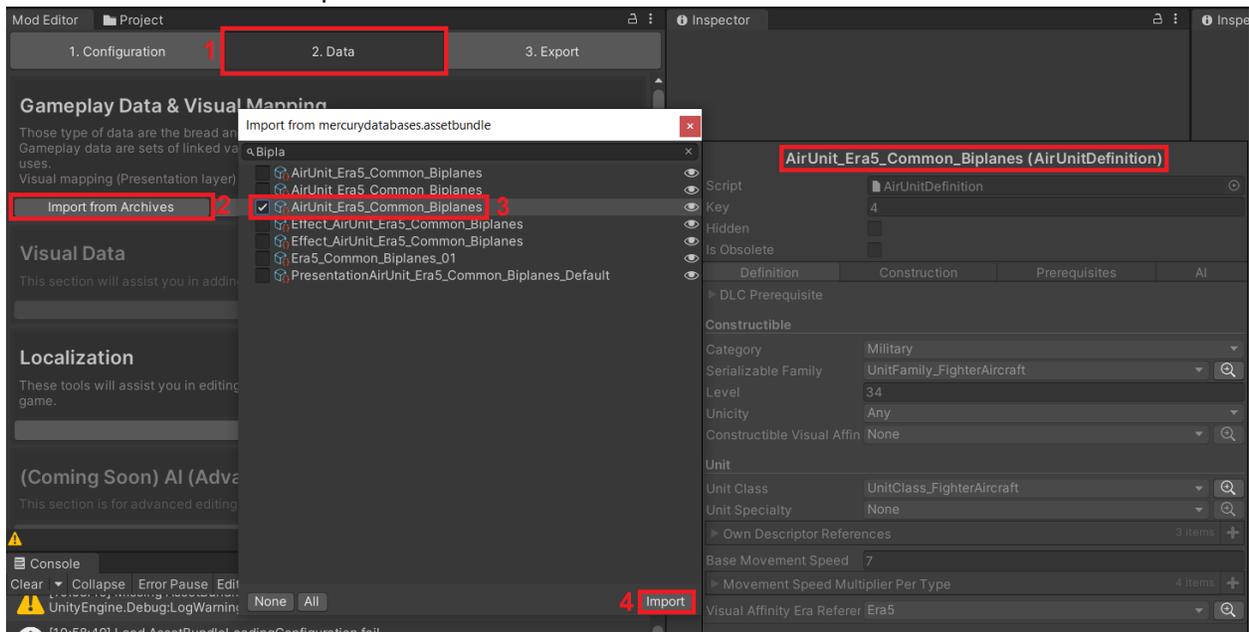
The screenshot shows a detailed view of a unit's properties in a game editor. The unit is named "Land Unit\_Era 2\_Common\_Citizens" and is associated with the "Constructible UI Mapper" script. The interface includes fields for "Title" and "Description" with placeholder text. A section for "Images" shows 19 items. The "Color" field is currently empty. The "Sort Priority" is set to 7800. A "Constructible Traits" section contains two traits: "Raised When Besieged" and "Raised From Militarist Affinity".

Script	ConstructibleUIMapper
Title	%LandUnit_Era2_Common_CitizensTitle
Description	%LandUnit_Era2_Common_CitizensDescription
Images	19 items
Color	
Symbol	
Sort Priority	7800
Constructible Traits	2 items
	Raised When Besieged
	Raised From Militarist Affinity

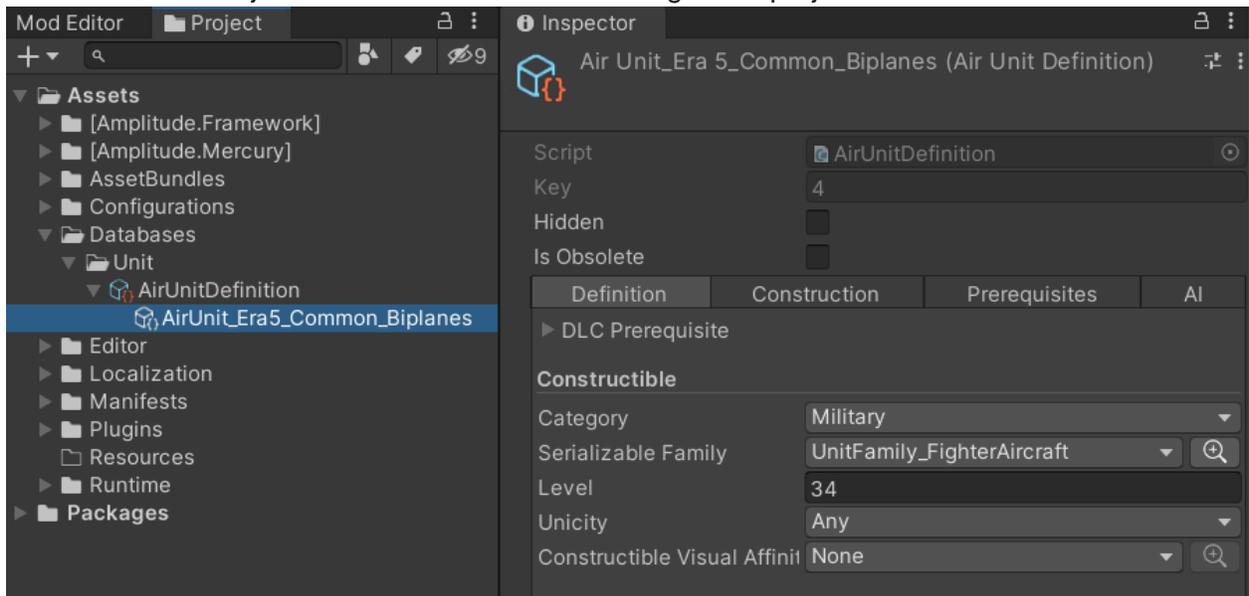
### 3 Game Asset Modification/Replacement

The easiest way to work with game data is to import existing database object and modify it:

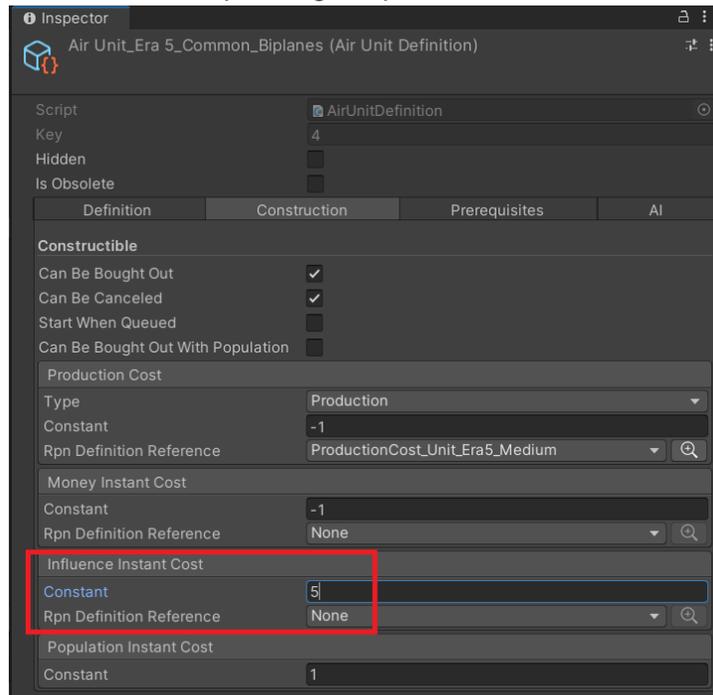
1. Go to “2. Data” of the Mod Editor tab.
2. Select “Import from Archives”.
3. Search for the necessary object (let’s modify Biplane’s movement speed and production cost). To select the correct asset type, please refer to the text in parentheses.
4. Press “Import”.



After the object becomes available for editing in the project structure.



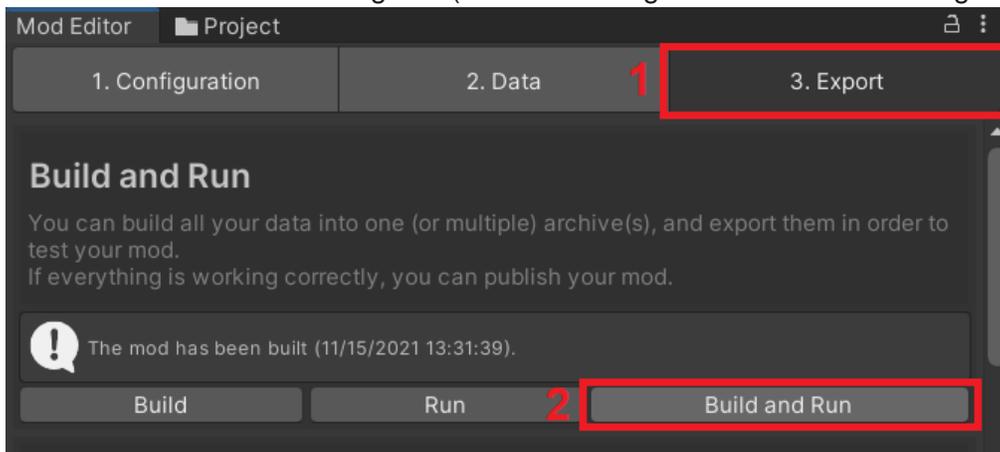
Now let's change the basic production cost of the unit. All the related fields will be described in more detail in the corresponding chapters later.

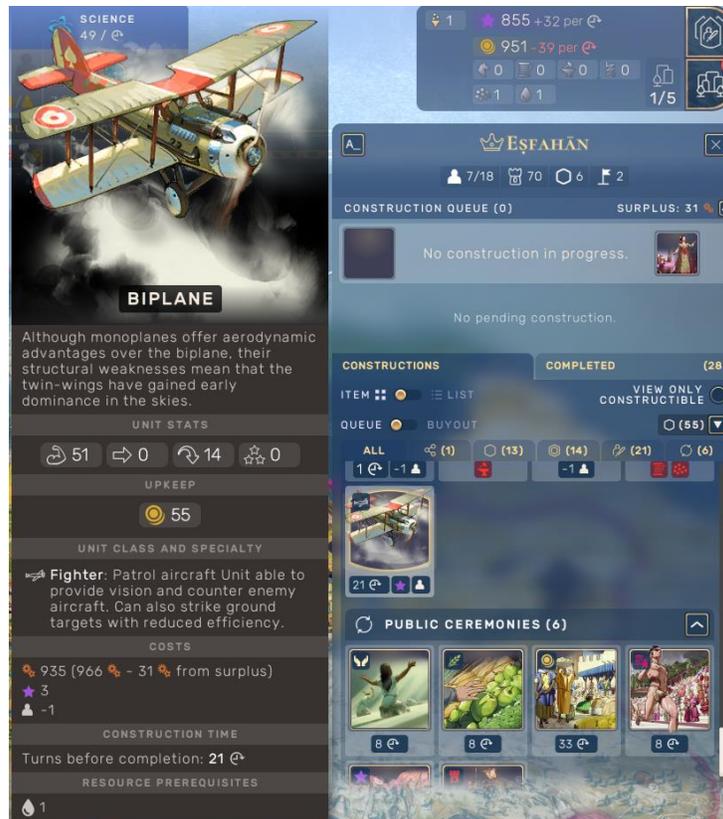


**Note!** This action will overwrite the initial object data.

Once the change is made, the mod can be tested.

1. Go to "3. Export" of the Mod Editor tab.
2. Select "Build and Run".
3. Check the result in the game (cost was changed a little bit due to the game speed).





**Important!** It is not recommended to store unused objects within the mod project structure. Please either not export unused objects, or remove unnecessary objects from collections/collections after the mod is created.

## 4 Game Asset Creation

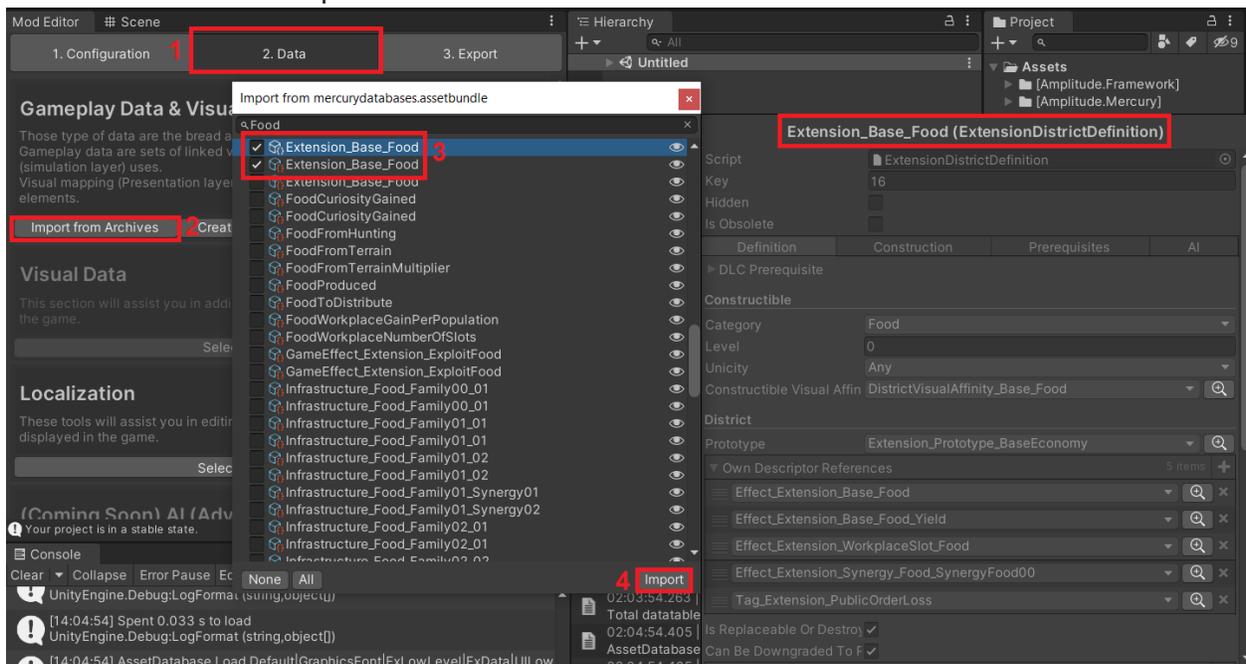
If you wish to create an in-game object from scratch there are 2 ways to do it.

- Duplicate the existing object and modify all the necessary fields.
- Create the object from scratch.

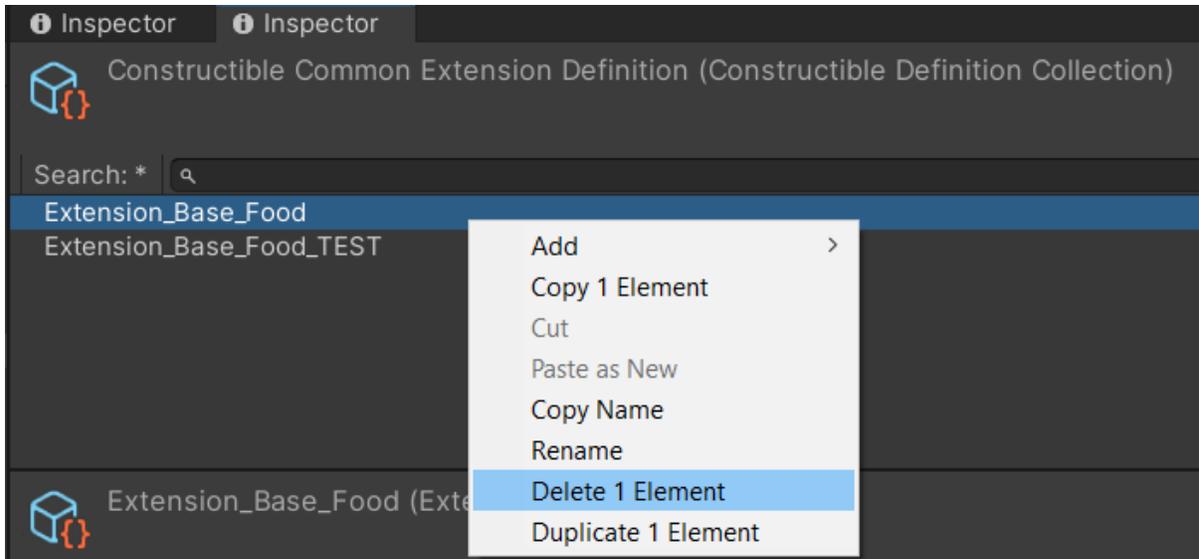
### 4.1 Duplicating the existing object

It is the easiest way of creating a new game asset. Simply duplicate the similar one and fill the fields accordingly.

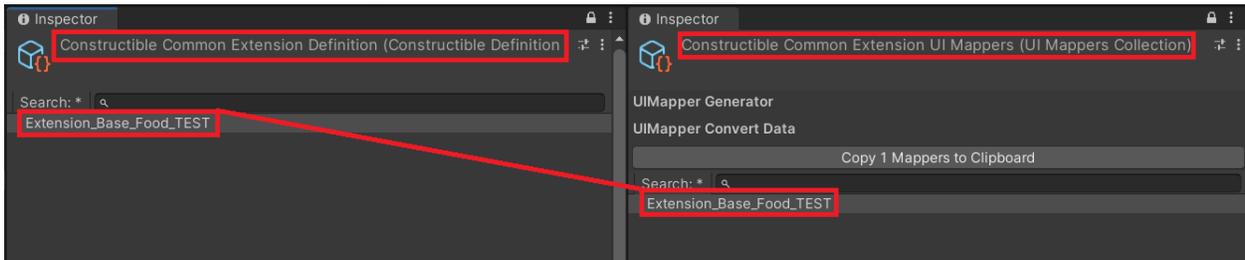
1. Go to “2. Data” of the Mod Editor tab.
2. Select “Import from Archives”.
3. Search for the necessary object definition and UI mapper objects. To select the correct object, please refer to the text in parentheses.
4. Press “Import”.



After the objects appear in the project structure window, go to both collections and duplicate the items. The original one should be deleted.



Rename both new objects so their names match from different collections.



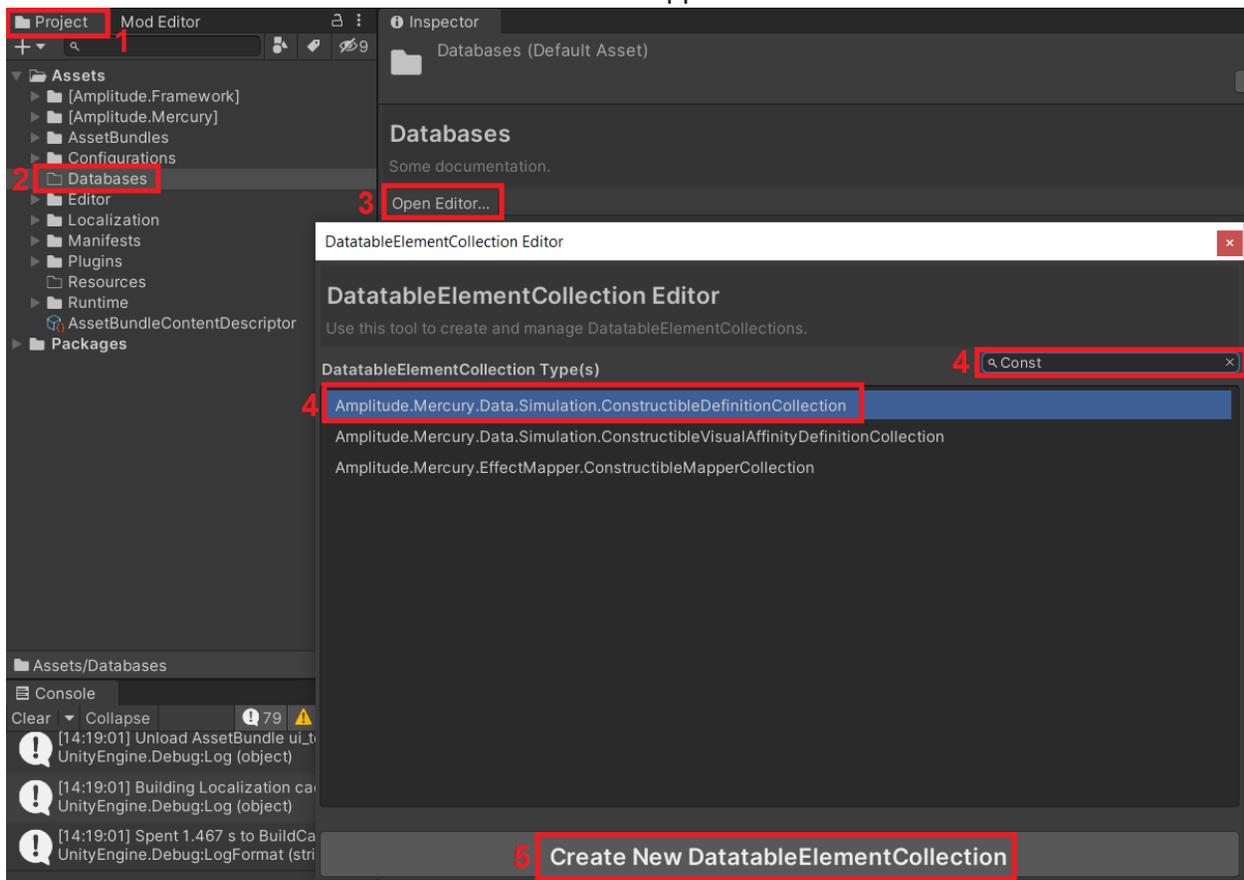
Modify both objects as required and test changes. As it is shown, a new district which produces food and science was created.

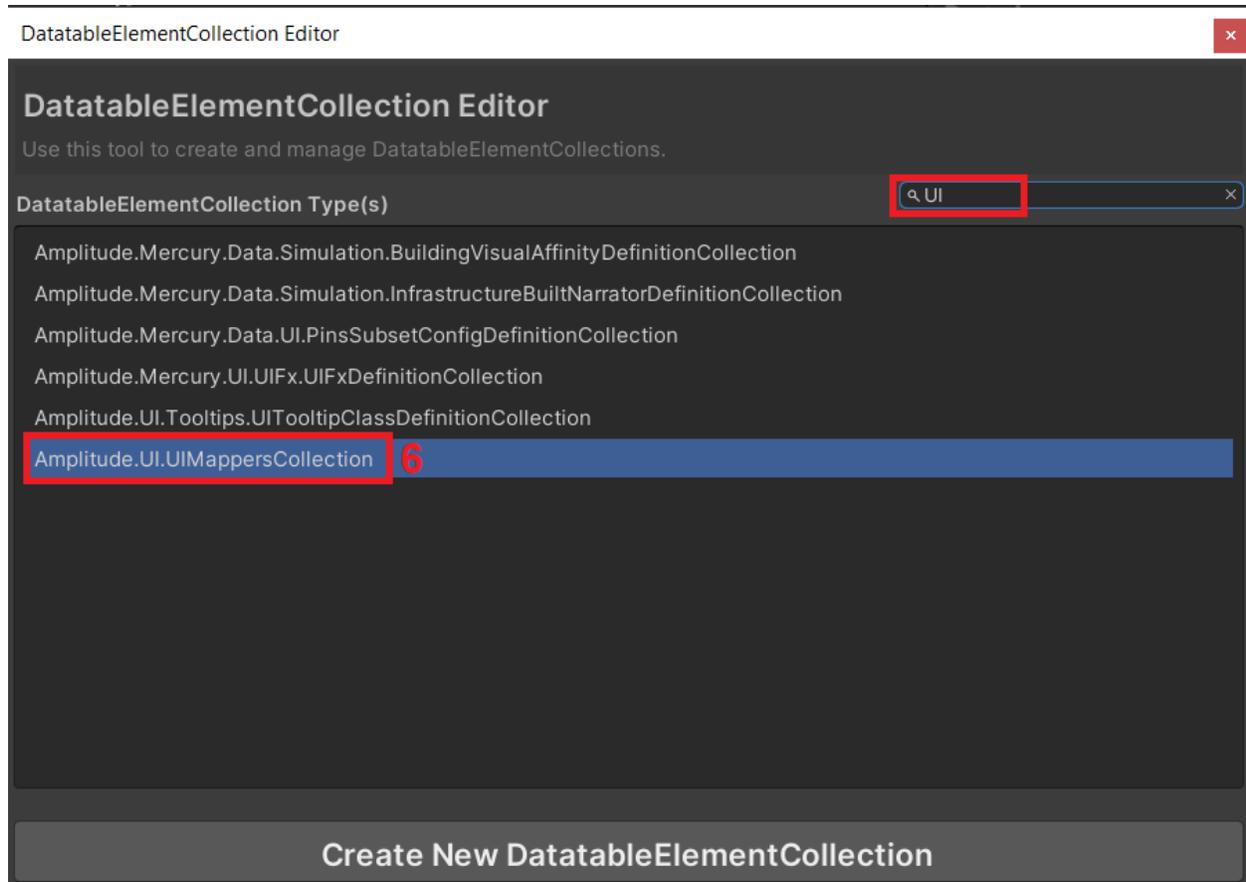


## 4.2 Creating objects from scratch

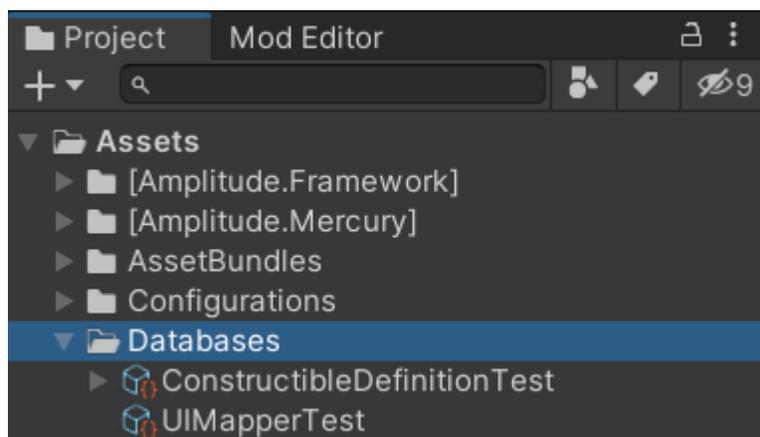
The more complicated way is to create collections and objects from scratch. You have to know the type of the object, its structure, and UI mappers before you are going to create one.

1. Go to the “Project” tab in Unity.
2. Select the “Databases” folder.
3. Press “Open Editor...”.
4. Search for the object you are going to create.
5. Press “Create New DatatableElementCollection”.
6. Reuse the window to create a UI mapper collection.



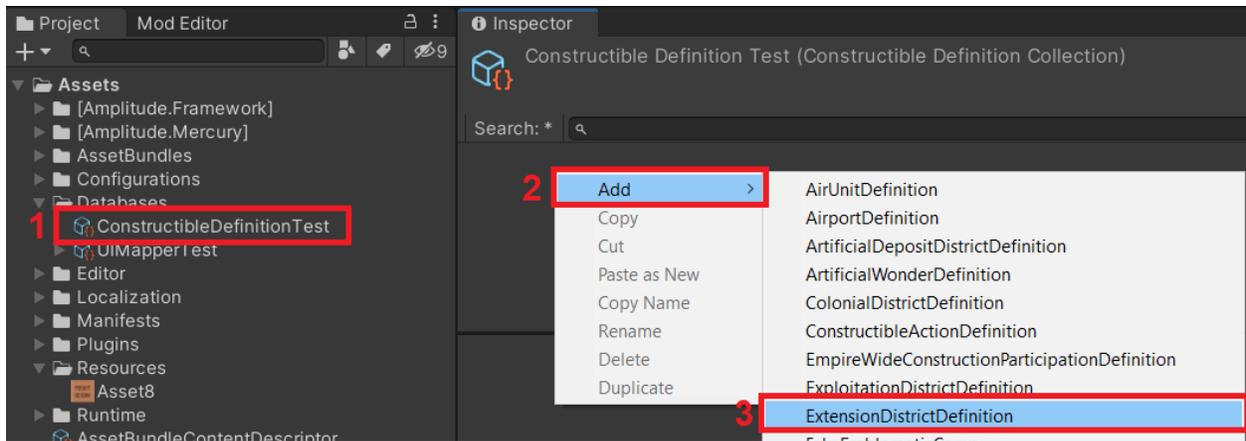


After the collections appear in the project structure, you can rename them for more efficient navigation.

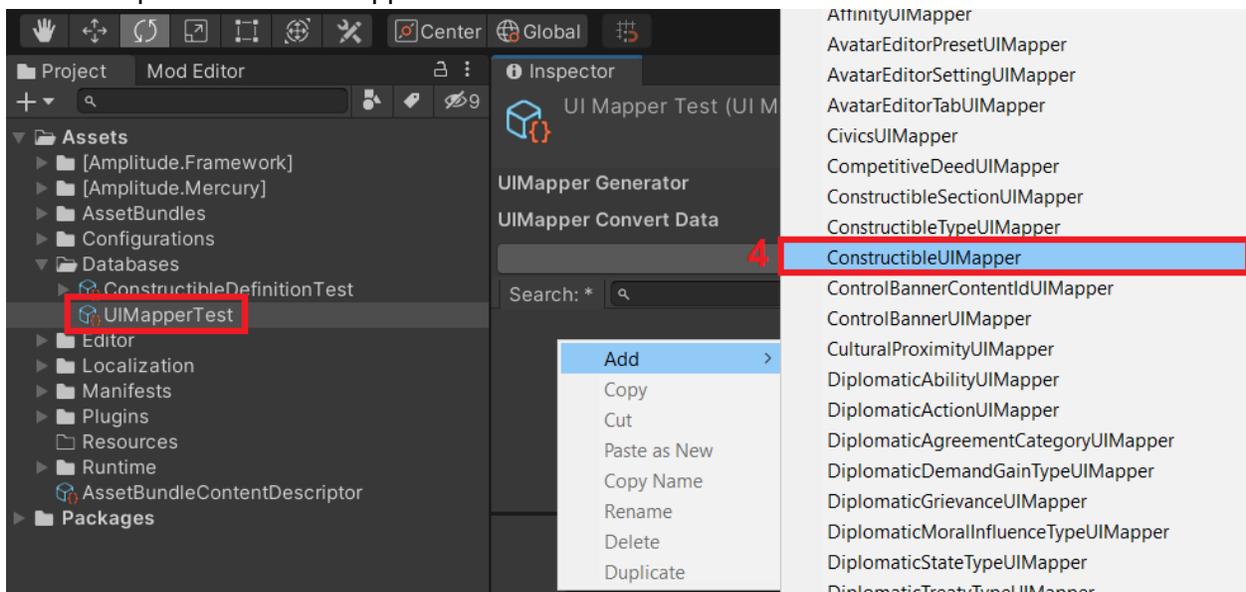


Collections are just containers for assets, create the definition and UI mapper objects separately:

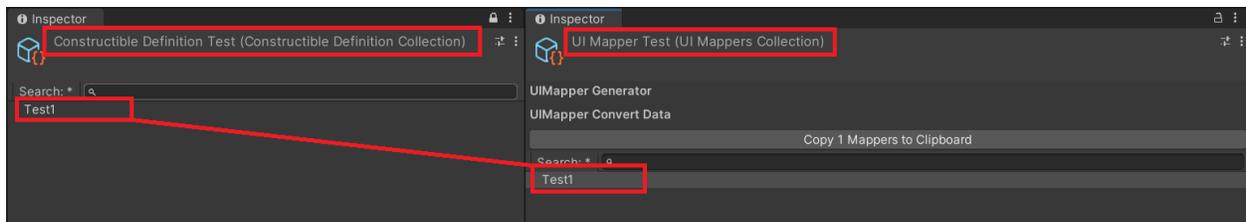
1. Select the collection.
2. Right-click in the inspector window of the selected collection.
3. Select the necessary object for creation from the "Add" menu.



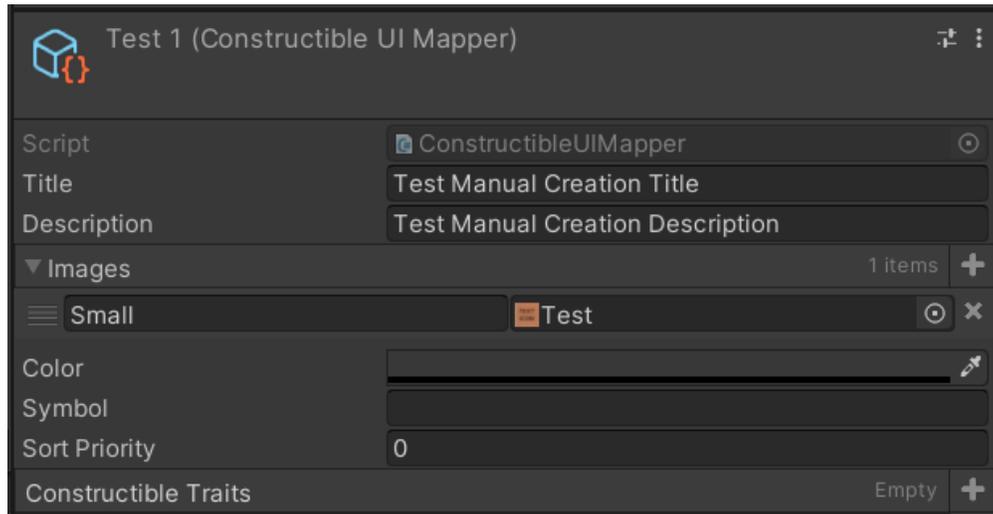
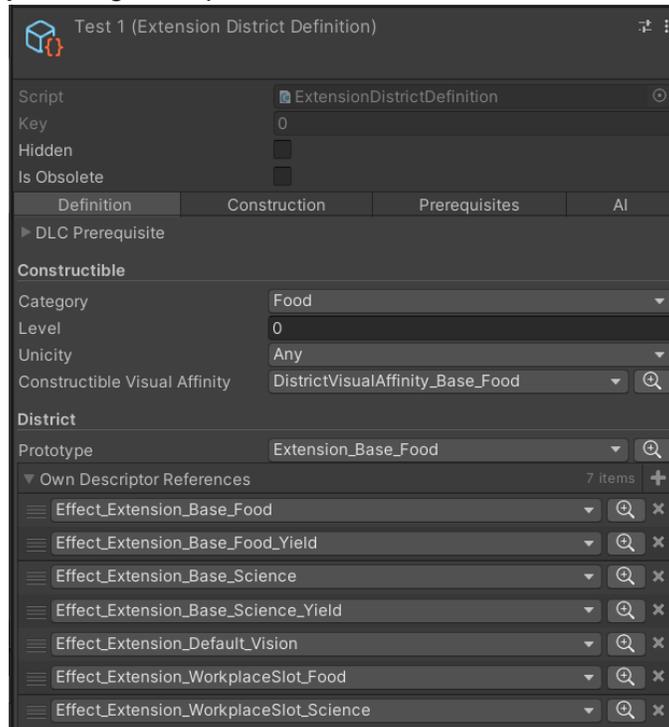
Repeat for the UI mapper.



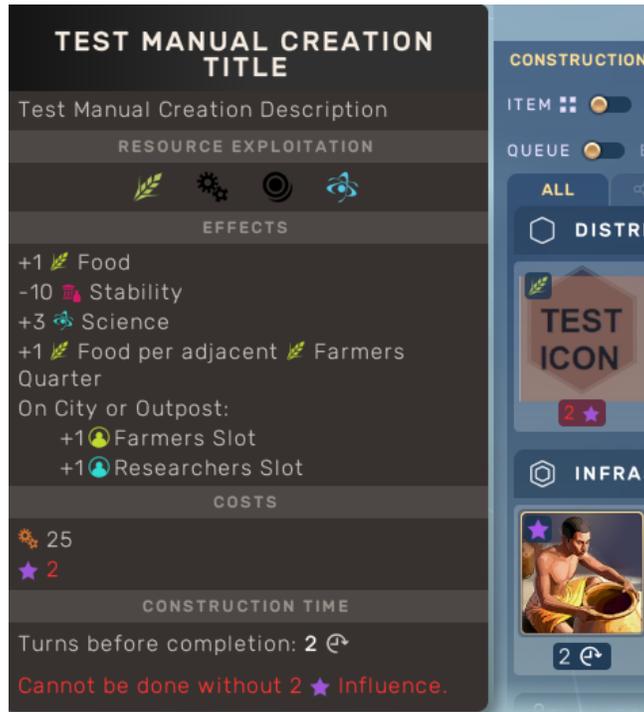
After the new objects are created in the collections, rename them both so their names match from different collections.



Modify both objects to get the preferred result and test them.



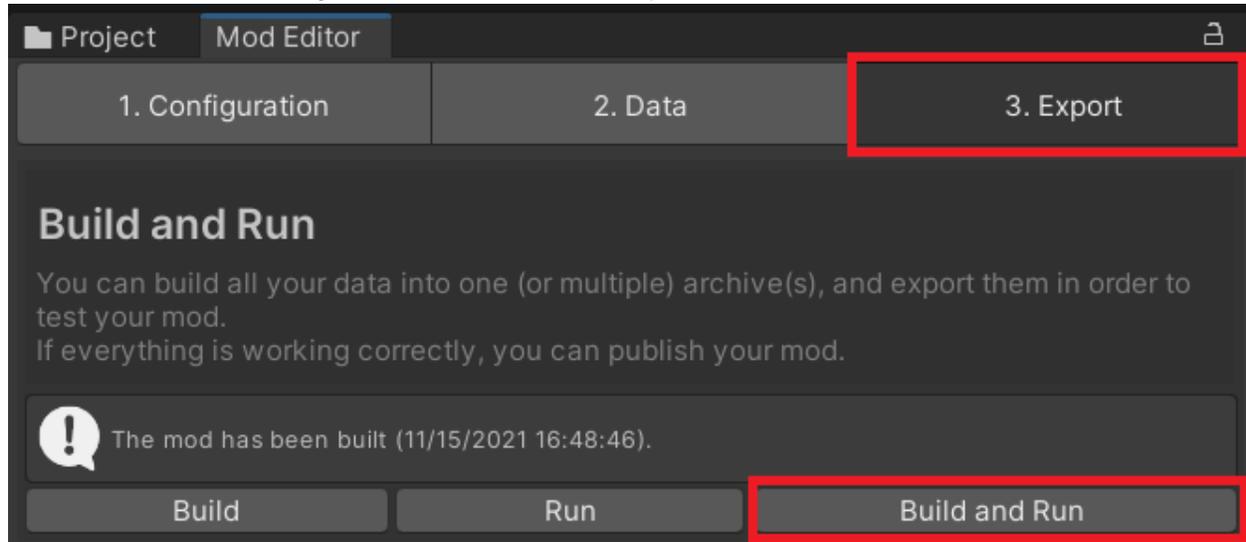
As it is shown, a new district which produces food and science was created.



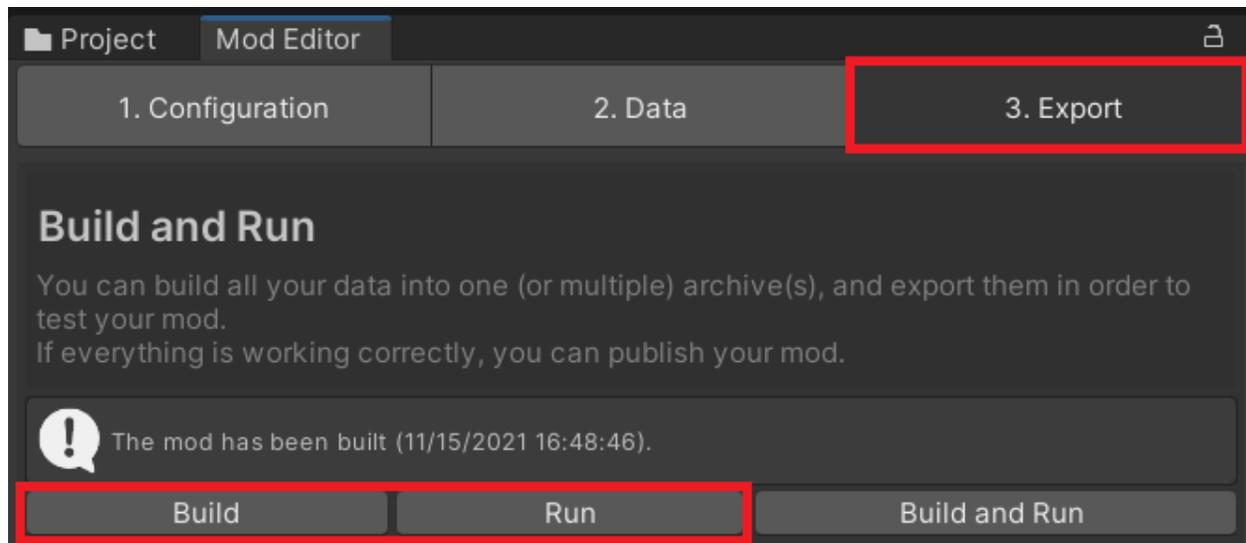
## 5 Building and Running the Mod

To test changes made in the modding tool you can:

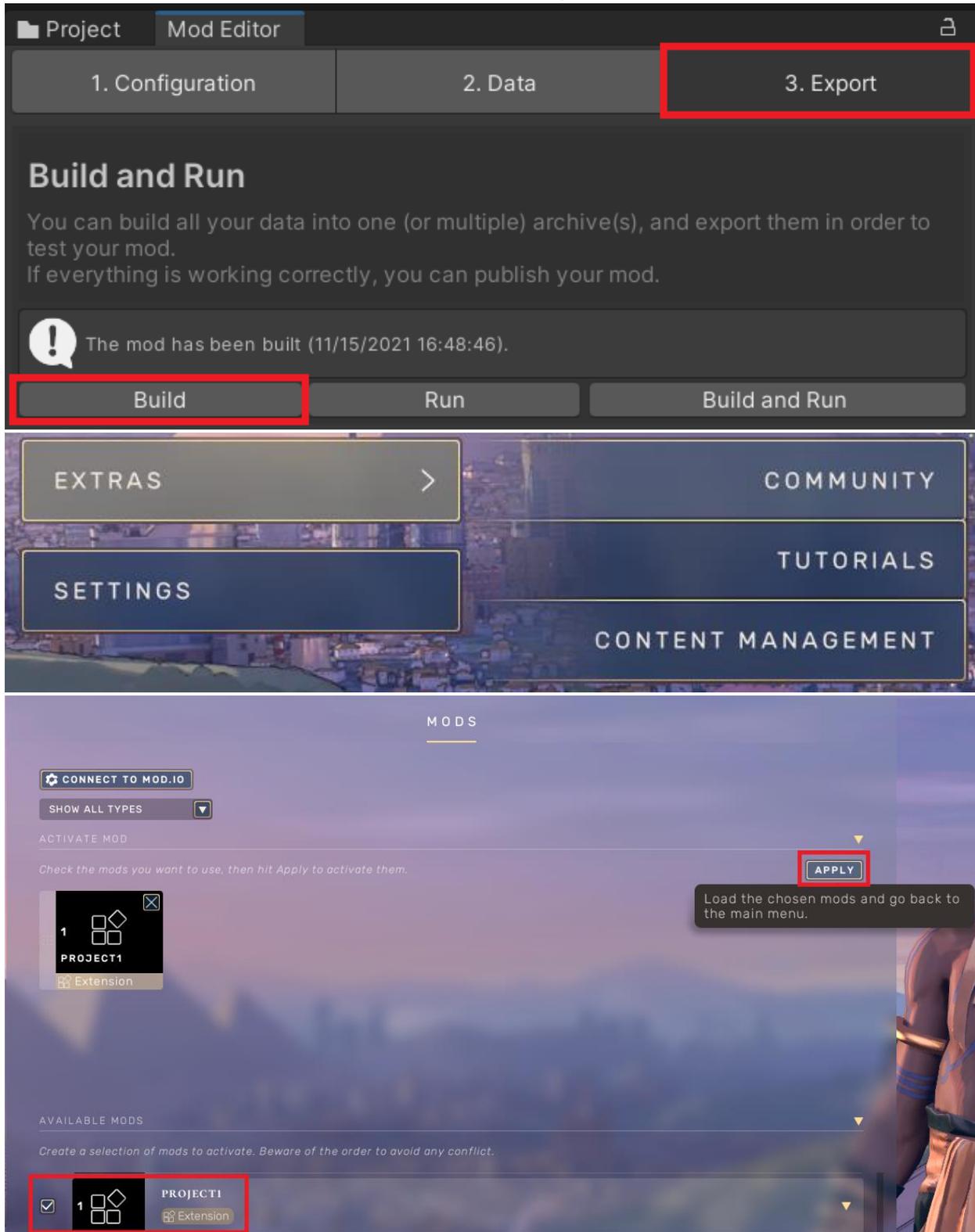
- Use the “Build and Run” button located in the “3. Export” section of the “Mod Editor” tab. The game will run automatically.



- Use “Build” and “Run” buttons separately located in the “3. Export” section of the “Mod Editor” tab.



- Build mod within the editor using the “Build” button, launch the game and select the correct mod in the game menu “Extras” -> “Community”. You can see the number of applied mods in the bottom right corner.



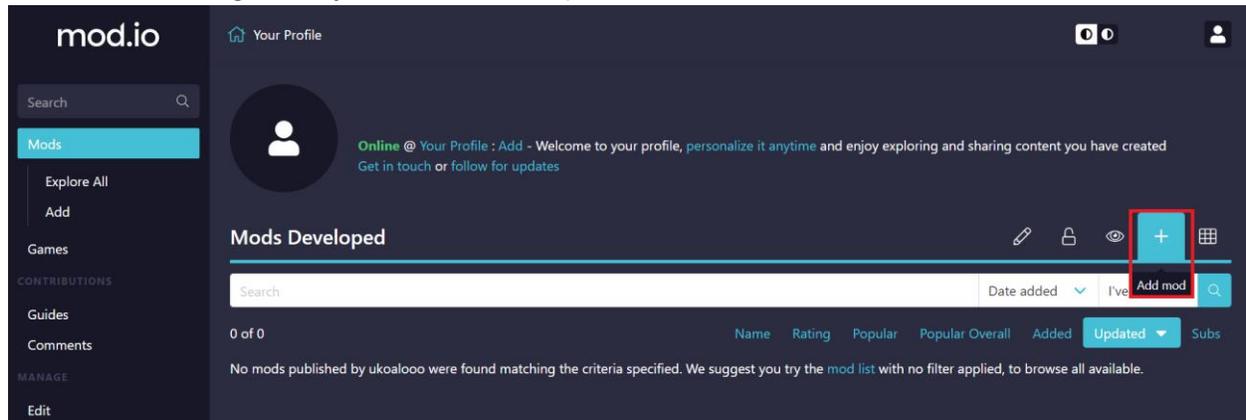


# 6 Shipping the Mod

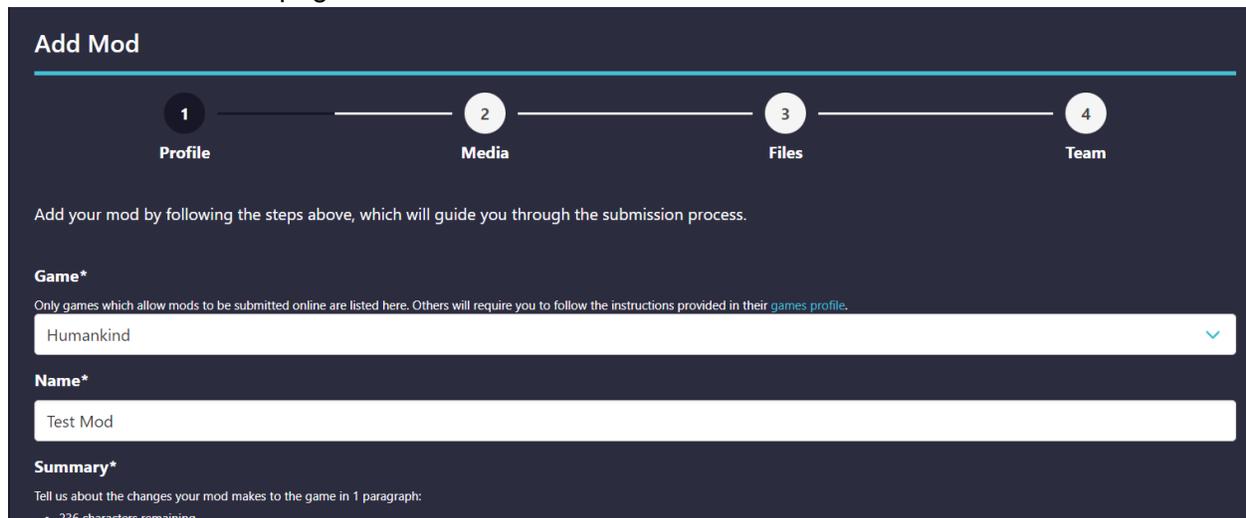
## 6.1 Shipping the mod within the site

Once the mod is ready, you can share it with others within the mod.io site:

1. Create the account on the <https://mod.io/> site.
2. Login into your account and press “Add mod”.



3. Fill all the corresponding fields and press “Save mod and next step” at the bottom of the page.



#### 4. Fill the "Media" tab and proceed or skip at the bottom of the page

### Manage Test Mod Media

Profile **2** Media Files **4** Team

Add Youtube videos, Sketchfab 3D models and images to your gallery, to properly showcase your mod.

**Images**

- Upload up to 30 images
- 8mb maximum

**Youtube videos**

Add another youtube video

**Sketchfab models**

#### 5. Attach the mod, fill the related fields and proceed.

### Manage Test Mod Files

Profile **Media** Files **3** Team **4**

Upload the latest build of your mod and manage its release history. Only one file can be the designated current release, and your mod must have at least one file to go live.

**Upload file\***

For compatibility you should ZIP the base folder of your mod, or if it is a collection of files which live in a pre-existing game folder, you should ZIP those files:

- Mods which span multiple game directories are not supported
- Mods which overwrite files are not supported
- Must be a ZIP file
- 5120mb maximum

From PC:

**Version**

We recommend a consistent version scheme like MAJOR.MINOR.PATCH

## 6. Add the teammates to moderate the mod.

**Manage Test Mod Team**

When a member accepts an invite, they will join the team with "moderator" access. Promote them to an "administrator" to give them full control over this mods profile, including managing other members access. Each level of access gives all of the rights of the lower levels, they are:

- **Moderator** - can moderate comments and content attached
- **Manager** - moderator access, including uploading builds and editing settings except supply and team members
- **Administrator** - full access, including editing the supply and team

## 6.2 Shipping the mod within Unity editor

Once the mod is ready, you can share it with others within the mod editor:

1. Go to "3. Export" of the Mod Editor tab and login into your mod.io account.

**Project** | **Mod Editor**

1. Configuration | 2. Data | **3. Export**

### Build and Run

You can build all your data into one (or multiple) archive(s), and export them in order to test your mod. If everything is working correctly, you can publish your mod.

**!** The mod has been built (11/15/2021 4:48:46 PM).

Build | Run | Build and Run

### Publish

Once you're ready, you can publish your mod.

### Account

Please login (or create a mod.io account) in order to view existing mod(s) or upload this mod.

Email Address | Security Code

Email Address  Submit

or

### Profile

Sync | Status: New | Open Website | **!** Fields marked with a \* are required.

Name\*

Project1

URL

2. Fill in the necessary fields of the mod profile.

The screenshot shows the 'Mod Editor' interface with the 'Profile' tab selected. The interface is divided into three sections: '1. Configuration', '2. Data', and '3. Export'. The 'Profile' section contains the following fields:

- Sync:** A button with a question mark icon.
- Status:** A dropdown menu currently set to 'New'.
- Open Website:** A button.
- Warning:** A yellow triangle icon followed by the text 'Fields marked with a \* are required.'
- Name\*:** A text input field containing 'Project1'. A question mark icon is to the right.
- URL:** A text input field containing 'https://humankind.test.mod.io/'. A question mark icon and a refresh icon are to the right.
- Visibility:** A dropdown menu set to 'Public'. A question mark icon and a refresh icon are to the right.
- Summary\*:** A text input field containing 'Test Mod for Food/Science district'. A character count '216 characters remaining.' and a question mark icon are to the right.
- Description:** A text input field containing 'Test Mod for Food/Science district'. A question mark icon and a refresh icon are to the right.
- Homepage:** A text input field. A refresh icon is to the right.

3. Press the "Upload to Server" button once fields are completed.

**Mod Changes**  
Tell us what changes with this update.

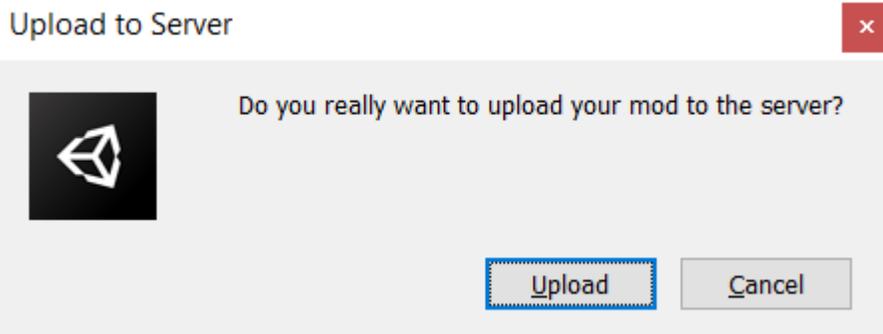
Version  ? Reset

Changelog\*

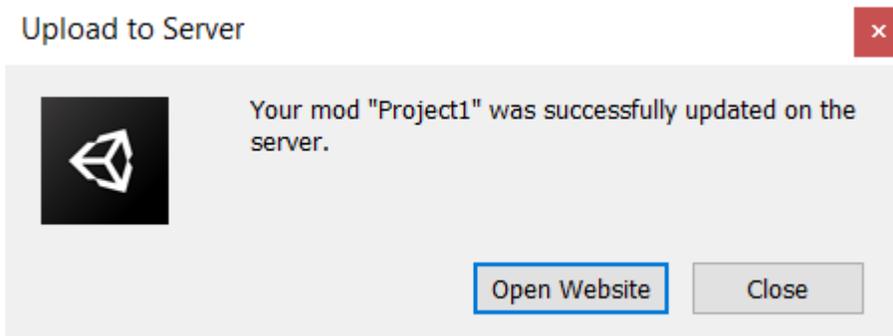
New district

Metadata

**Upload to Server**



4. Once it is uploaded to the server, go and check it on the website.



**HUMANKIND**

Search

Mods 140

PROJECT 1

- Profile
- Edit
- History 1
- Contact
- Statistics

DEVELOPMENT

- Learn more
- Add Mod

© 2021 mod.io (test env)  
About - DMCA - Terms - Privacy  
Test Environment

## Project1 for Humankind

Released Nov 16th, 2021. Ranked 9,999 of 9,999 with 0 (0 today) downloads  
Published by ukoalooo (mod ID: 6494)

---

### Description

Test Mod for Food/Science district

Mod

14.56kb

✎ 🗑 🚩

---

### Releases

Filename	Size	Version	Added	Options
132815438760228738_6494.zip	14.56kb	1.0.0	6m	<span style="font-size: 12px;">✎</span> <span style="font-size: 12px;">🔒</span> <span style="font-size: 12px;">🗑</span>

👍 👎 + 🗑

---

### Comments

Share your thoughts...

Preview
Post comment

Subscribe

Unrated 👍 0 👎 0

TEST IMAGE

+

Test environment

# Modding Tool Possibilities

With the modding tool it is possible to create different in-game objects from scratch to diversify the gameplay. In the following several chapters, it will be shown how to create or modify different game elements including:

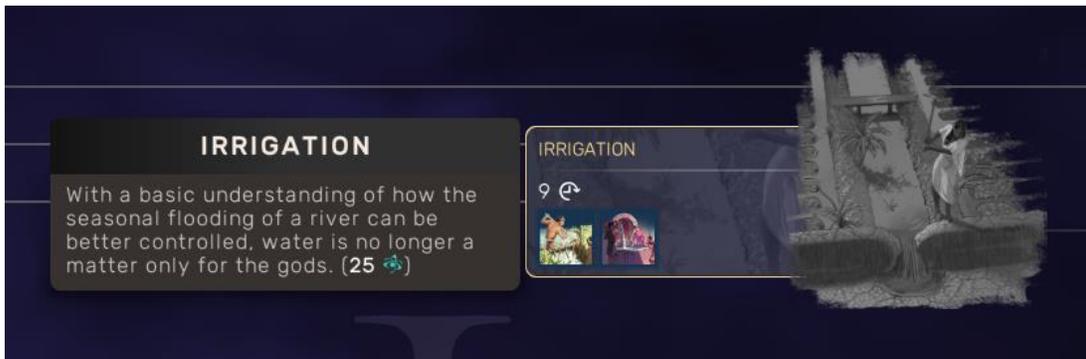
- technology;
- constructible district;
- unit;
- culture;
- narrative event;
- civic;
- battles.

## 7 Adding a new Technology

Technology is an upgrade located on the Tech Tree which players research with Science to unlock Civics, Building, Units, abilities, etc.

In the game technology consists of:

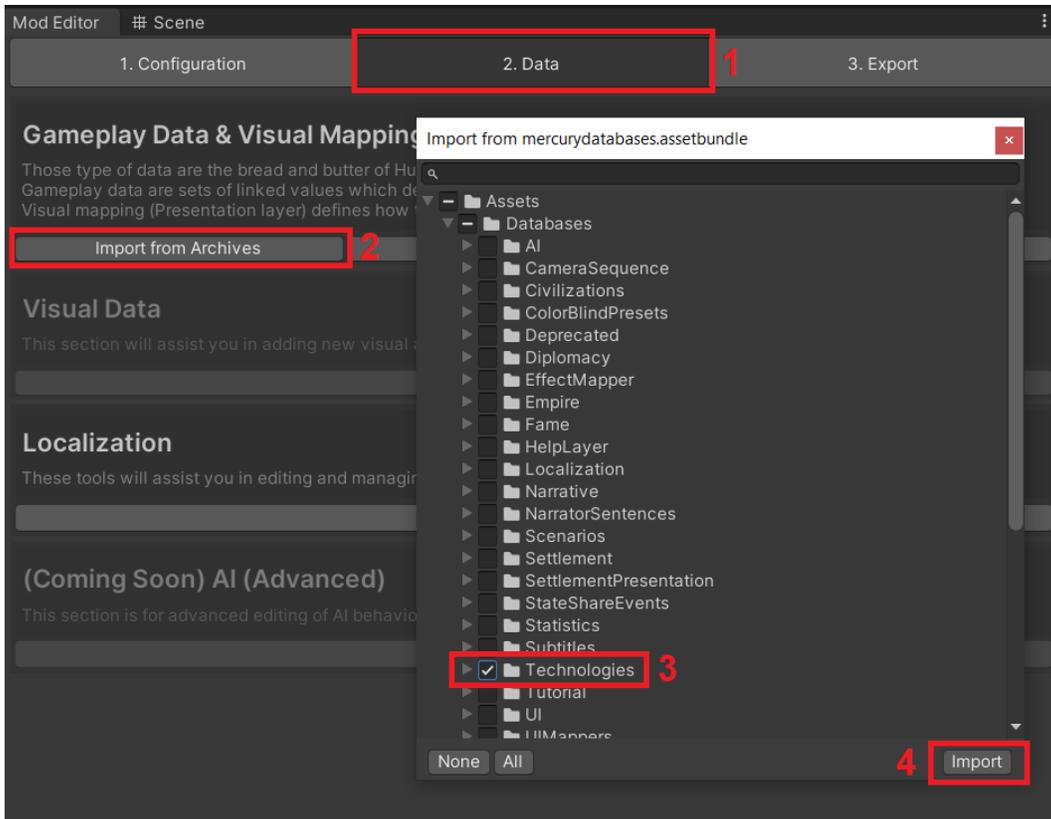
- a name;
- a description, shown in the tooltip;
- an image;
- a cost (defined by the Era Tier);
- unlocks.



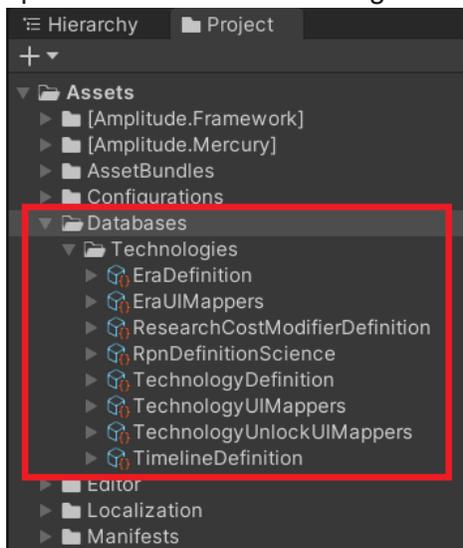
### 7.1 Setting up the environment

To modify or create a new Technology, the “Technologies” database has to be used:

1. Go to “2. Data” of the Mod Editor.
2. Select “Import from Archives”.
3. Select the “Technologies” database.
4. Press “Import”.



After you will find the exported database “Technologies” in the project structure:



The “Technologies” database consists of 8 Collections:

- “EraDefinition” has objects which define the era’s general attributes and in which era the technology is located.
- “EraUIMappers” has objects which define the era’s name, description, etc.
- “ResearchCostModifierDefinition” is used for balancing purposes.
- “RpnDefinitionScience” has objects which define the cost of the technology.

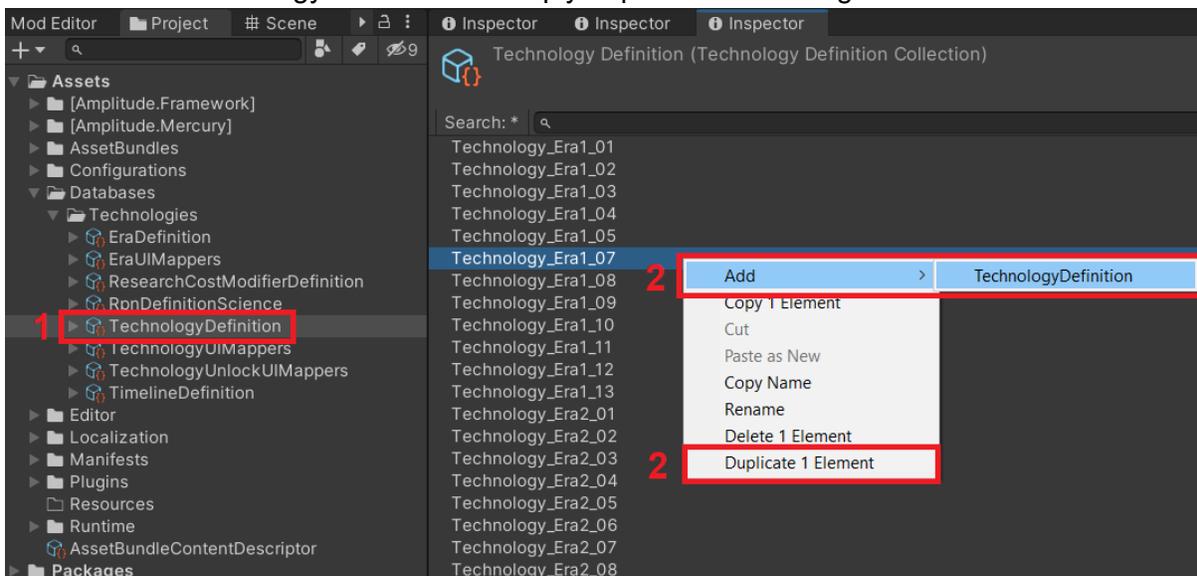
- “TechnologyDefinition” contains technology assets itself with main settings (tier, cost, prerequisites, unlocks).
- “TechnologyUIMappers” contains UI mappers for the “TechnologyDefinition” assets.
- “TechnologyUnlockUIMappers” defines the unlocks of Technologies.
- “TimelineDefinition” is unused.

**Important!** It is not recommended to store unused objects within the mod project structure. Please either not export unused objects, or remove unnecessary objects from collections/collections after the mod is created.

## 7.2 Creating the new technology

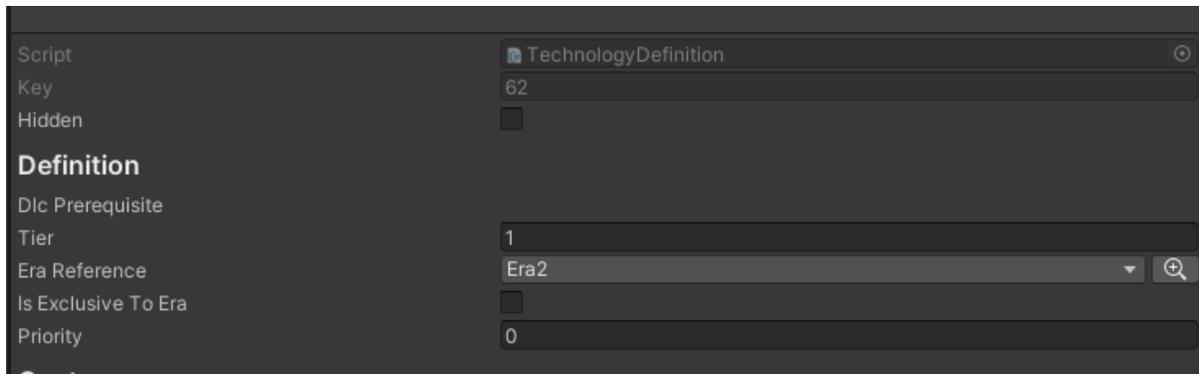
To create the new technology, a new technology definition must be created:

1. Select the “TechnologyDefinition” collection in the project structure.
2. Add a new element by right-clicking in the Inspector space and selecting “Add-> TechnologyDefinition” or simply duplicate an existing one.



## 7.3 Setting up the definition

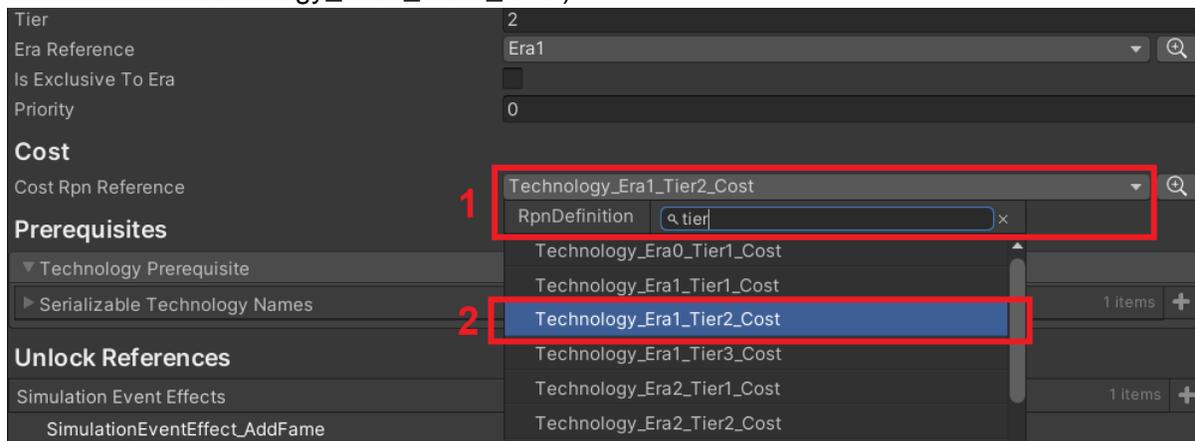
In the “Definition” area select your desired tier, era reference (in which era Technology is located), and whether it’s exclusive to an Era (better not check it). **Era Reference** will take your technology to its necessary class.



## 7.4 Setting up the cost

In the “Cost” area set up the cost for the new technology:

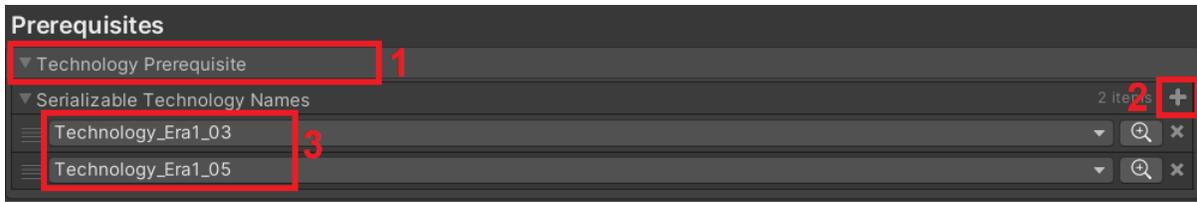
1. In the “Cost Rpn Reference” field search for “tier”.
2. Select the corresponding “Technology\_Era\_Tier\_Cost”, which will automatically define the cost for your technology. The “Cost Rpn Reference” must match era and tier from the “Definition” area (e.g Tier 2 and Era1 = Technology\_Era1\_Tier2\_Cost).



## 7.5 Setting up the prerequisites

The “Prerequisites” section defines if any Technology must be opened before the current one. It is possible to pick several prerequisite Technologies, so the current Technology will be available once at least one of the prerequisites is unlocked.

1. Expand the “Technology Prerequisite”.
2. Click plus on the “Serializable Technology Names”.
3. Select prerequisite Technologies (can be from different eras BEFORE the current one).

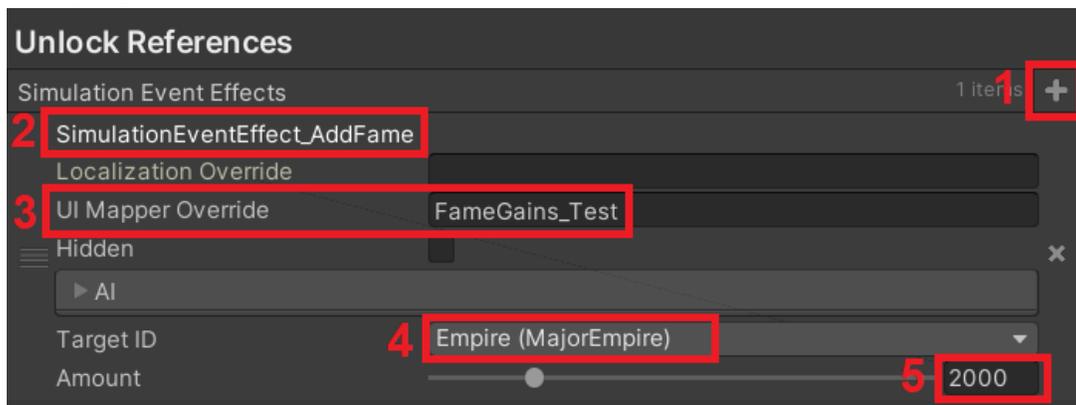


## 7.6 Setting up the unlock references

### 7.6.1 Setting up the unlock reference

To add some unlocks to the technology:

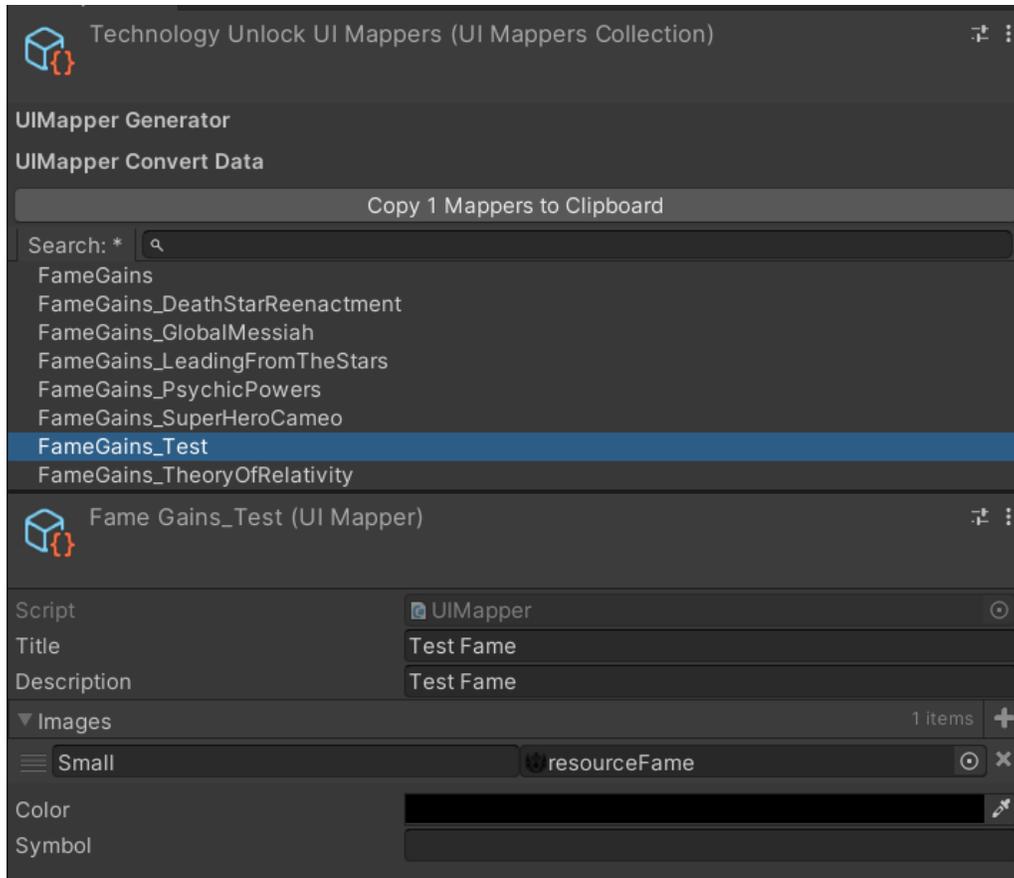
1. Click plus on the “SimulationEventEffects”.
2. Pick the preferable effect in “Plugin Editor Types” (“AddFame” for the example).
3. Set the “UI Mapper Override” field to name the reward correctly. It will be created in the next step.
4. Set the “TargetID” to “Empire (MajorEmpire)”.
5. Set the amount.



### 7.6.2 Setting up the unlock reference UI mapper

To add the correct display name the UI mapper has to be created.

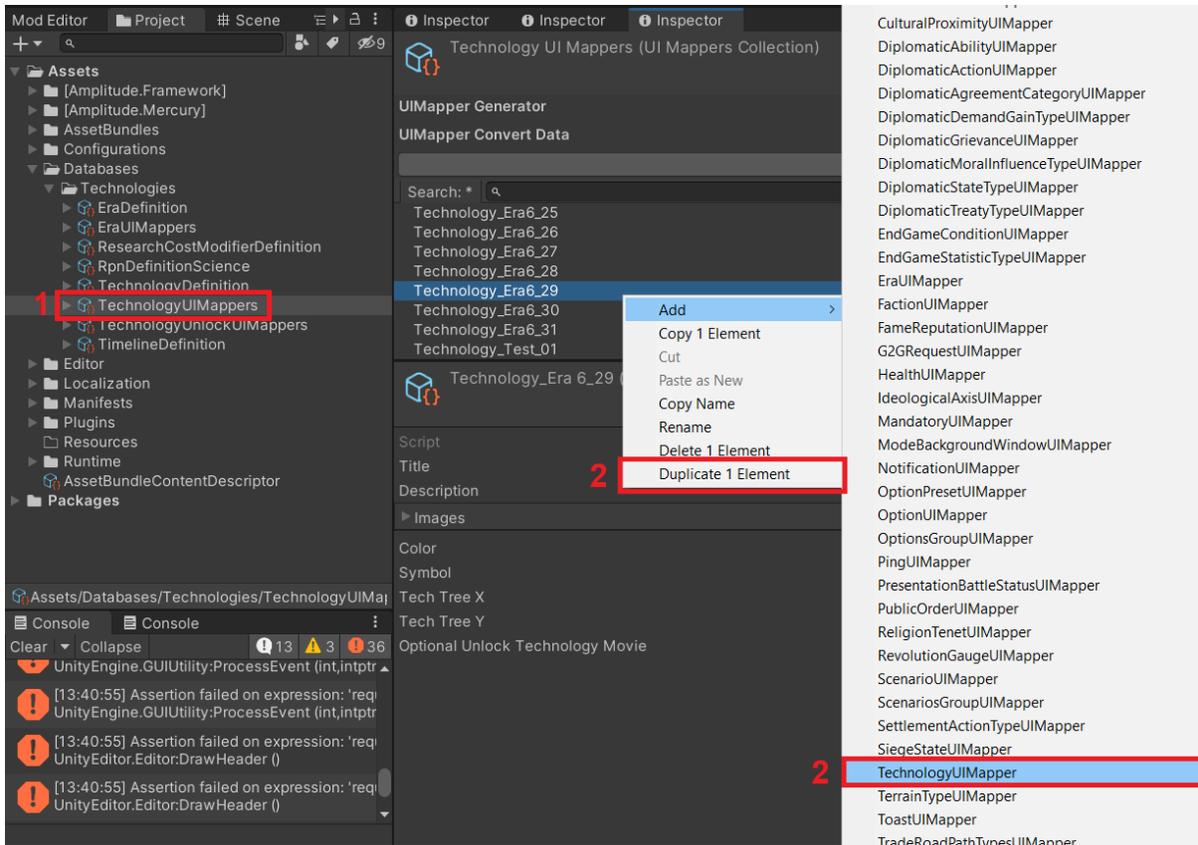
1. Go to the “TechnologyUnlockUIMappers” collection.
2. Add a new element by right-clicking in the Inspector space and selecting “Add”->”UI Mapper” or simply duplicate an existing one.
3. Rename the object to the same name from the “UI Mapper Override” field.
4. Fill corresponding fields.



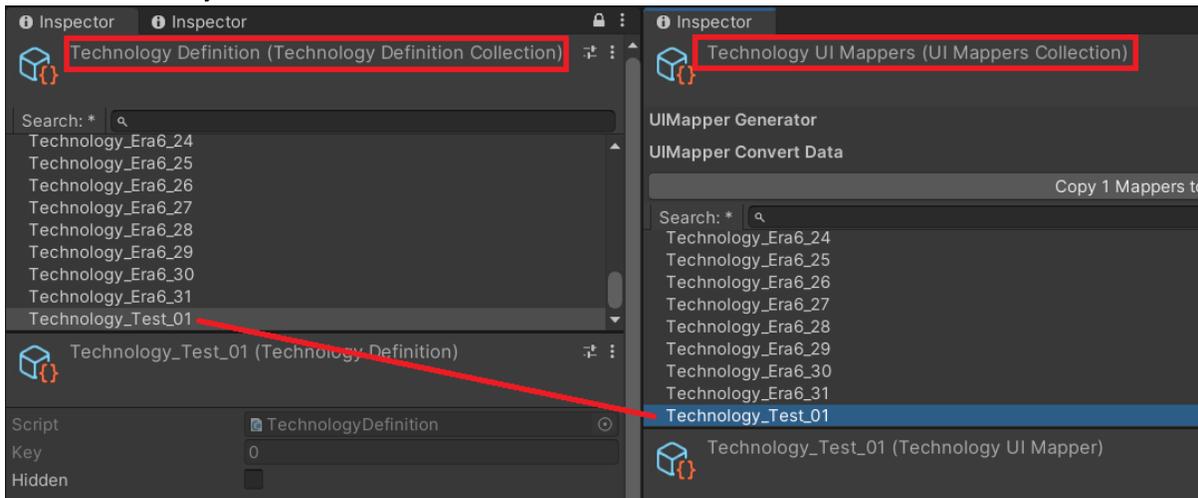
## 7.7 Mapping the technology to the Tech Tree

To display the technology on the Tech Tree correctly and assign the correct naming/description/location, the new UI mapper must be created:

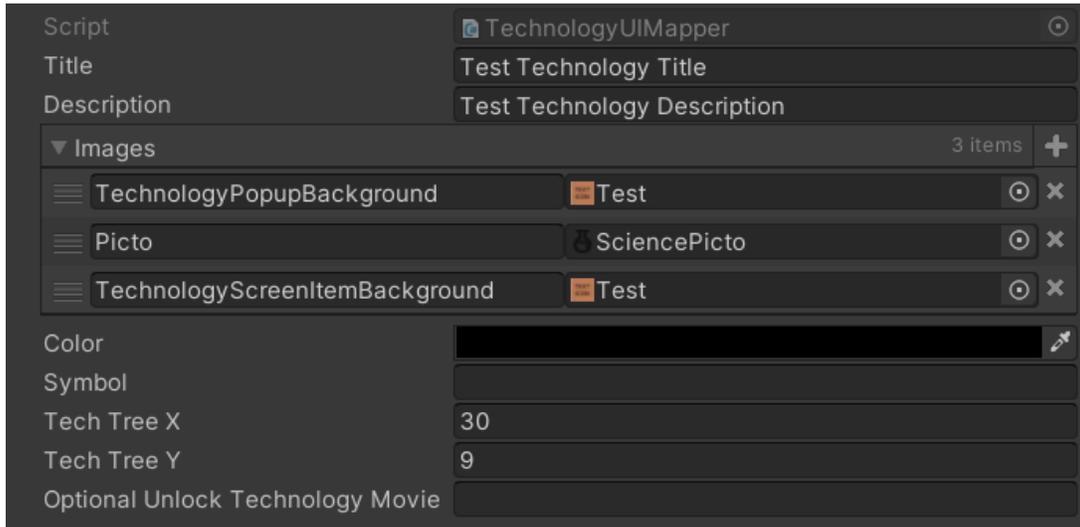
1. Go to the “TechnologyUIMappers” collection.
2. Add a new element by right-clicking in the Inspector space and selecting “Add-> TechnologyUIMapper” or simply duplicate an existing one.



3. Rename the newly created object to the same name the “TechnologyDefinition” object has. Names from 2 collections MUST match.



4. Fill the corresponding fields with pictures, name, and the location on the Tech Tree of the technology.



The X axis goes from left to right, the Y axis goes from top to bottom.



## 7.8 Testing

After all steps were done - build and start the mod. Start a new game. Go to the Tech Tree and find the new technology.



Enjoy the result. The Technology is automatically connected to its prerequisites. Once the technology is unlocked, you receive 2000 Fame.



## 8 Adding a new Constructible District

District is a city component (also known as Quarters and Extensions) that can be built on the map tile, within the chosen region limits. It produces FIMS, as well as influence, religion, stability, population limit, extracts resources, etc.



In the game District consists of:

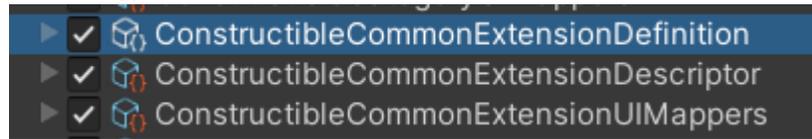
- a name;
- a description;
- an image;
- a cost;
- game effect;
- prerequisite requirements.

### 8.1 Setting up the environment

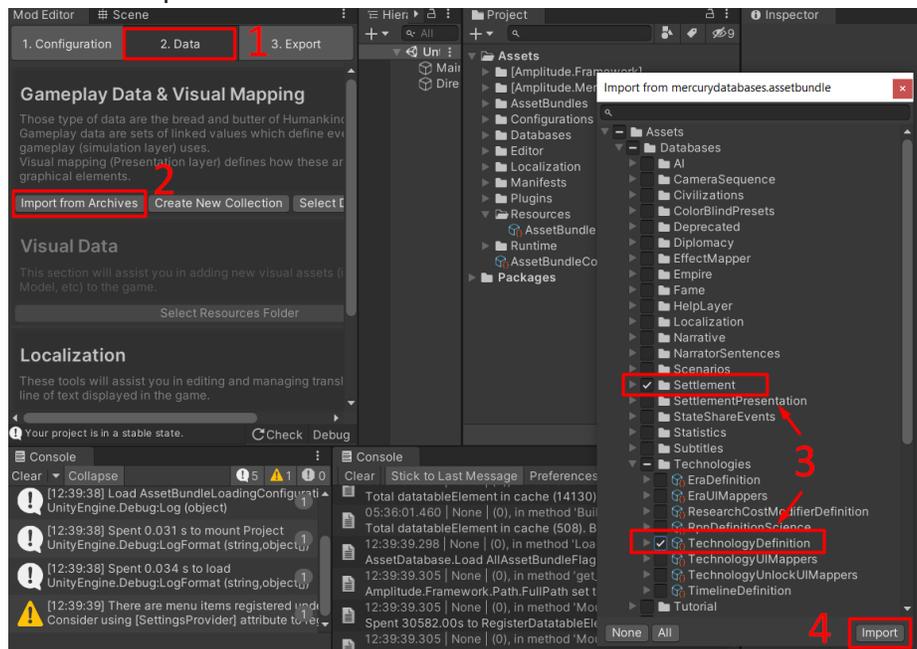
To modify or create a new constructible district, the related collections have to be used ( "Settlement" and "TechnologyDefinition" libraries):

1. Go to the "2. Data" tab of the Mod Editor.
2. Select "Import from Archives".

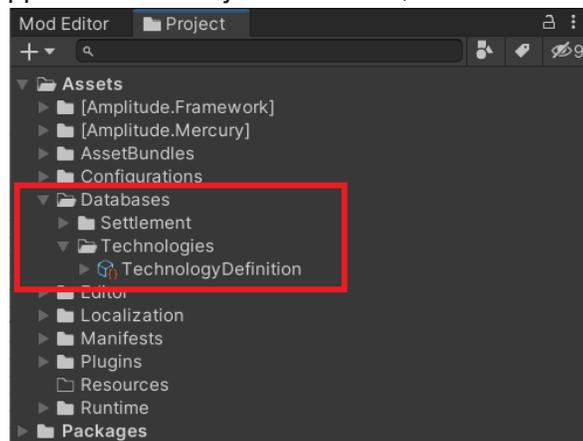
3. Select related collections from the “Settlement” database collection and “TechnologyDefinition” collection.



4. Press “Import”.



These items will appear in the “Project” structure, inside the “Databases” folder.

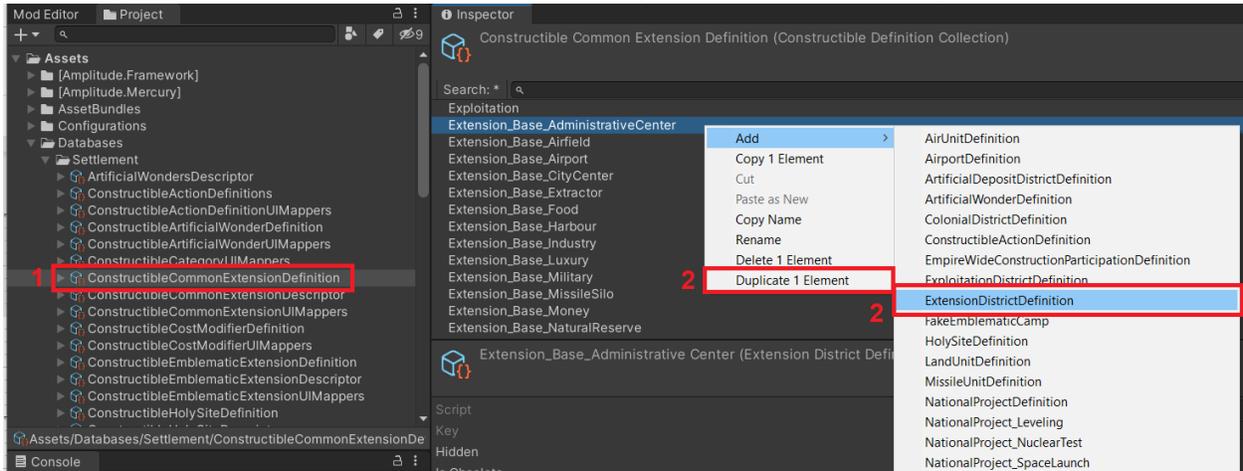


**Important!** It is not recommended to store unused objects within the mod project structure. Please either not export unused objects, or remove unnecessary objects from collections/collections after the mod is created.

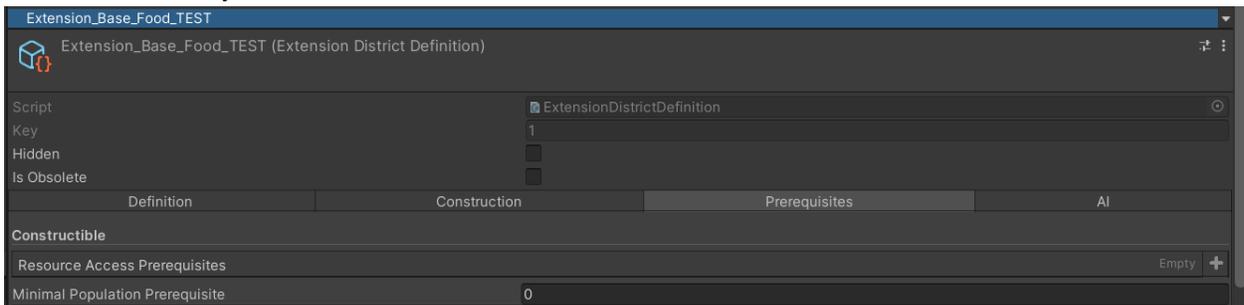
## 8.2 Creating the district definition

Definition allows setting basic properties of the future constructible district.

1. Select the “ConstructibleCommonExtensionDefinition” collection.
2. Add a new element by right-clicking in the Inspector space and selecting “Add” -> “ExtensionDistrictDefinition” or simply duplicate an existing one.



After the new definition appears in the same inspector window, it's strongly recommended to rename the object.

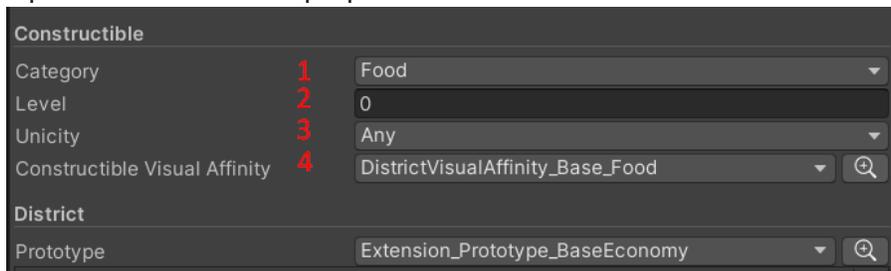


## 8.3 Setting up the “Definition” tab

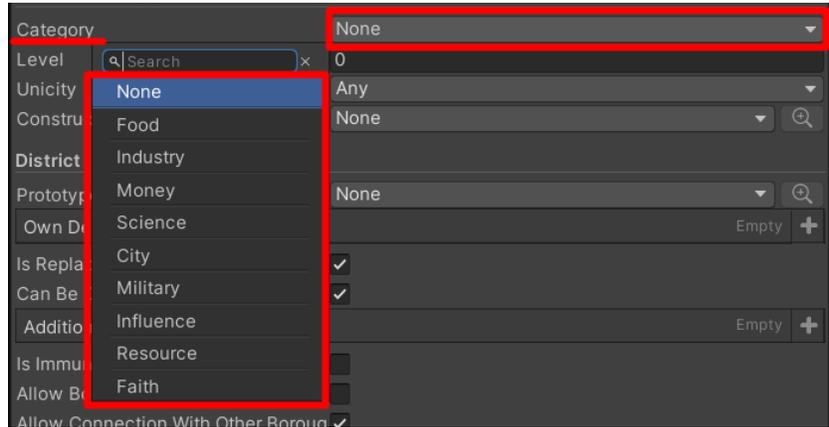
On the “Definition” tab the properties of the district can be selected.

### 8.3.1 Setting up the “Constructible” section

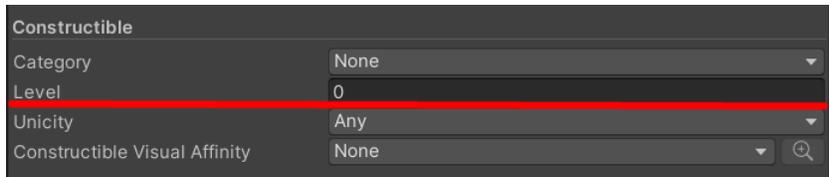
This step will set some basic properties of the district.



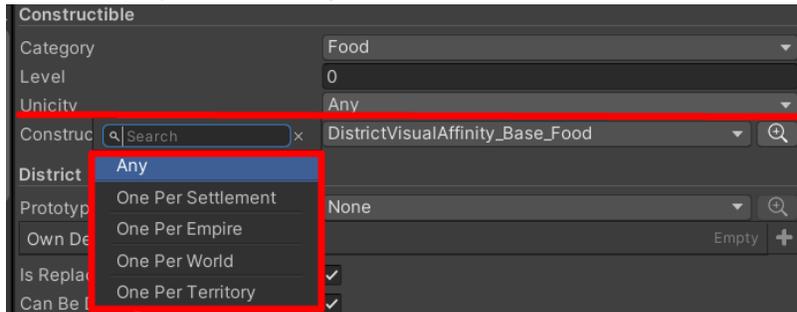
1. Select the type of the resource district will be connected with.



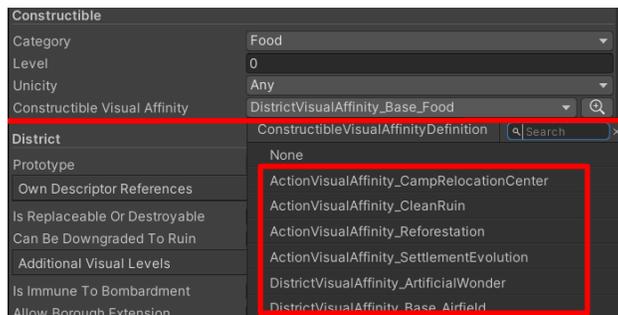
2. Leave level value at “0”. This field is used for infrastructures.



3. Set the “Unicity”, it will designate how often the district can be built.

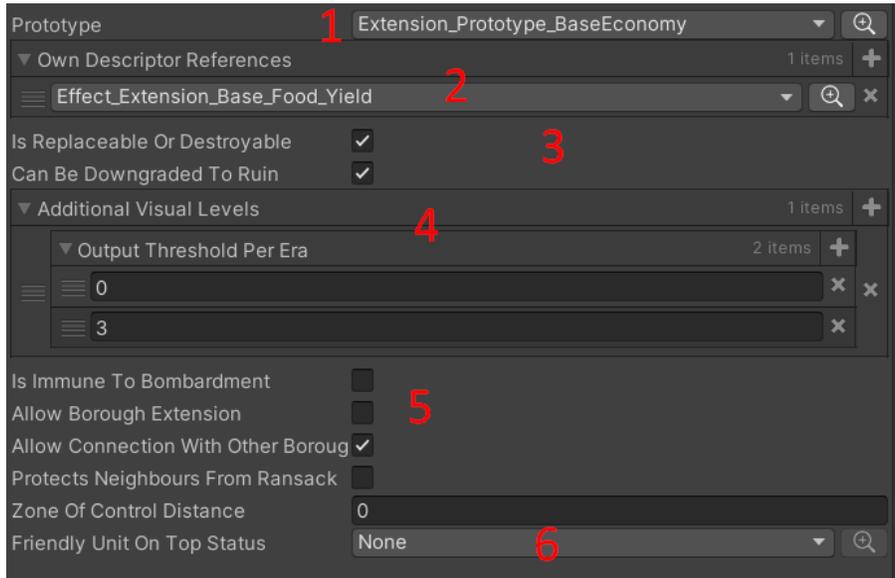


4. The “Constructible visual affinity” contains a list of visual appearances of the district. Select the related one to the district.  
**Note!** This field is not modifiable at the moment.

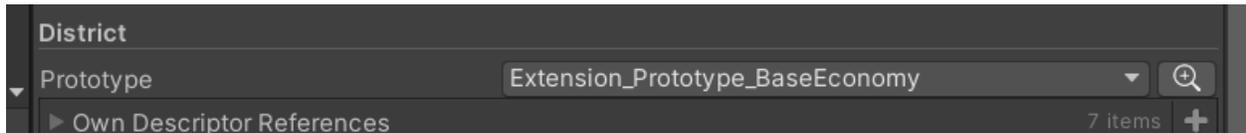


### 8.3.2 Setting up the “District” section

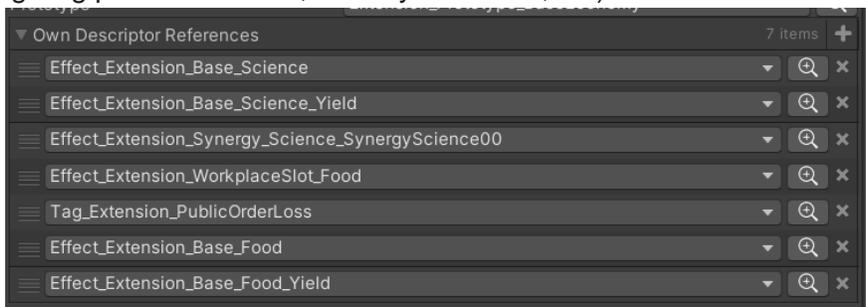
This step will help to add specific properties to the constructible district.



1. The “Prototype” defines which existing constructible definition will be used as a reference for the new one.



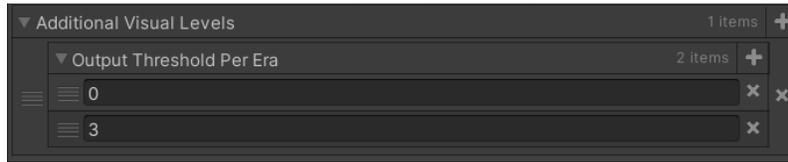
2. The “Own Description References” defines constructible district properties in addition to the already defined one from the “Prototype” field (like producing FIMS, giving production slots, stability decrease, etc).



3. The following properties define if the district can be destroyed/replaced with another or be converted into ruins.



- The “Additional Visual Levels” defines the intensity of the visual effects over the district. Value stands for the quantity of the produced resource per turn.



- In the section below, additional properties can be added.

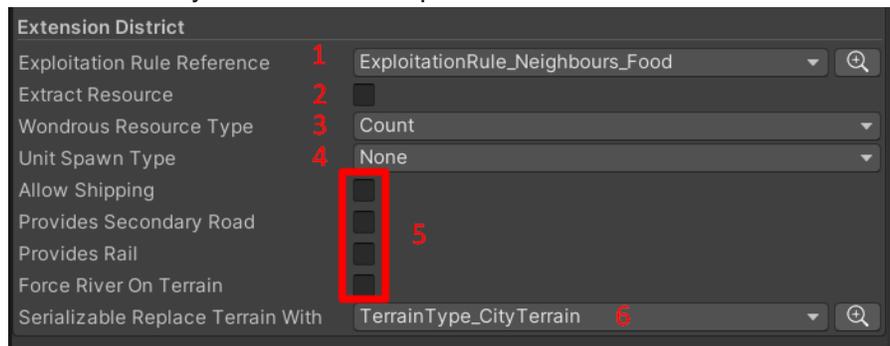


- It is possible to apply special status when a friendly unit is on the constructible district tile.



### 8.3.3 Setting up the “Extension District” section

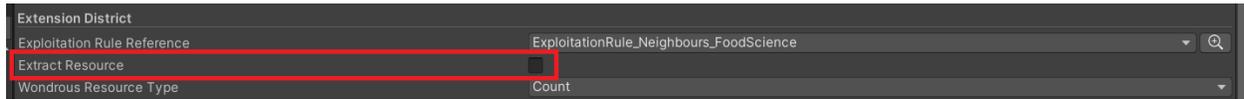
This section is mostly used to define specifics for districts that extract resources.



- The “Exploitation Rule Reference” is used for exploitation bonus from the tiles around the district.



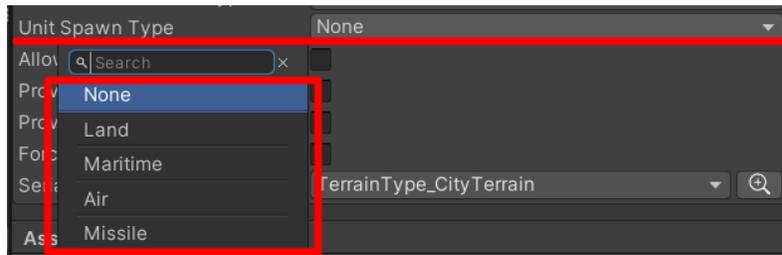
- For the extractor buildings like mines or stables, the “Extract Resource” option should be checked.



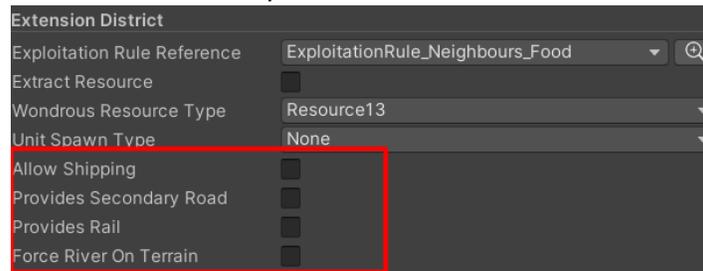
- If the building produces a wondrous type of resource, it should be chosen exactly which one. Please refer to the end of the guide for the names of the resources.



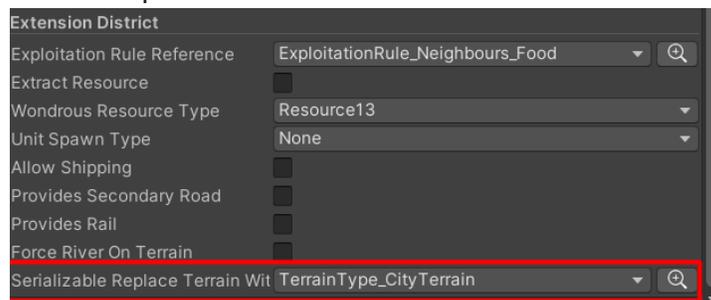
- The “Unit Spawn Type” sets the unit type, if the building is supposed to produce one.



- In this section additional options can be selected.

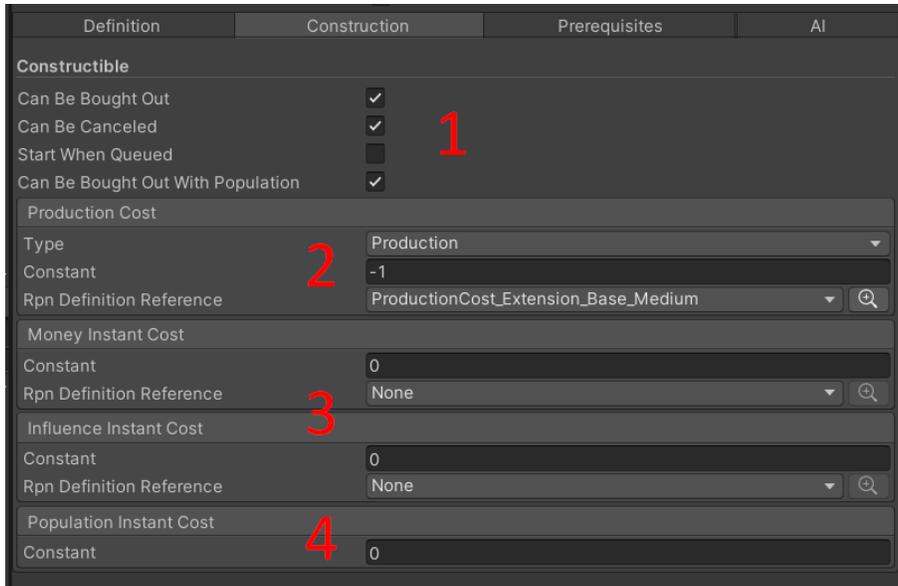


- If the district is supposed to change the tile terrain type, it may be indicated in the “Serializable Replace Terrain With”.

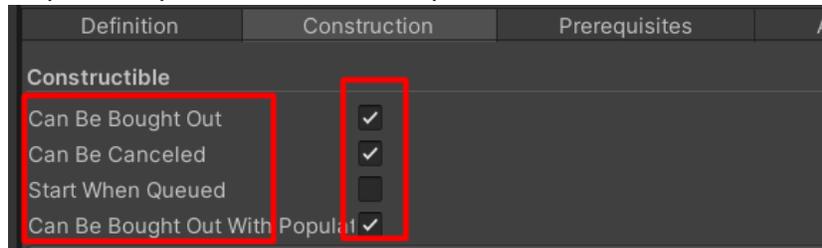


## 8.4 Setting up the “Construction” tab

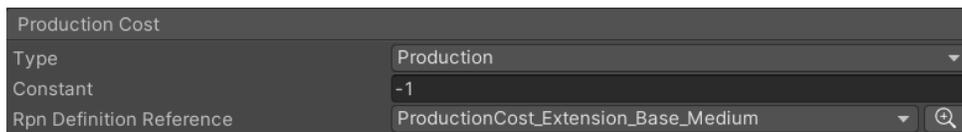
If the previous “Definition” tab was all about what a constructible district does, then the “Construction” allows setting the material requirements for its creation and the cost.



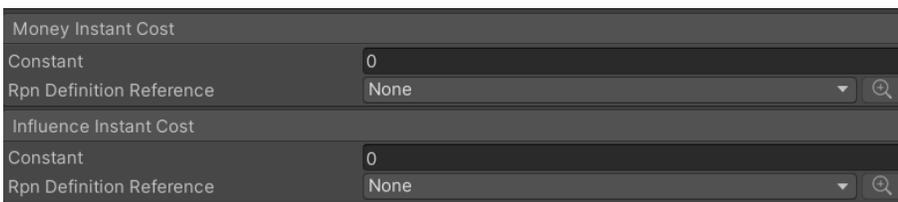
1. There are some options that could be chosen to manage the construction process. To speed it up or cancel, for example.



2. In the “Production Cost” section, the exact resource and its quantity for construction may be set. Or there is an option to choose one of the default patterns via the “RPN Definition Reference” item.



3. Same for the “Money Instant Cost” and “Influence Instant Cost”, just without the resource choice.



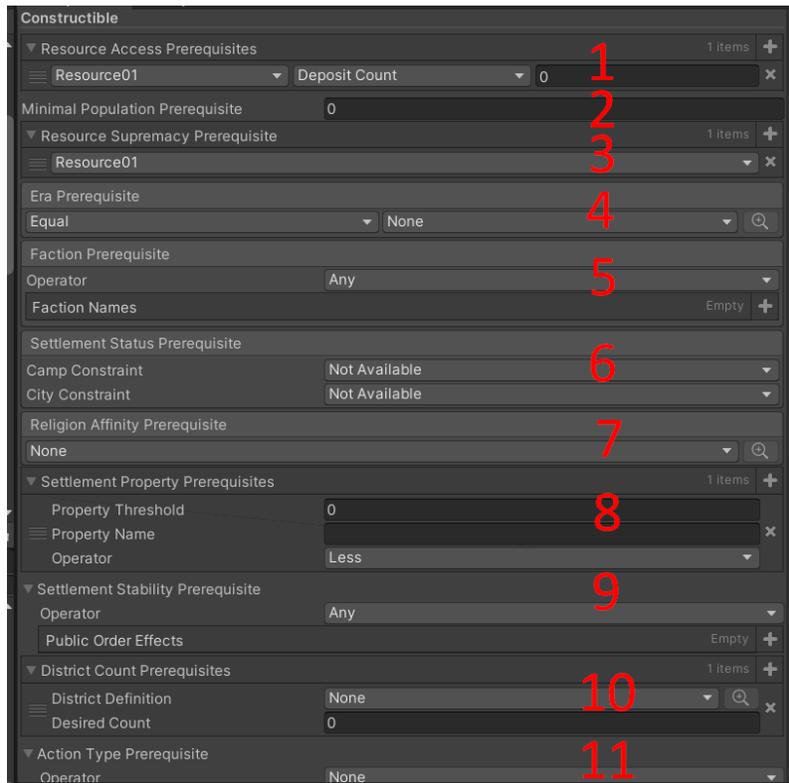
4. The “Population Instant Cost” works the same way as the previous 2 fields.



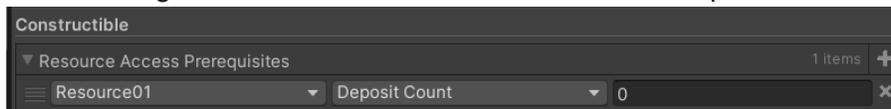
## 8.5 Setting up the “Prerequisites” tab

### 8.5.1 Setting up the “Constructible” section

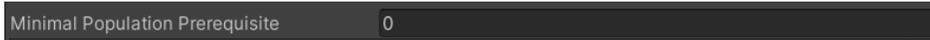
This section is responsible for defining general prerequisites for a constructible district to be created.



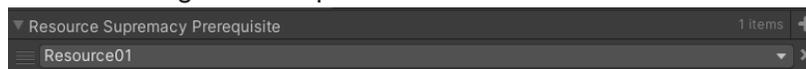
1. If a building needs access to a resource, it should be pointed here.



2. The minimum needed population quantity can be pointed in this setting item.



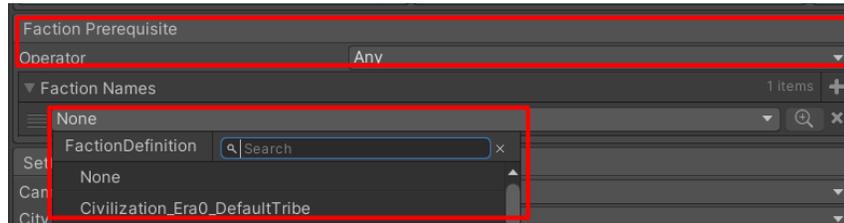
3. The “Resource Supremacy Prerequisites” means that building will be available for construction only if the player’s empire produces the biggest quantity of a certain resource among other empires.



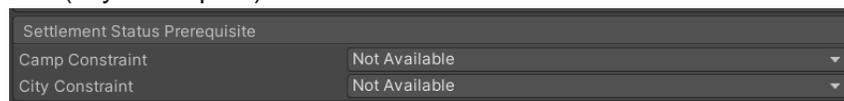
- The “Era Prerequisites” allows to designate in which era(s) building can be constructed.



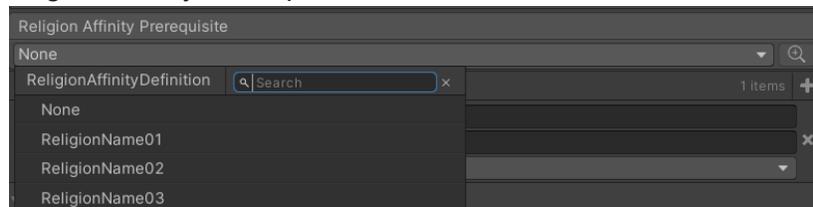
- If a building is meant to be created only by a certain civilization, then this has to be chosen in the “Faction Prerequisites” item. It is used for emblematic buildings.



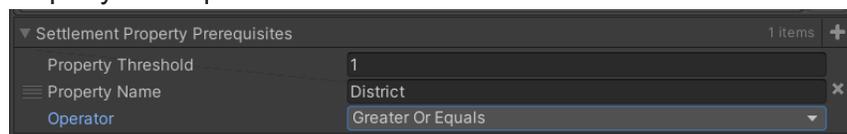
- The “Settlement Status Prerequisite” defines where a constructible district can be built (city or outpost).



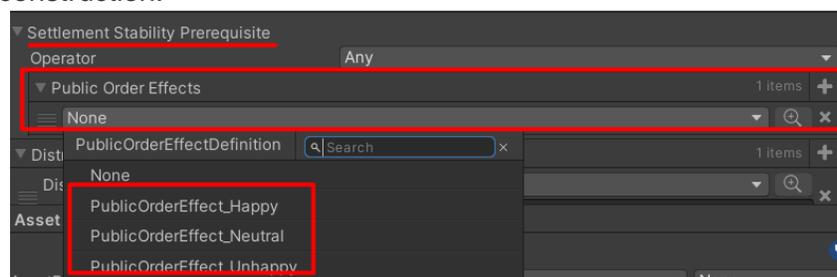
- If a building is meant to belong to a certain religion, it can be specified in the “Religion Affinity Prerequisite”.



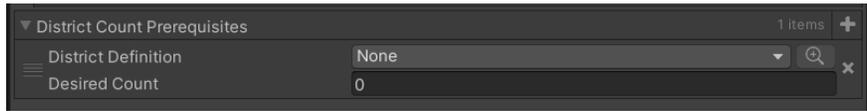
- If a building needs a certain amount of a settlement’s property type, the “Settlement Property Prerequisites” should be filled.



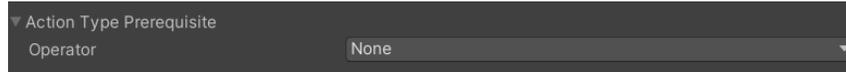
- The “Settlement Stability Prerequisites” defines the stability level required for the construction.



- The “District Count Prerequisites” specifies a certain amount of the required districts for the construction.

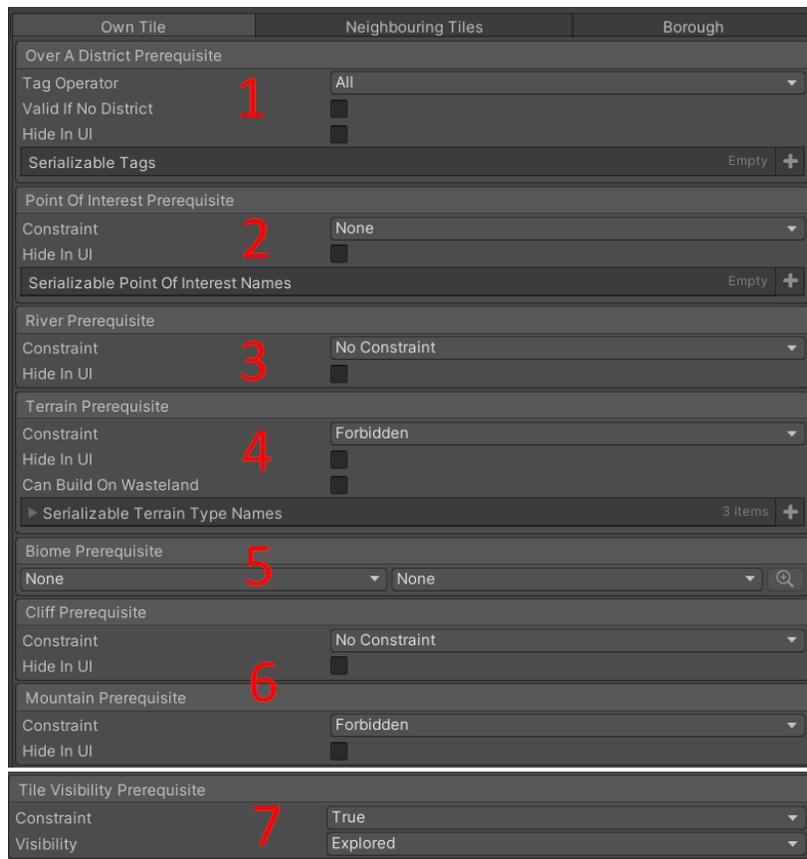


- The “Action Type Prerequisites” is recommended to leave as “None”.



### 8.5.2 Setting up the “District/Own Tile” section

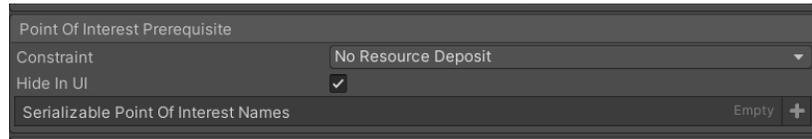
This section will help to set certain requirements for a terrain on which a constructible district can be built.



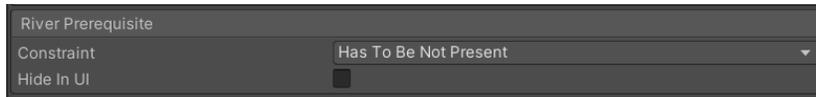
- The following example works for a vast majority of constructibles.



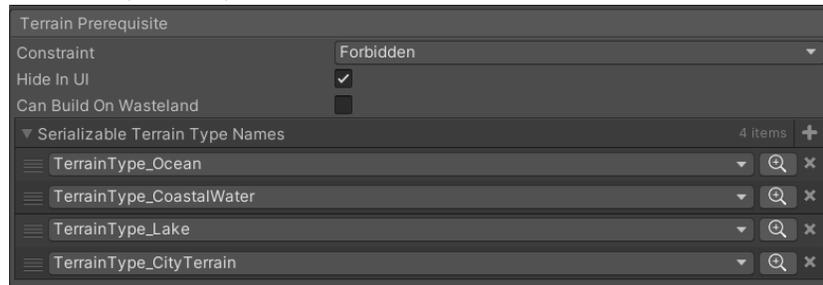
- In case a building needs to be placed on a certain resource deposit, it can be designated in the “Point of Interest Prerequisite” section.



- The “River Prerequisite” defines whether the river is needed or not for the construction.



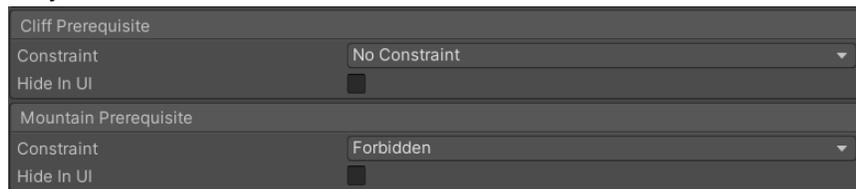
- The “Terrain Prerequisite” allows defining on which type of terrain a constructible district may or may not be built.



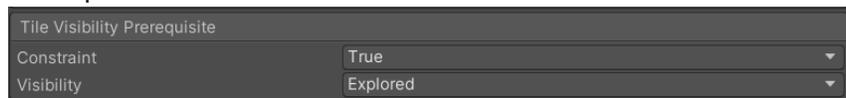
- If a building should belong to a special climate zone, then it has to be specified in the “Biome Prerequisites”.



- The “Cliff Prerequisite” and the “Mountain Prerequisite” define whether a building may be built in these zones or not.



- If land has to be explored for the construction, it may be specified in the “Visibility Prerequisite” section.



### 8.5.3 Setting up the “District/Neighbouring Tiles”

Conditions of the neighbour tiles for a constructible district may be set in this tab.

Own Tile	Neighbouring Tiles	Borough
Neighbour Tiles Prerequisite		
Neighbour Operator	Any Tile	
Territory Constraint	Same Region	
Ignore Cliff	<input checked="" type="checkbox"/>	
Terrain Prerequisite		
Constraint	None	
Hide In UI	<input type="checkbox"/>	
Can Build On Wasteland	<input type="checkbox"/>	
Serializable Terrain Type Names		
Empty +		
River Prerequisite		
Constraint	No Constraint	
Hide In UI	<input type="checkbox"/>	
District Prerequisite		
Tag Operator	All	
Valid If No District	<input type="checkbox"/>	
Hide In UI	<input type="checkbox"/>	
Serializable Tags		
Empty +		

1. Basic conditions may be set in the “Neighbour Tiles Prerequisite”.

Neighbour Tiles Prerequisite	
Neighbour Operator	Any Tile
Territory Constraint	Same Region
Ignore Cliff	<input checked="" type="checkbox"/>

2. The “Terrain Prerequisite” defines which type of terrain the neighbour tile is supposed to have.

Terrain Prerequisite	
Constraint	None
Hide In UI	<input type="checkbox"/>
Can Build On Wasteland	<input type="checkbox"/>
Serializable Terrain Type Names	
Empty +	

3. The “River Prerequisite” defines whether a river is needed next to the building tile or not.

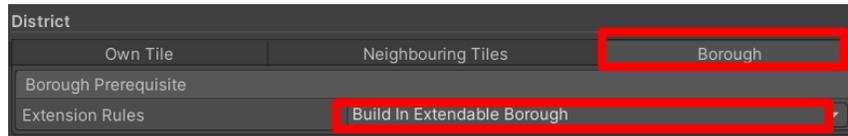
River Prerequisite	
Constraint	No Constraint
Hide In UI	<input type="checkbox"/>

4. The following example works for a vast majority of constructibles.

District Prerequisite	
Tag Operator	All
Valid If No District	<input type="checkbox"/>
Hide In UI	<input type="checkbox"/>
Serializable Tags	
Empty +	

## 8.5.4 Setting up the “District/Borough”

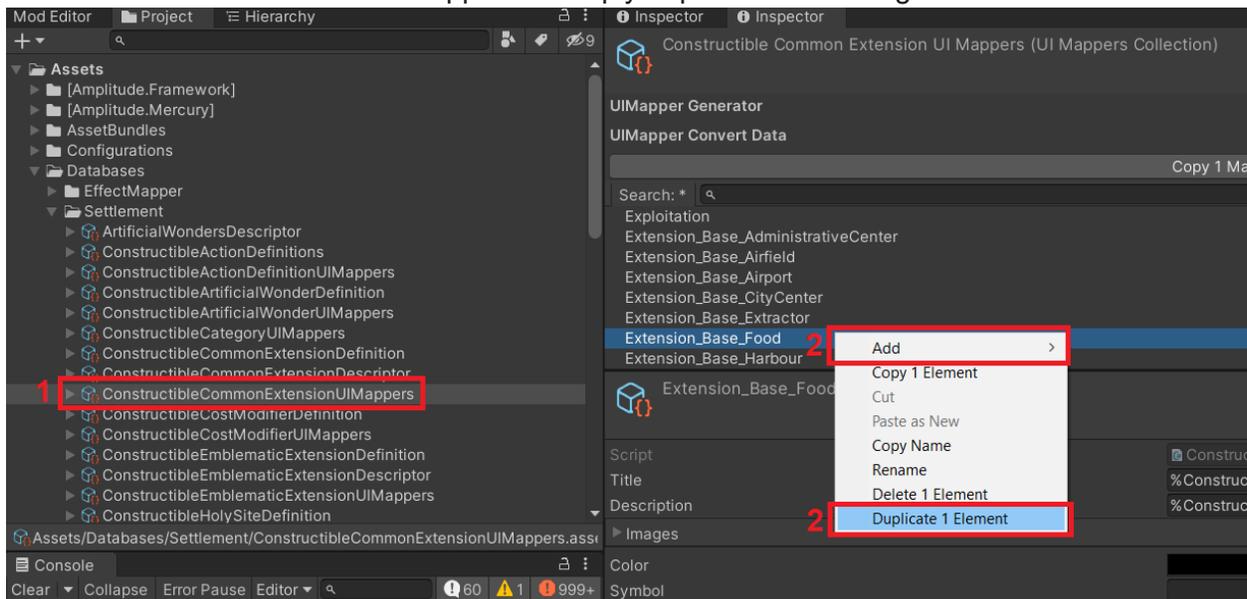
If a building is supposed to be built on borough land, then it must be specified in a certain tab.



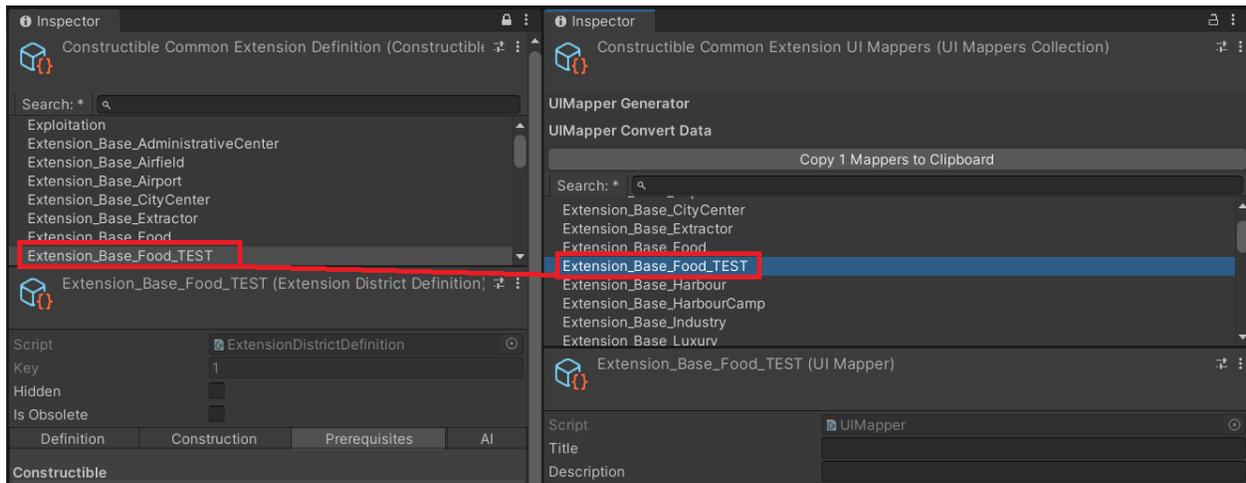
## 8.6 Setting up the UI and GeoLocalization

To display the newly created district in the game correctly, a new object in the “ConstructibleCommonExtensionUIMappers” collection has to be created.

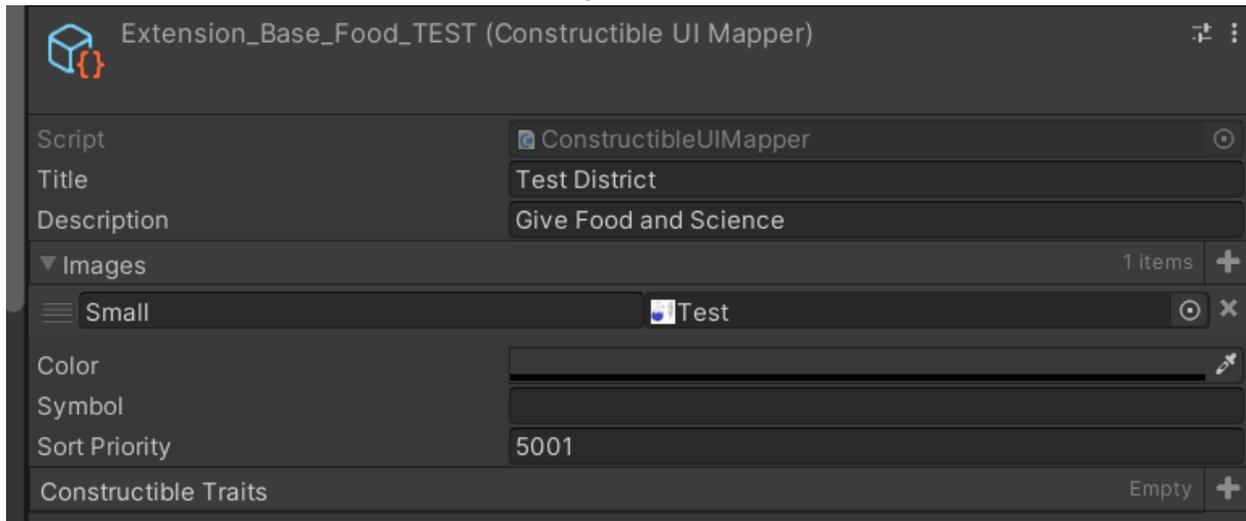
1. Select the “ConstructibleCommonExtensionUIMappers” collection.
2. Add a new element by right-clicking in the inspector space and selecting “Add” -> “ConstructibleUIMapper” or simply duplicate an existing one.



- Rename the new UI object to the same name the district definition object has. Names from 2 collections MUST match



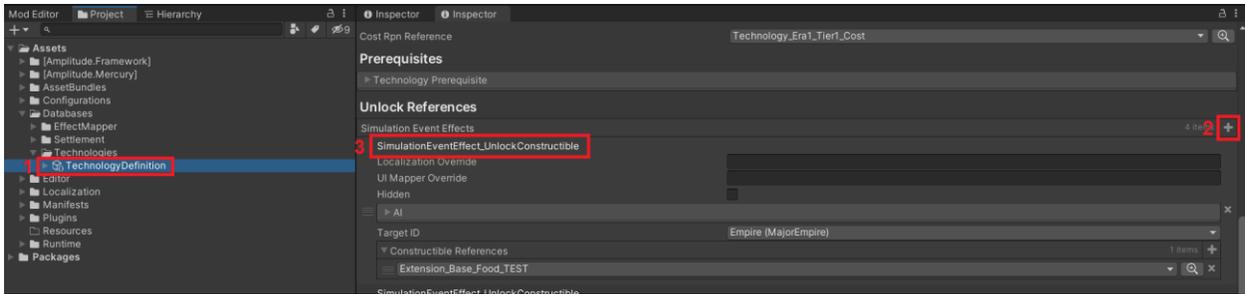
- Fill the related fields for the object.



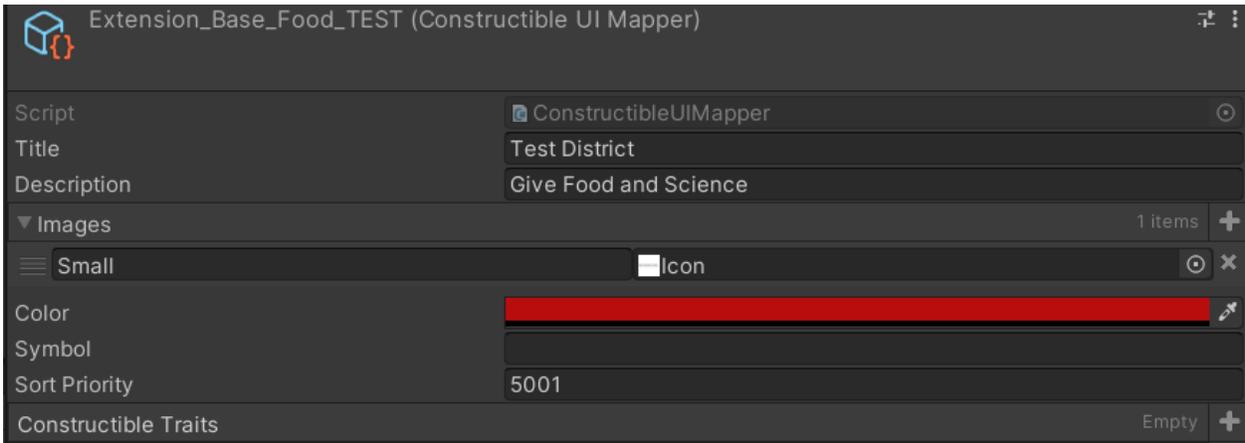
## 8.7 Adding the constructible district into the Tech Tree

In order to unlock the building in the game within a Tech Tree, it has to be added into one of the existing technologies. Be sure that the era of the technology corresponds with the building's era prerequisites.

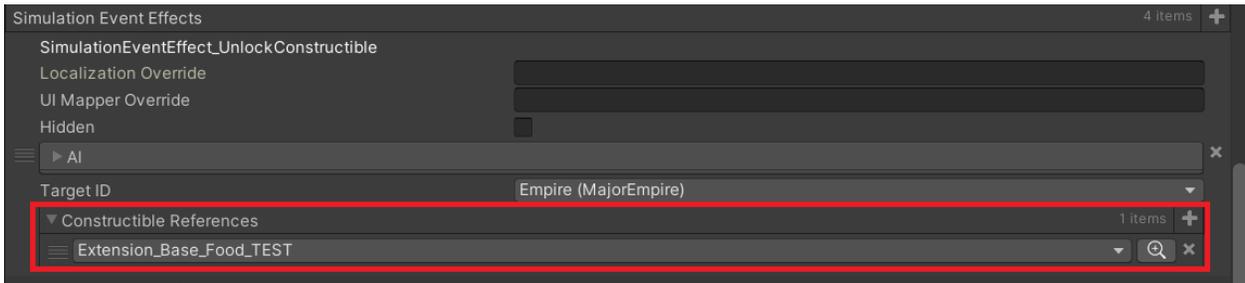
- Go to "Technologies" -> "Technology Definition" inside the "Project" panel.
- Add an item in the "Simulation Event Effects".
- Choose the "SimulationEventEffect\_UnlockConstructible".



- Find the newly added item, click on the “Constructible References” field and choose the “DatatableElementReference”.



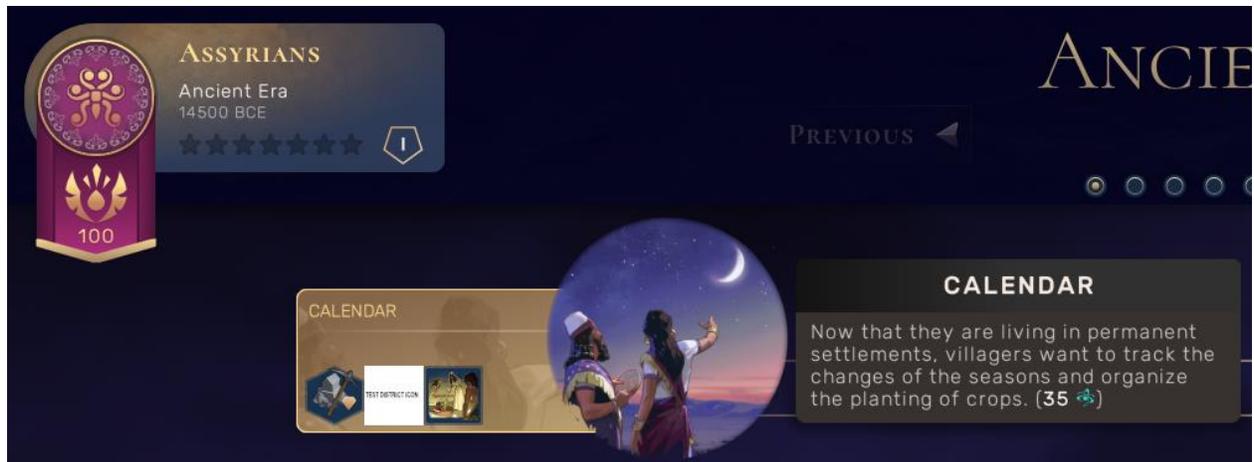
- Add an item in the “Constructible References” then select the constructible district created earlier from the drop-down menu.



## 8.8 Testing

After all steps were done - click “Build and Run”. Start a new game and go to the needed era.

Open “Technologies” and find the modified one. The constructible district will be shown in the technology card. Point a cursor on it and a name given in the modding tool will appear.



Research this technology and fulfill other prerequisites set for this building. It will become available for construction. Build it and check if the properties correspond with the ones in the modding tool.



If everything is okay, then the constructible district was successfully created.

# 9 Adding a new Unit

Units - are moveable entities on the map that can perform certain functions (move around the global map, visit points of interest, and fight enemy units).



There are three types of units in the game:

- Air.
- Naval.
- Land.

Each unit has a set of parameters that affect its movement speed, effectiveness in battle, and the presence of unique abilities:

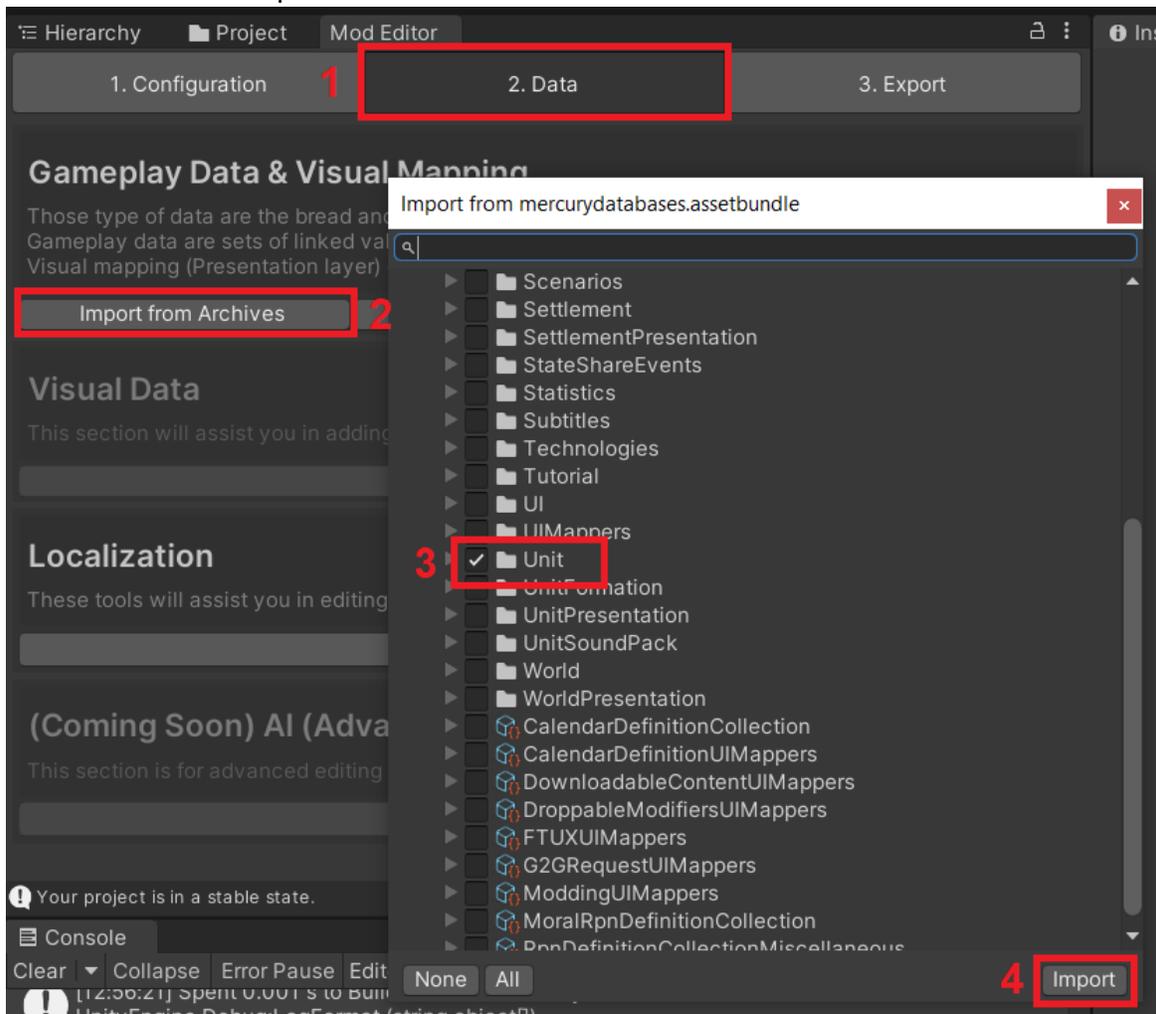
- combat power;
- move points;
- attack range;
- veterancy;
- class and specialty;
- upkeep;
- name, description, and image;
- cost.

## 9.1 Setting up the environment

To modify or create a new Unit, the "Unit" database has to be exported:

1. Go to "2. Data" of the Mod Editor.
2. Select "Import from Archives".
3. Select "Unit".

4. Press “Import”.

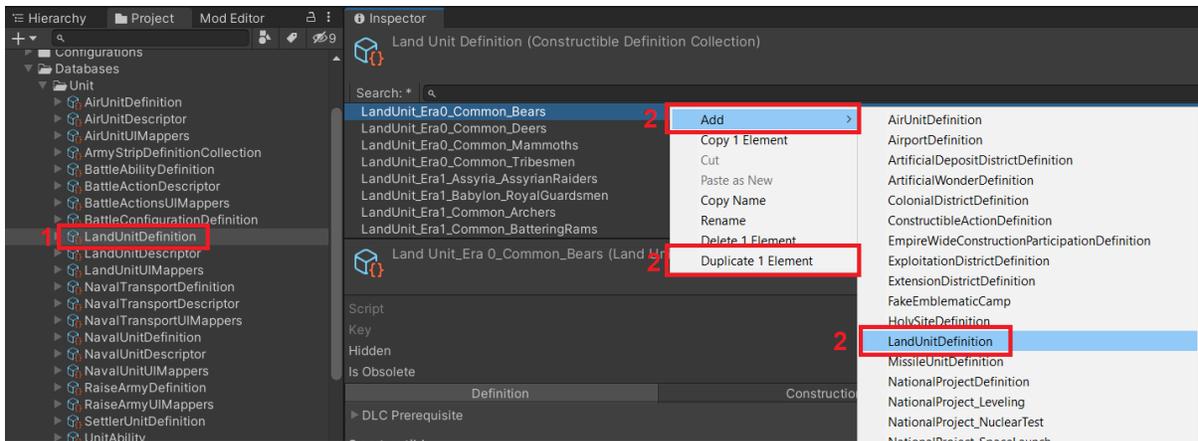


**Important!** It is not recommended to store unused objects within the mod project structure. Please either not export unused objects, or remove unnecessary objects from collections/collections after the mod is created.

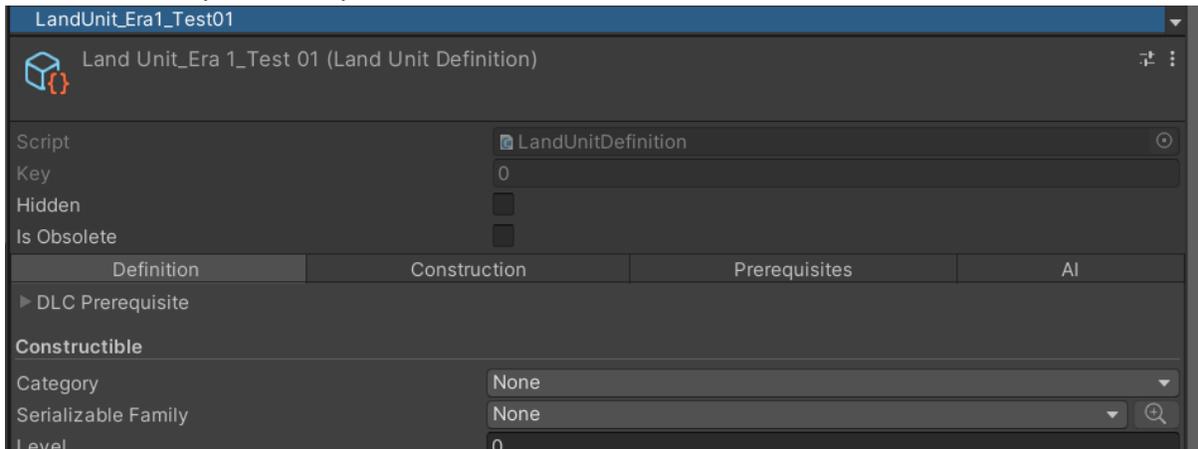
## 9.2 Creating the new unit

In order to create a new unit, a new unit definition object has to be created.

1. Decide which type of a unit is going to be created or modified and select the corresponding collection (in our case the collection is “LandUnitDefinition”).
2. Add a new element by right-clicking in the inspector space and selecting “Add-> LandUnitDefinition” or simply duplicate an existing one.



After the new definition appears in the same inspector window, it's strongly recommended to rename the object. The newly created unit has a similar structure as a constructible district described in a separate chapter.

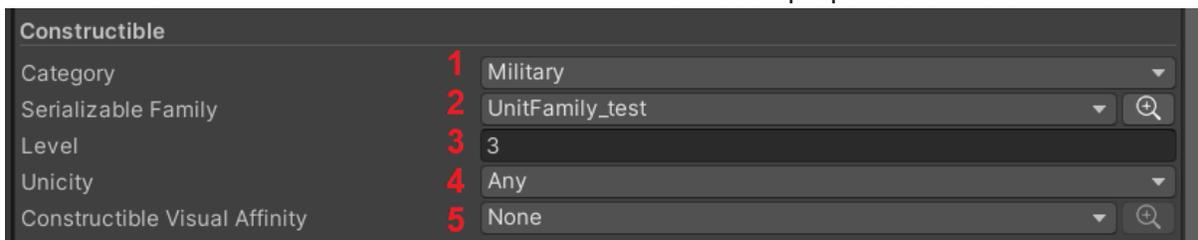


## 9.3 Setting up the “Definition” tab

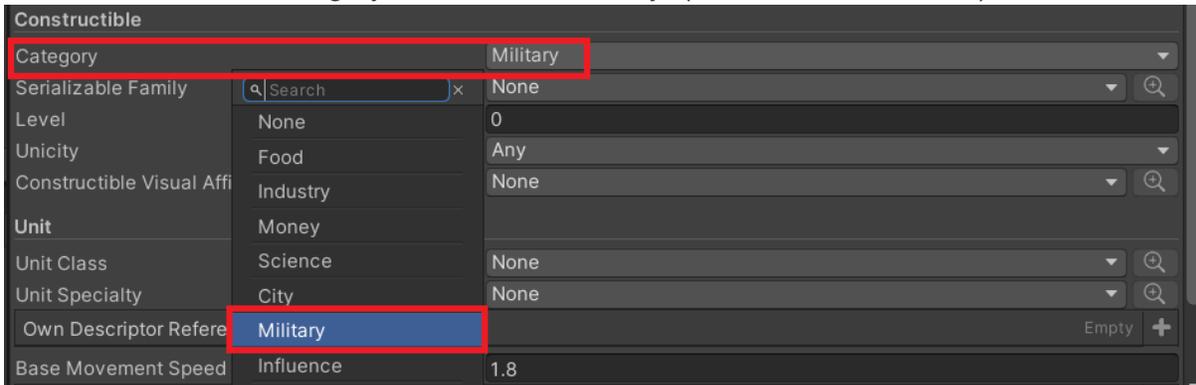
On the “Definition” tab the properties of the unit can be selected.

### 9.3.1 Setting up the “Constructible” section

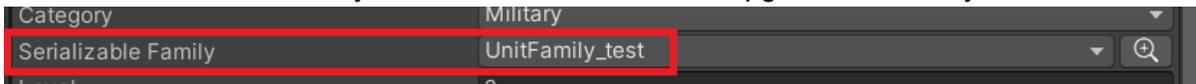
The “Constructible” section defines some basic hidden properties of the unit.



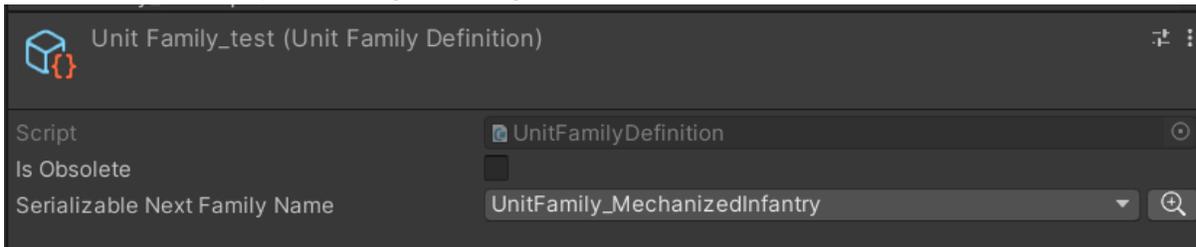
1. Set the category of the unit to “Military” (must be used for units).



2. Set the “Serializable Family” to the preferable one. This field defines how the unit will be upgraded in the modern era and not visible to the player. Please refer to the “UnitFamilyDefinition” collection for the upgrades hierarchy.



The “UnitFamily\_test” example was duplicated from the existing one (unit will be upgraded to mechanized infantry on later game stages).

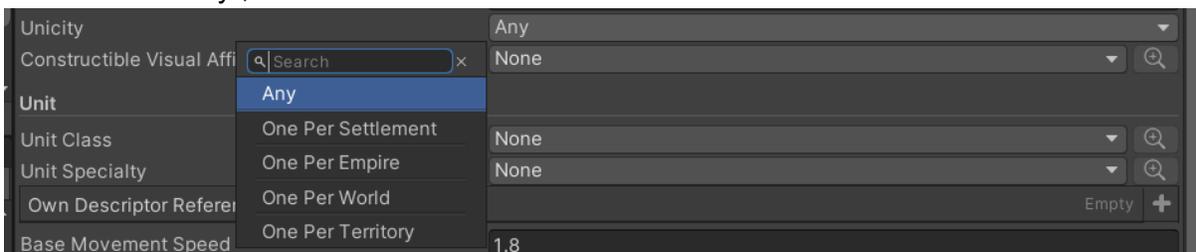


If you don't want it to upgrade the unit to anything, set the value to “None”.

3. Leave level value at “0”.



4. Set the “Unicity” defining how many new units can be built per game. If it's set to “Any”, it can be built an infinite amount of times.



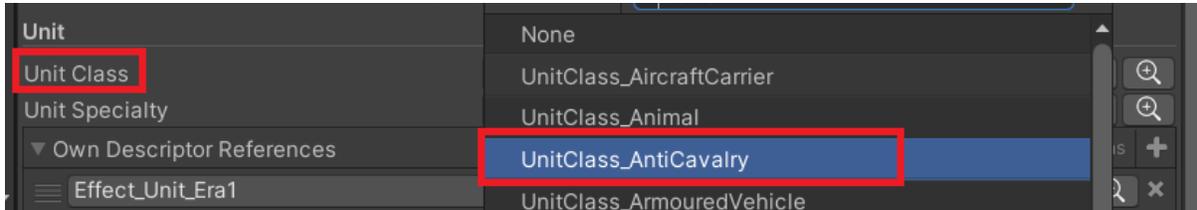
5. Set the “Constructible visual affinity” to “None”. This field contains a list of visual appearances of the object and is used for districts.



### 9.3.2 Setting up the “Unit” section

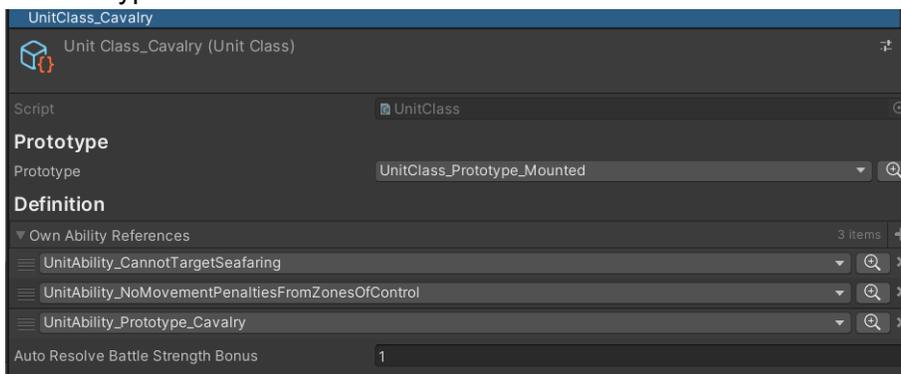
The “Unit” section defines some basic properties of the unit visible to the player.

1. Select the “Unit Class” of the unit. The class defines which gameplay capabilities a unit has and is displayed to the player.

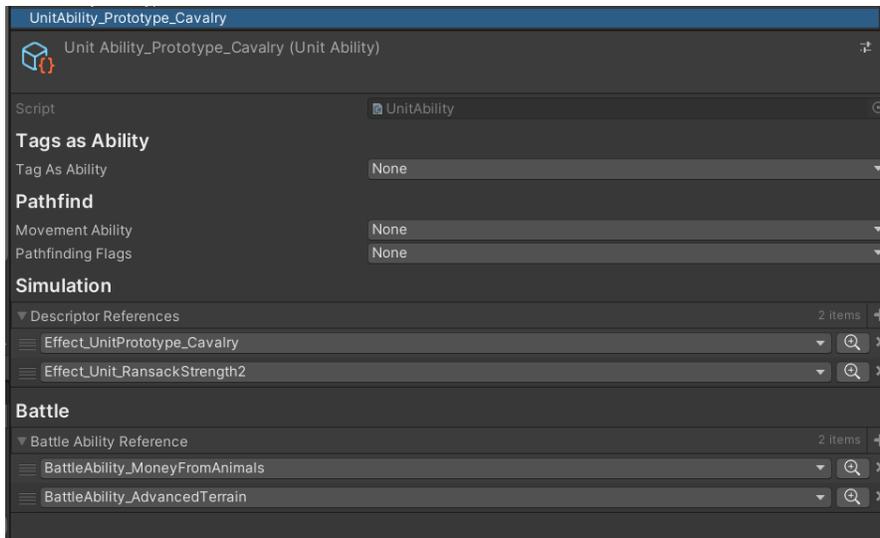


All the classes are stored in the “UnitClass” collection. The class consists of:

- Prototype - inherits definition from another class. Example of the prototype inheritance: “UnitClass\_Cavalry” -> “UnitClass\_Prototype\_Mounted” -> “UnitClass\_Prototype\_LandUnit” -> “UnitClass\_Prototype\_Unit”.
- Definition - gives new abilities to the class in addition to the inherited one from “Prototype” section.



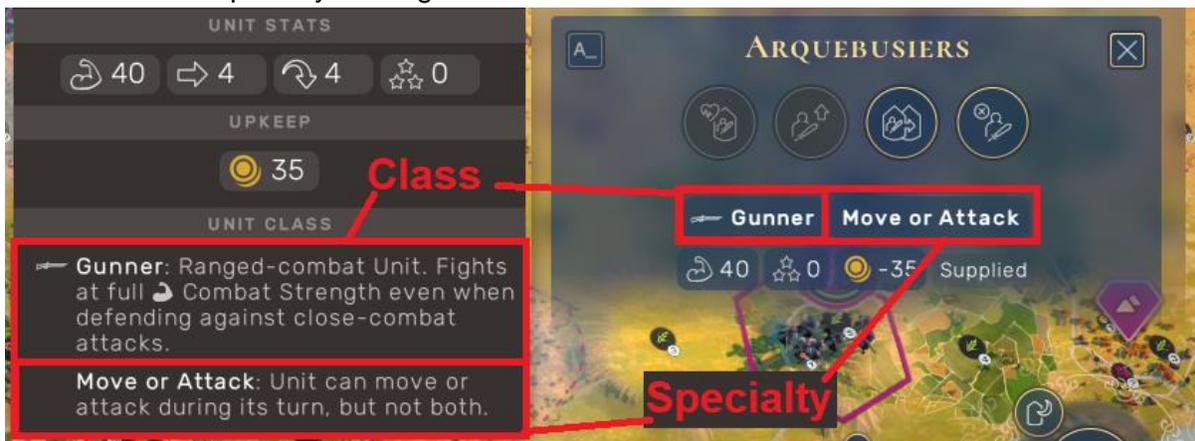
If the new class is required, the UI mapper must be created as well. The class abilities are stored in the “UnitAbility” collection.



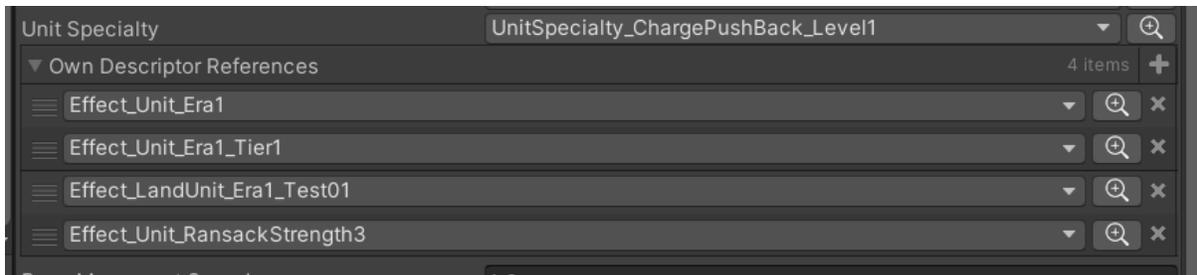
2. Select the “Unit Specialty” which defines some extra characteristics and abilities of the unit.



Class and specialty in the game:



3. Assign the “Own Descriptor References” to the unit. It is used to provide the unit with some in-game data like upkeep, strength, spoil of wars, etc. Each unit has its own descriptor and some common one.



The "Effect\_Unit\_Era1" adds spoils of war and can be found in the "UnitDescriptor" collection.

The "Effect\_Unit\_Era1\_Tier1" adds some upkeep and can be found in the "UnitDescriptor" collection.

The "Effect\_LandUnit\_Era1\_Test01" adds strength, leadership, range attack, etc to the unit. Existing unit descriptors are stored in the "Land/Naval/AirUnitDescriptor" collections.

The "Effect\_Unit\_RansackStrength3" gives additional strength for ransack and can be found in the "UnitDescriptor" collection.

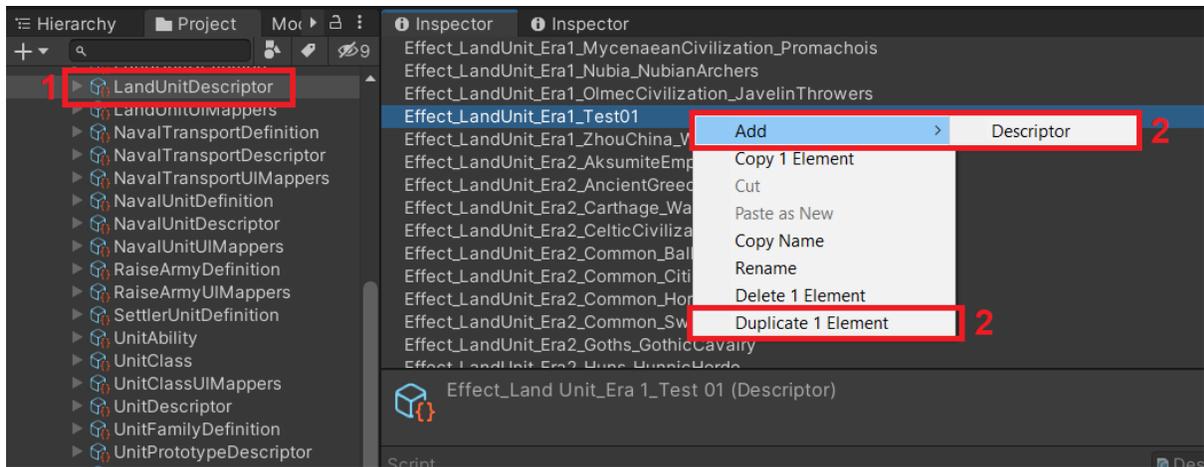
4. The "Base Movement Speed" is defined by default and must not be edited.
5. The "Visual Affinity Era Reference" defines to which Era unit belongs in terms of look.



### 9.3.3 Creating a unit descriptor

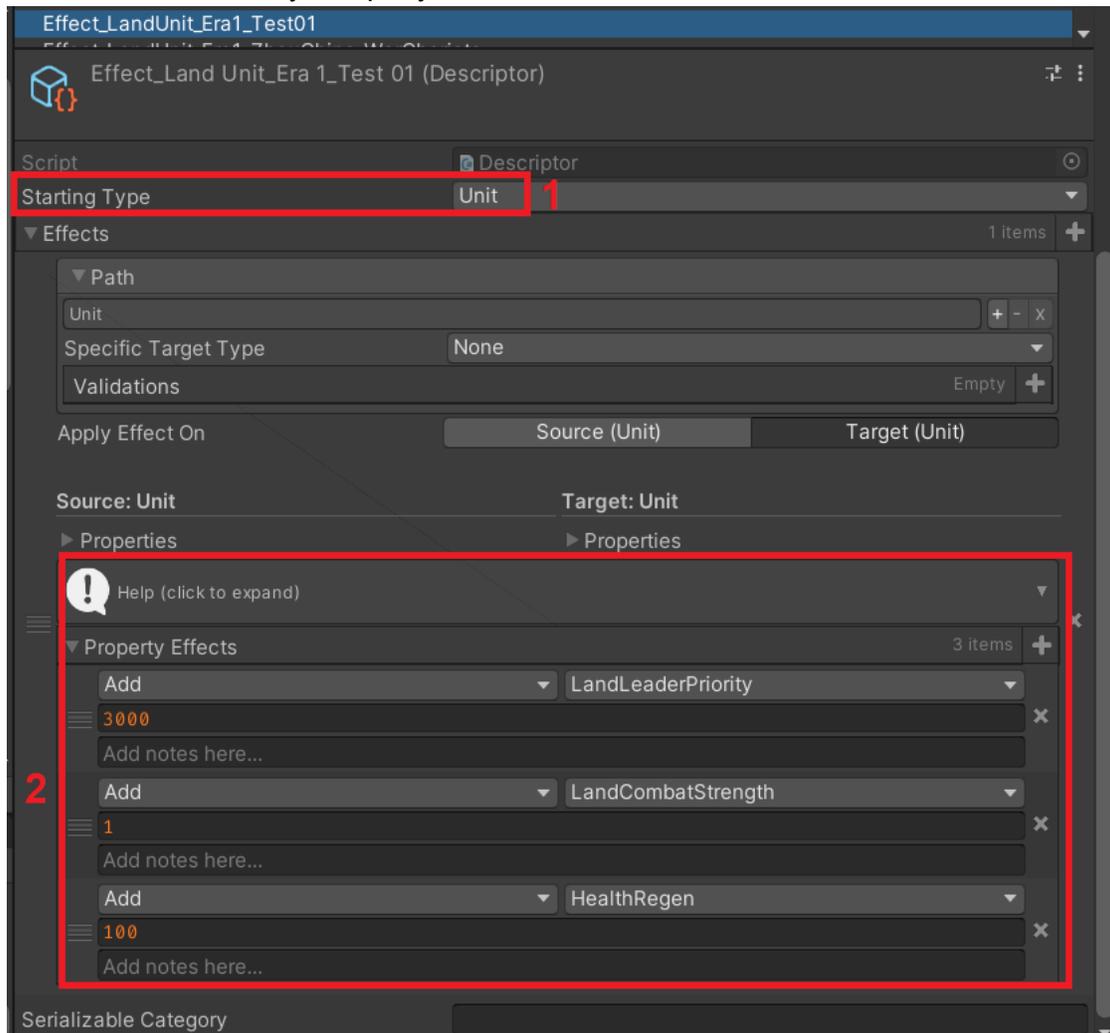
The "Unit Descriptor" is used to assign combat strength, lead priority or change the combat range, etc in the "Unit Speciality" field. To add a new descriptor:

1. Select the corresponding collection (in our case the collection is "LandUnitDescriptor").
2. Add a new element by right-clicking in the inspector space and selecting "Add-> Descriptor" or simply duplicate an existing one.



After the new descriptor appears in the same inspector window, it's strongly recommended to rename the object.

1. Set the "Starting Type" to "Unit".
2. Add as many "Property Effects" as needed.



## 9.4 Setting up the “Construction” tab

If the previous “Definition” tab was all about what the unit does, then the “Construction” allows to set material requirements for its creation and cost.

Constructible	
Can Be Bought Out	<input checked="" type="checkbox"/>
Can Be Canceled	<input type="checkbox"/>
Start When Queued	<input type="checkbox"/>
Can Be Bought Out With Population	<input checked="" type="checkbox"/>
Production Cost	
Type	Production
Constant	300
Rpn Definition Reference	None
Money Instant Cost	
Constant	10
Rpn Definition Reference	None
Influence Instant Cost	
Constant	10
Rpn Definition Reference	None
Population Instant Cost	
Constant	1

- The “Can Be Bought Out” and the “Can Be Bought Out With Population” define whether players can skip the construction and just build a unit instantly using corresponding resources.
- The “Can Be Cancelled” defines if the production can be canceled.
- The “Production Cost” defines the price of the unit. It can be constant (example above) or relative (example below). Constant “-1” means “none” and forces the game to use the “Rpn Definition Reference”.

Production Cost	
Type	Production
Constant	-1
Rpn Definition Reference	ProductionCost_Unit_Era3_Medium

- The “Rpn Definition Reference” defines the relative cost of the unit. Splitted into high, medium, and low costs per each era.
- The “Money/Population Instant Cost” defines how much resources will be consumed before the production start.

## 9.5 Setting up the “Prerequisites” Tab

On the “Prerequisites” tab the prerequisite requirements of the unit can be selected.

The screenshot shows the 'Prerequisites' tab of a game editor. The interface is divided into four main sections: Definition, Construction, Prerequisites, and AI. The 'Prerequisites' section is expanded, showing various prerequisite categories. The 'Era Prerequisite' is set to 'Greater Or Equal' and 'Era1'. The 'Settlement Status Prerequisite' is set to 'Available Buyout Influence Only' for 'Camp Constraint' and 'Available' for 'City Constraint'. The 'Settlement Property Prerequisites' section is empty. The 'Settlement Stability Prerequisite' is set to 'Any'. The 'District Count Prerequisites' section shows a warning icon and a message: 'Reference "Extension\_ArtificialWonder\_Era1\_Stonehenge" is missing or invalid!'. Below this, the 'District Definition' is set to 'Extension\_ArtificialWonder\_Era1\_Stonehenge' and the 'Desired Count' is set to '1'. The 'Action Type Prerequisite' is set to 'None'.

1. Set the “Era Prerequisites” as of when the unit appears.

This close-up screenshot shows the 'Era Prerequisite' field. The 'Operator' is set to 'Greater Or Equal' and the 'Era' is set to 'Era1'.

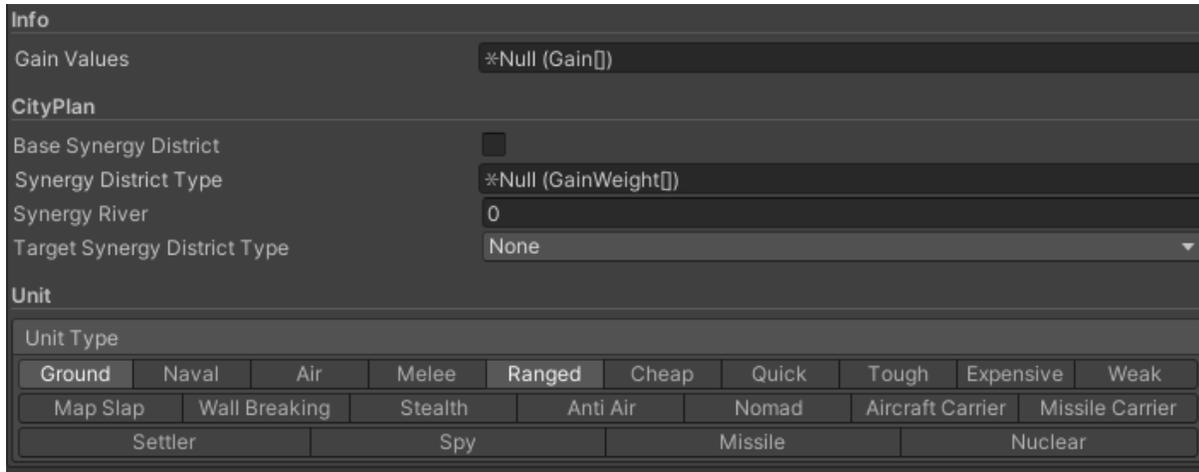
2. Set the “Settlement Status Prerequisites” to validate where the unit can be built (camp or city).

This close-up screenshot shows the 'Settlement Status Prerequisite' section. The 'Camp Constraint' is set to 'Available Buyout Influence Only' and the 'City Constraint' is set to 'Available'.

It is also possible to define whether special resources are needed, religion must be presented or any district presented. Please go through the fields to find out more.

## 9.6 Setting up the “AI” Tab

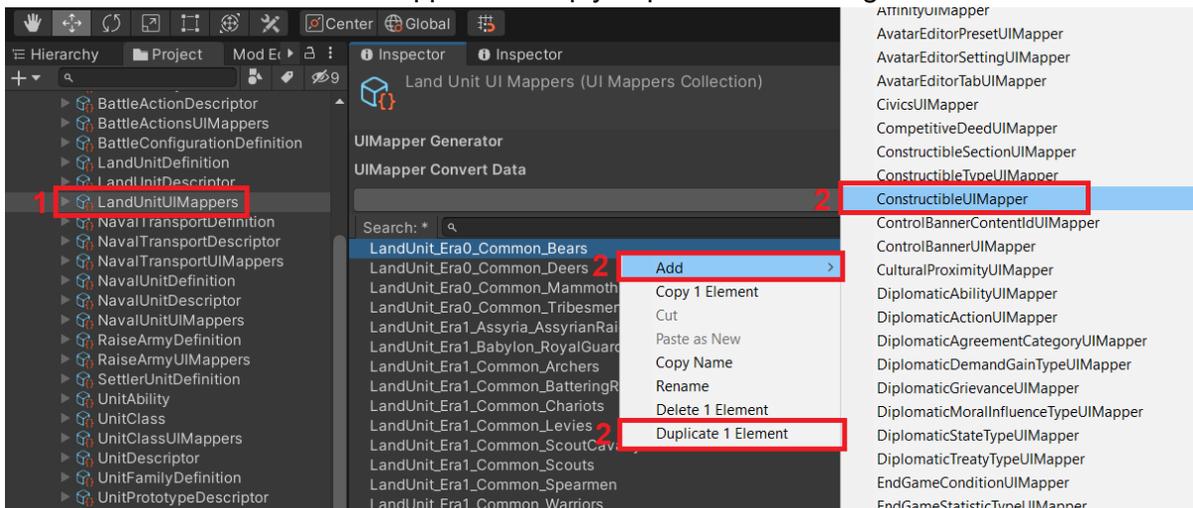
The “AI” tab defines how bots treat the unit. Select the related “Unit Type” to help AI treat the newly created unit correctly.



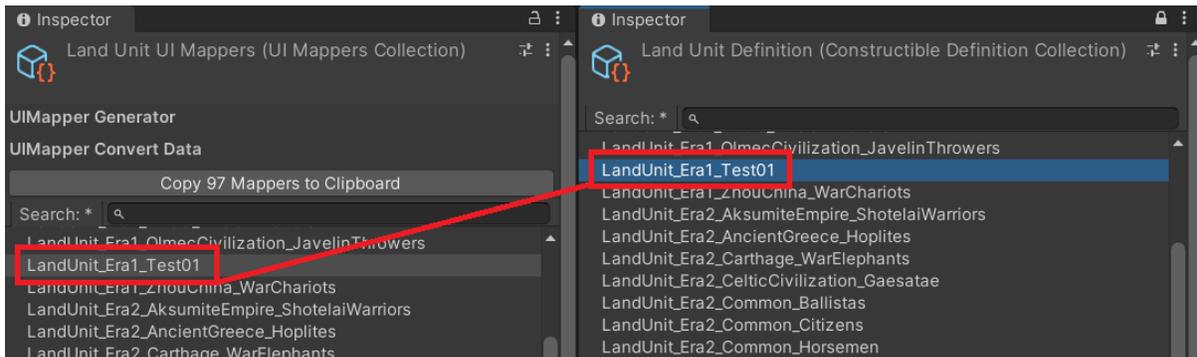
## 9.7 Adding the UI mapper for the unit

In order to rename and display the unit icon correctly, the UI mapper object has to be created.

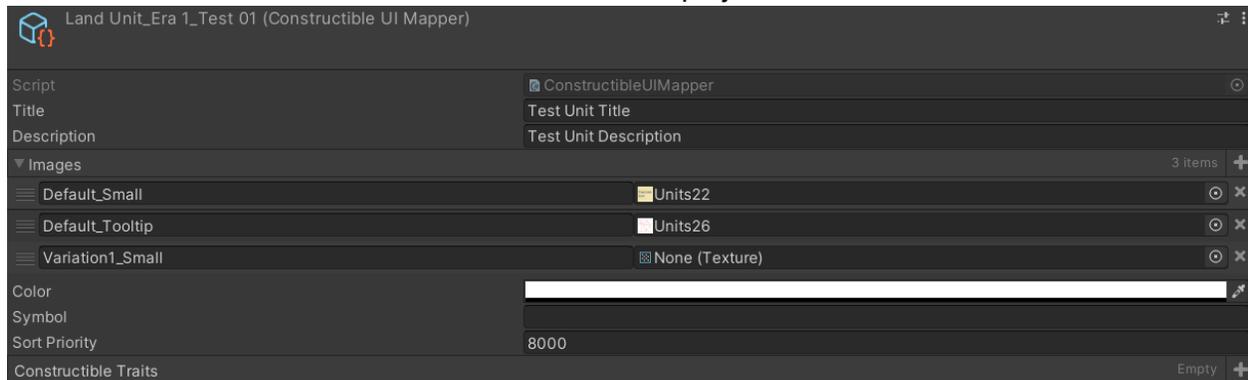
1. Select the corresponding UI collection (in our case the collection is “LandUnitUIMappers”).
2. Add a new element by right-clicking in the inspector space and selecting “Add-> ConstructibleUIMapper” or simply duplicate an existing one.



- Rename the newly created object to the same one the definition object has. Names from 2 collections MUST match.



- Fill the corresponding fields with pictures, name, description. The image must be stored in the “Resource” folder of the project.



## 9.8 Creating the unit presentation

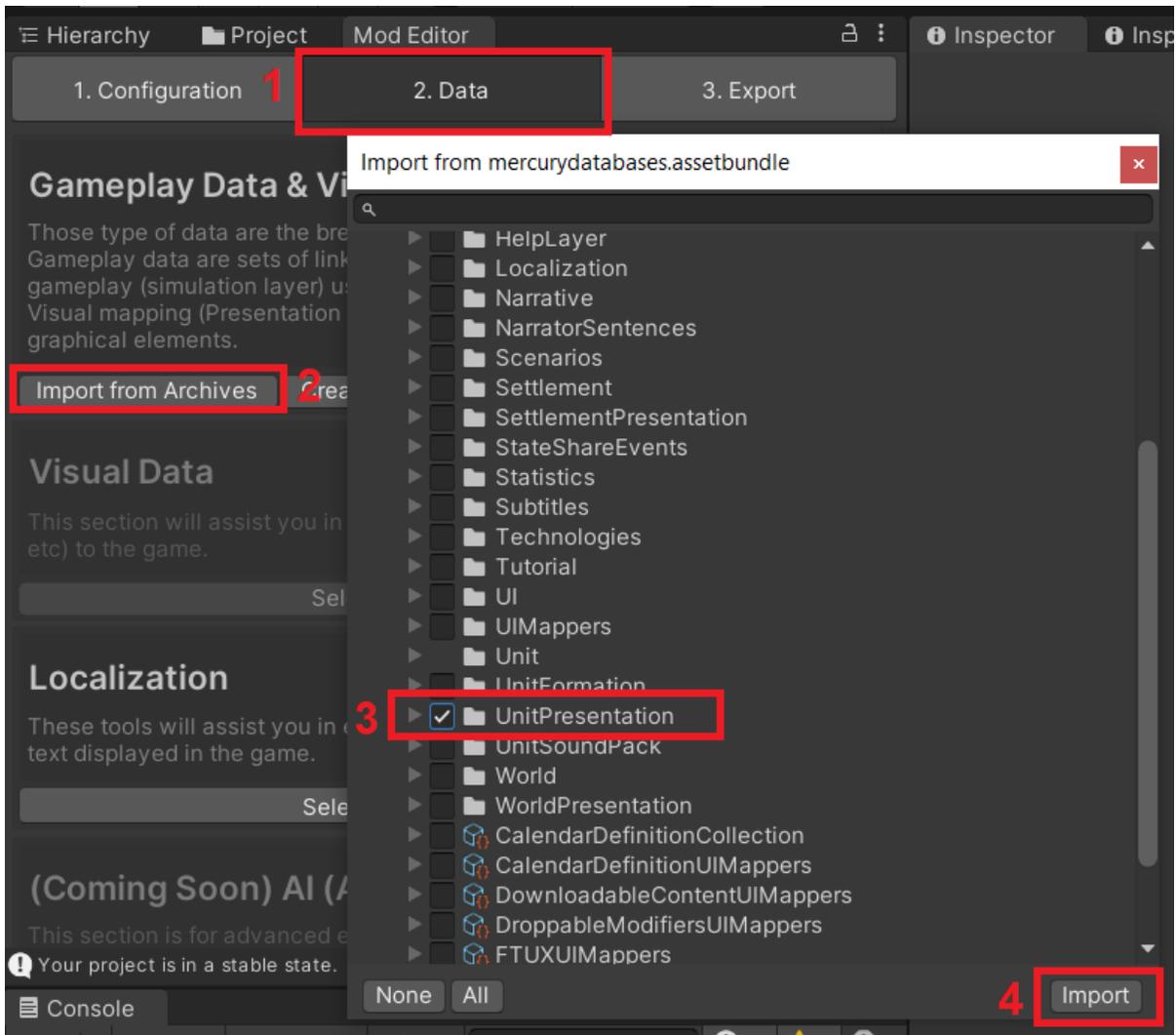
To display a 3D model of the unit, the new presentation pawn object must be created.

**Note! It is highly recommended not to mix melee unit models/assets with ranged unit abilities or vice versa.**

### 9.8.1 Setting up the environment

To modify or create a new unit presentation, the “UnitPresentation” database has to be exported:

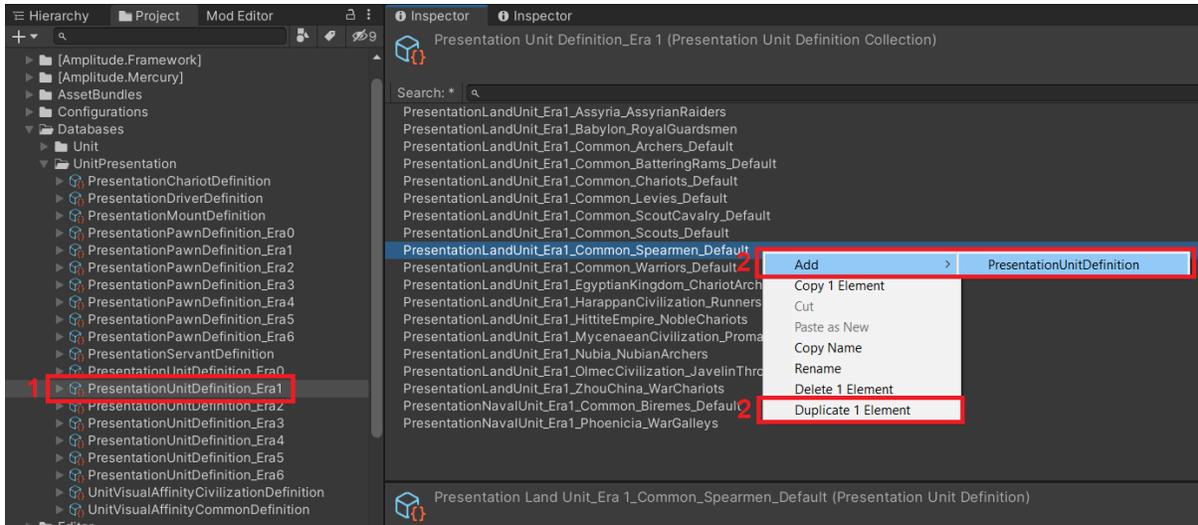
- Go to “2. Data” of the Mod Editor.
- Select “Import from Archives”.
- Select “UnitPresentation”.
- Press “Import”.



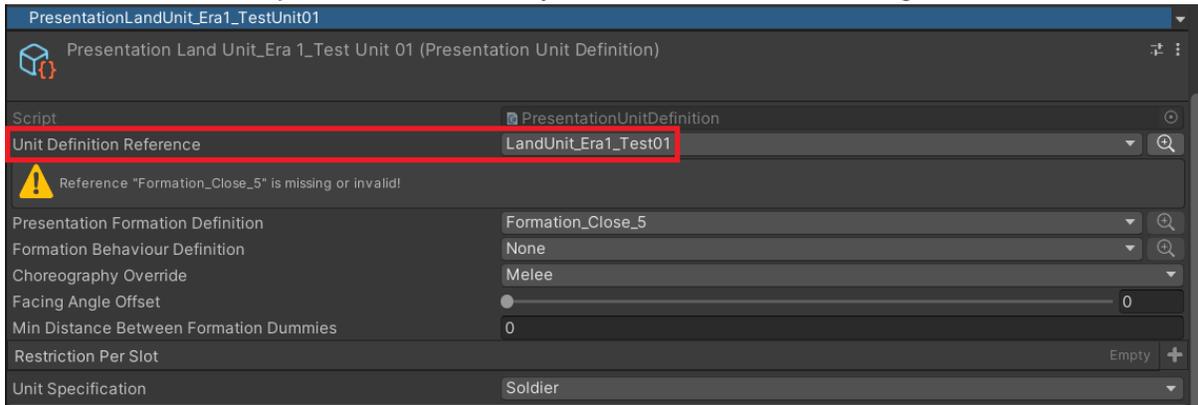
## 9.8.2 Creating the new unit presentation

The unit presentation defines how the group of units is presented on the map tile.

1. Select the corresponding presentation era collection (in our case the collection is "PresentationUnitDefinition\_Era1").
2. Add a new element by right-clicking in the inspector space and selecting "Add-> PresentationUnitDefinition" or simply duplicate an existing one.



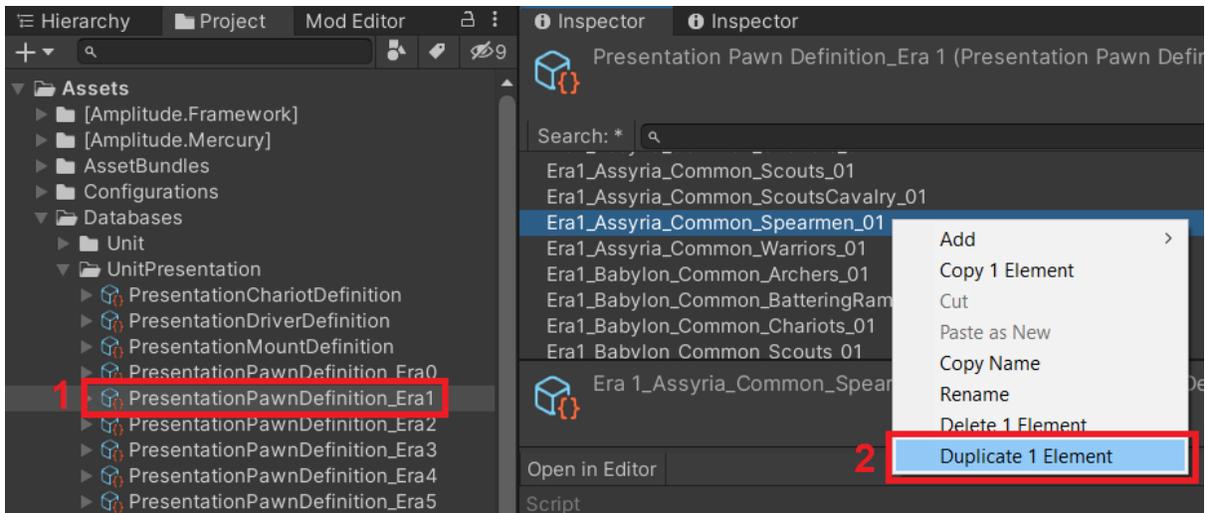
3. Set the correct “Unit Definition Reference” to the newly created unit. It is possible to modify other fields but they won’t be described in this guide.



## 9.9 Creating the new unit pawn presentation

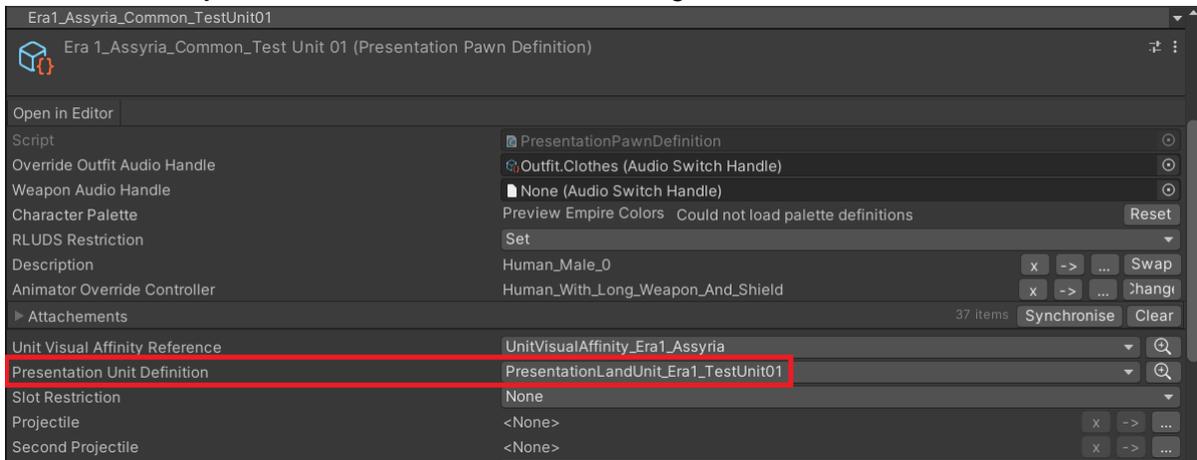
The unit pawn presentation defines how the single unit looks like.

1. Select the corresponding pawn presentation era collection (in our case the collection is “PresentationPawnDefinition\_Era1”).
2. Add a new element by right-clicking in the inspector space and duplicating an existing one. **Note!** It is highly recommended to work only with the existing objects and duplicate them.



After the new object appears in the same inspector window, it's strongly recommended to rename it.

3. Set the "Presentation Unit Definition" field to the previously created presentation object. Leave other fields with no changes.



## 9.10 Testing

After all steps were done - click "Build and Run" to check results. Start a new game and reach the era in which the newly created unit appears (depends on prerequisites).

The image shows a game interface with two main panels. The left panel displays the details for a "TEST UNIT".

**TEST UNIT TITLE**

Test Unit Description

**UNIT STATS**

- 1 (Shield icon)
- 4 (Arrow icon)
- 1 (Shield icon)
- 0 (Star icon)

**UPKEEP**

- 1 (Gold icon)

**UNIT CLASS**

**Anti-Cavalry** : Close-combat Unit. Receives bonus Combat Strength when fighting mounted enemies.

**El Dorado**: Generates additional Money from winning battles and ransacking.

**COSTS**

- 300 (Gear icon)
- 10 (Gold icon)
- 10 (Star icon)
- 1 (Person icon)

**CONSTRUCTION TIME**

Turns before completion: 22

**DISTRICT PREREQUISITES**

0/1 Stonehenge (missing 1)

Cannot be done without 10 Money.  
Cannot be done without 1 Population.  
Cannot be done without building the required District types in your City.

The right panel shows the city interface for "AŠŠUR".

0 +3 per  
0 0  
1/2

**AŠŠUR**

0/8 15 2 1

**CONSTRUCTION QUEUE (0)**

No construction in progress.

No pending construction.

**CONSTRUCTIONS** COMPLETED

ITEM LIST VIEW ONLY CONSTRUCTIBLE

QUEUE BUYOUT (6)

**UNITS (2)**

- Test Unit Icon (Cost: -1, 10, 10, -1)

**PUBLIC CEREMONIES (1)**

36

Test unit appeared in the city and it has exactly the same prerequisites and cost as we put (production is decreased due to game speed).

# 10 Adding a new Culture

Culture defines which bonuses the player receives through eras. Players are able to combine multiple cultures by adopting a new culture or transcending a previous culture each Era to form their civilization.



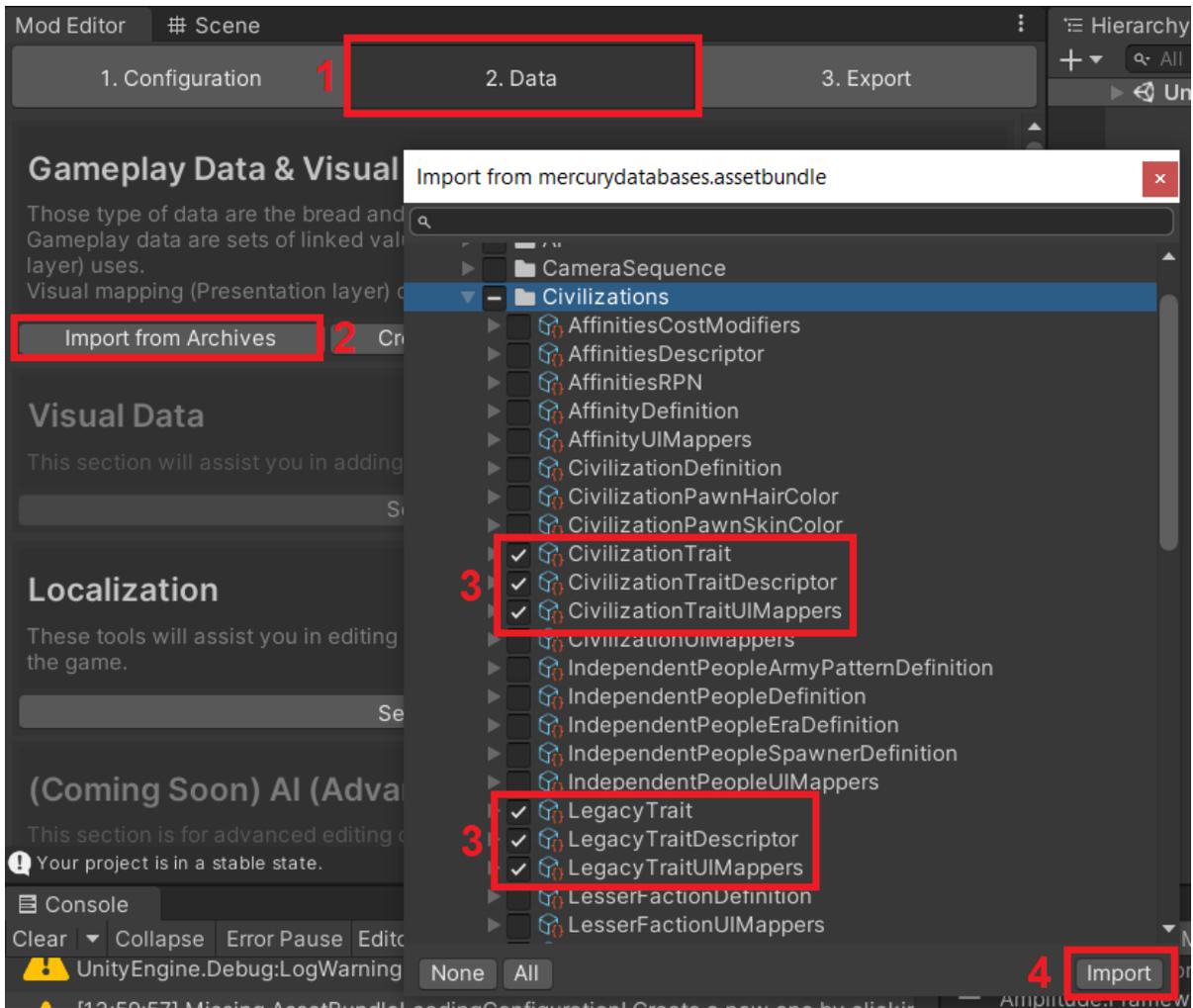
The culture choice affects the following parameters:

1. The appearance of units and buildings.
2. Emblematic Units and Quarters unique to each culture and only available within a specific era.
3. Culture Affinities gives the gameplay orientation with special effects and bonuses.
4. Legacy Trait - unique powerful effects and bonus for that culture till the end of the game.

## 10.1 Setting up the environment

To modify or create a new culture, the “Civilization” database collections have to be exported:

1. Go to “2. Data” of the Mod Editor.
2. Select “Import from Archives”.
3. Select the related collections under the “Civilizations” folder.
4. Press “Import”.

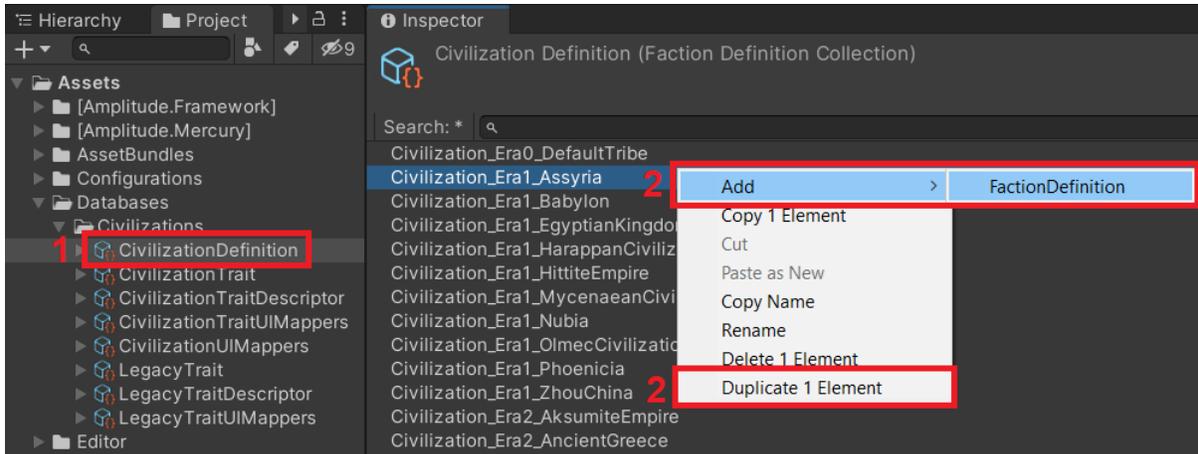


**Important!** It is not recommended to store unused objects within the mod project structure. Please either not export unused objects, or remove unnecessary objects from collections/collections after the mod is created.

## 10.2 Creating the new culture

In order to create a new culture, a new “FactionDefinition” object has to be created.

1. Select the “CivilizationDefinition” collection in the project structure.
2. Add a new element by right-clicking in the inspector space and selecting “Add-> FactionDefinition” or simply duplicate an existing one.

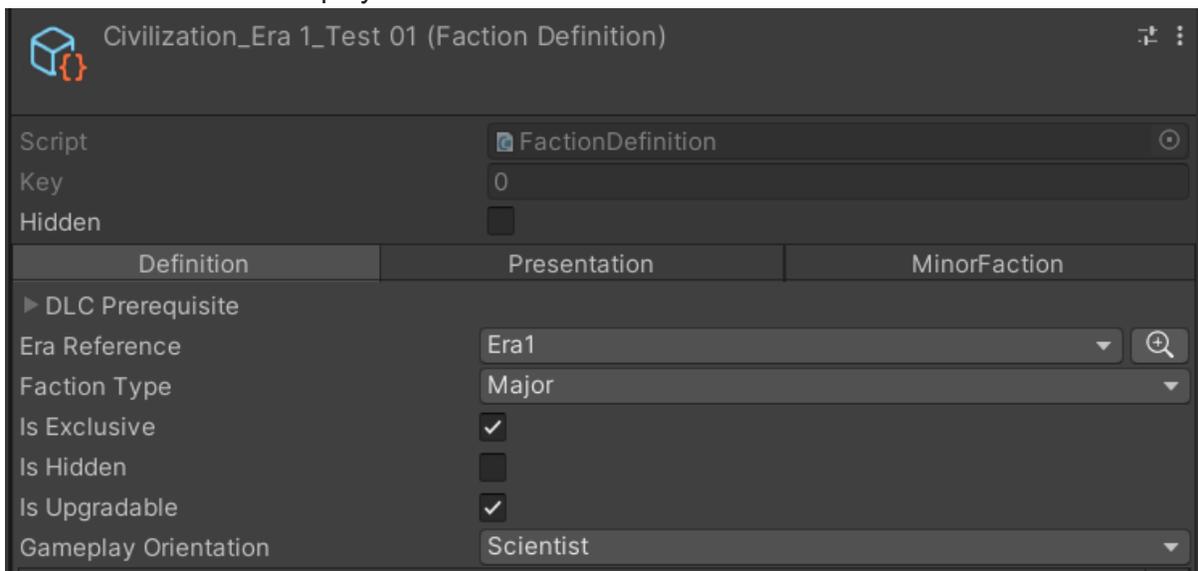


After the new definition appears in the same inspector window, it's strongly recommended to rename the object.

### 10.3 Setting up the “Definition” tab

On the “Definition” tab the properties of the culture can be selected. Fulfill the fields according to needs. Some explanations on the fields are below:

- The “Era Reference” defines in which era the culture is available.
- The “Faction Type” must be “Major” to be playable.
- The “Is Exclusive” defines if only 1 player can select the culture per game.
- The “Is Hidden” technical field to hide the culture in the game.
- The “Is Upgradable” defines if other cultures can be selected after this is picked.
- The “Gameplay Orientation” defines an orientation bonus for the culture.

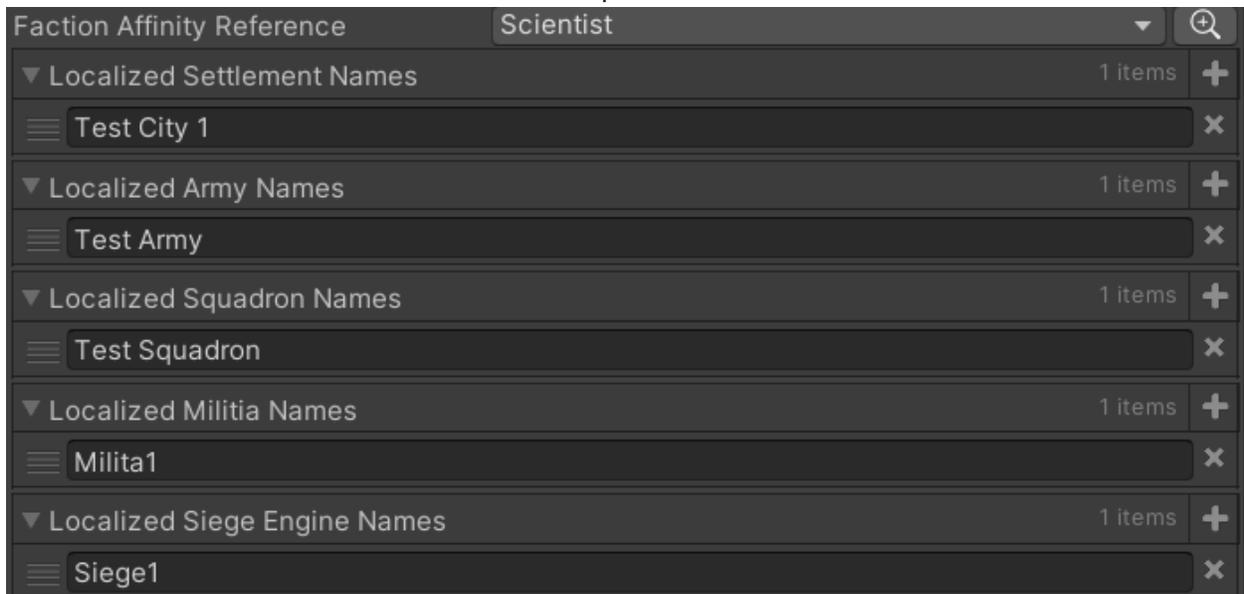


- The “Legacy Trait References” gives the legacy trait for the culture. Objects are stored in the “LegacyTrait” collection.

- The “Trait References” gives additional legacy traits for the culture. Objects are stored in the “CivilizationTrait” collection.



- The “Faction Affinity Reference” should be the same as the “Faction Type” and is used for visual culture reference.
- Localized fields are used for special namings on different objects inside the culture.  
**Note!** At least 1 name must be presented in each localized field.



## 10.4 Setting up the “Presentation” tab

On the “Presentation” tab the visual properties of the culture can be defined. Fulfill the fields according to needs and refer to the existing cultures for the example.

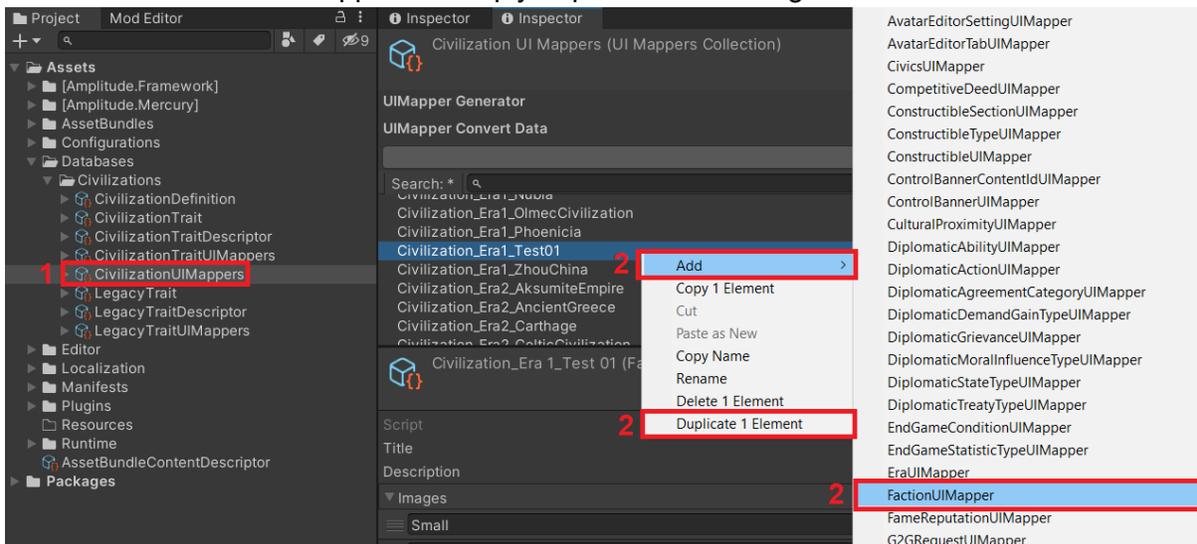
Definition	Presentation	MinorFaction
▼ Building Visual Affinities <span style="float: right;">7 items +</span>		
Era Index	0	
Building Visual Affinity Reference	BuildingVisualAffinity_NoVisualAffinity	+ 🔍 ×
Era Index	1	
Building Visual Affinity Reference	BuildingVisualAffinity_Era1_ZhouChina	+ 🔍 ×
Era Index	2	
Building Visual Affinity Reference	BuildingVisualAffinity_Era1_ZhouChina	+ 🔍 ×
Era Index	3	
Building Visual Affinity Reference	BuildingVisualAffinity_Common_Asian	+ 🔍 ×
Era Index	4	
Building Visual Affinity Reference	BuildingVisualAffinity_Common_Asian	+ 🔍 ×
Era Index	5	
Building Visual Affinity Reference	BuildingVisualAffinity_Common_Asian	+ 🔍 ×
Era Index	6	
Building Visual Affinity Reference	BuildingVisualAffinity_Common_Eastern	+ 🔍 ×
▶ Unit Visual Affinities <span style="float: right;">7 items +</span>		
Faction Constructible Visual Variation	Variation5	▼
⚠ Reference "AvatarCostumeAffinity_HarappanCivilization" is missing or invalid!		
Avatar Costume Visual Affinity	AvatarCostumeAffinity_HarappanCivilization	▼ 🔍
▼ Skin Color References <span style="float: right;">1 items +</span>		
AvatarColor_Skin_11		▼ 🔍 ×
▼ Hair Color References <span style="float: right;">1 items +</span>		
AvatarColor_Haircut_Blond02		▼ 🔍 ×
▼ Landmark Name Affinity References <span style="float: right;">1 items +</span>		
LandmarkNameAffinity_Placeholders		▼ 🔍 ×

- The “Building Visual Affinities” is used for visual reference of the culture buildings through eras.
- The “Unit Visual Affinities” is used for visual reference of the culture units through eras.
- The “Faction Constructible Visual Variation” is used for small visual details.
- The “Avatar Costume Visual Affinity” is used for avatar costume reference.

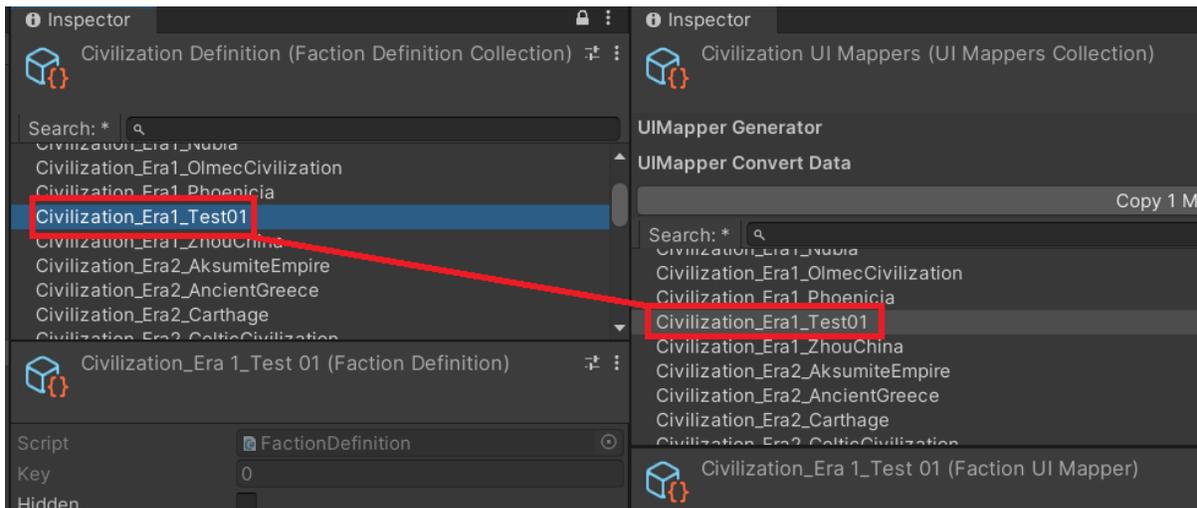
## 10.5 Adding a UI mapper for civilization

In order to rename and display the culture correctly, the UI mapper object has to be created:

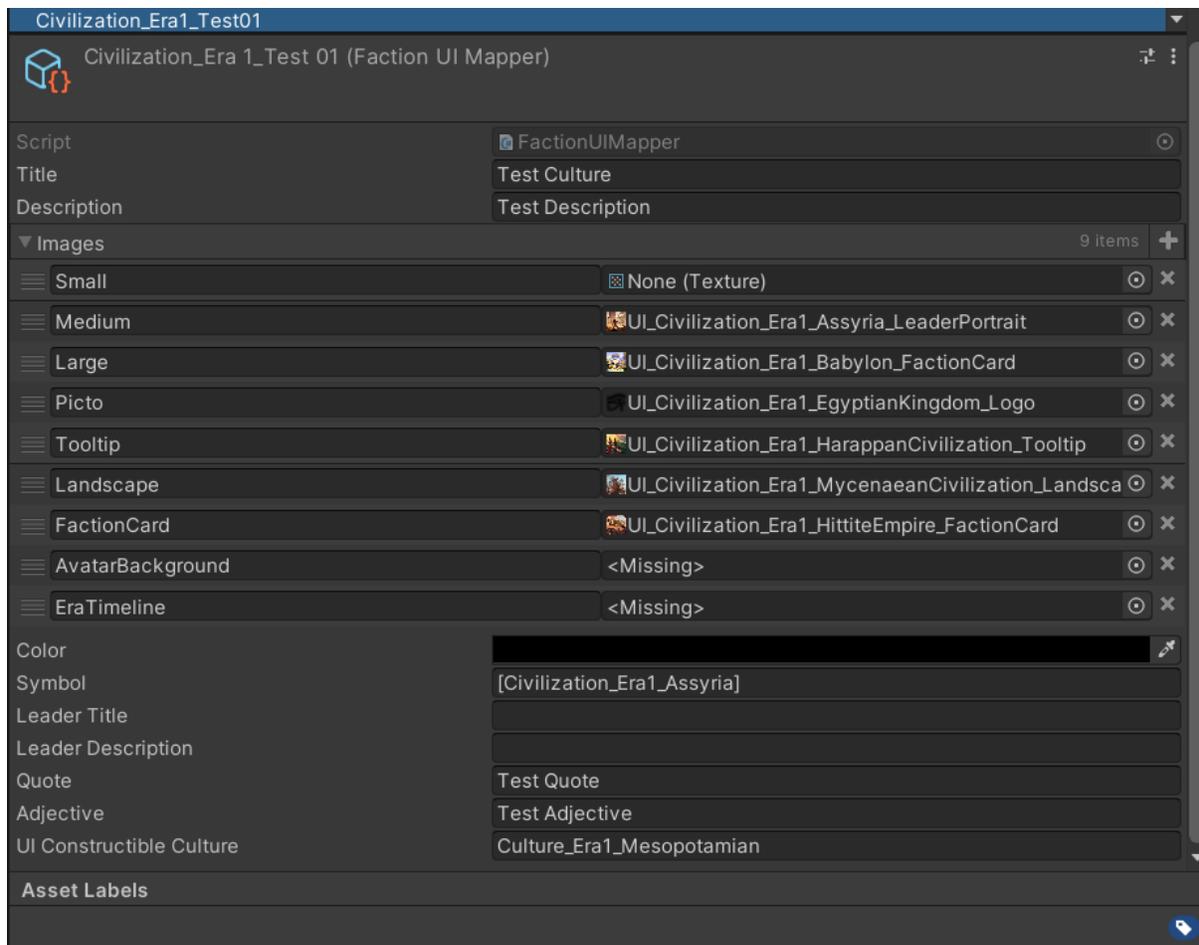
1. Select the “CivilizationUIMappers” collection.
2. Add a new element by right-clicking in the inspector space and selecting “Add-> FactionUIMapper” or simply duplicate an existing one.



3. Rename the newly created object to the same one the definition object has. Names from 2 collections MUST match.



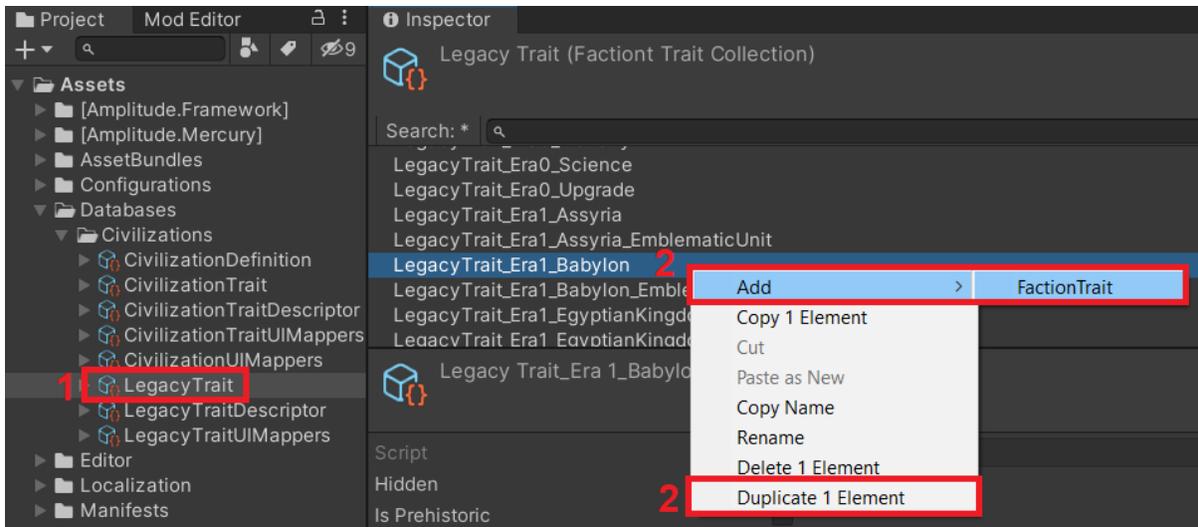
4. Fill the corresponding fields with pictures, name, description. The images must be stored in the “Resource” folder of the project.



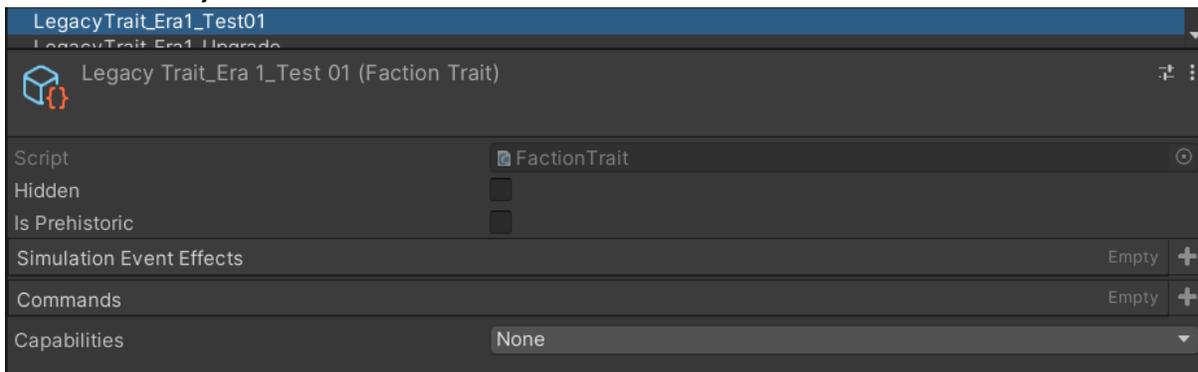
## 10.6 Creating the new legacy trait

Legacy trait is used to give bonuses or unlock emblematic units within a specific culture. In order to create a new legacy trait, a new "FactionTrait" object has to be created.

1. Select the "LegacyTrait" collection in the project structure.
2. Add a new element by right-clicking in the inspector space and selecting "Add-> FactionTrait" or simply duplicate an existing one.



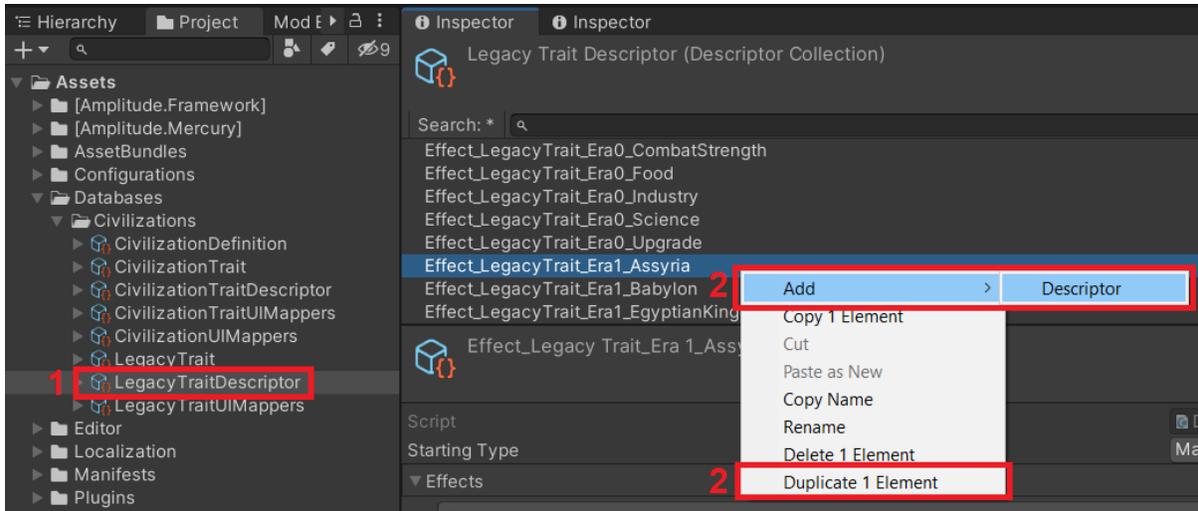
After the new definition appears in the same inspector window, it's strongly recommended to rename the object.



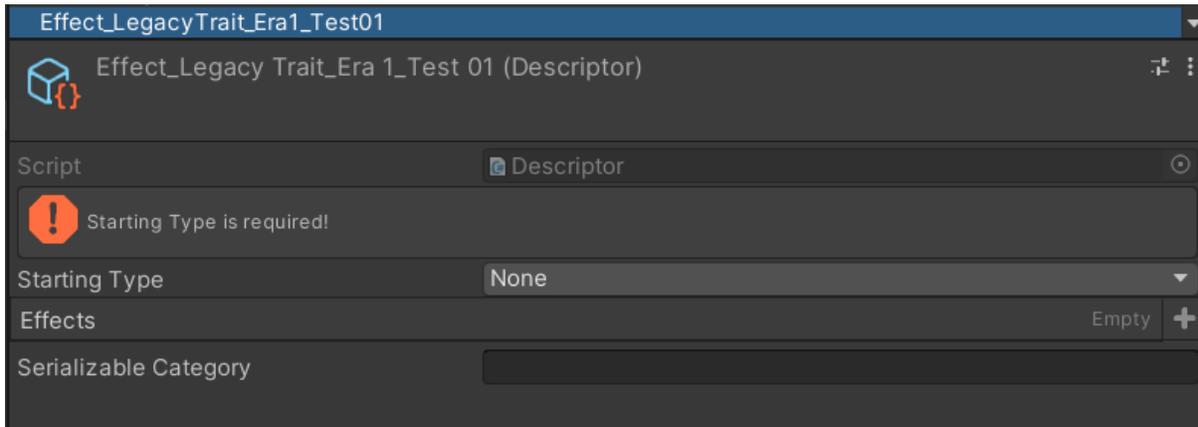
## 10.7 Creating the new legacy trait descriptor

In order to give a bonus effect to a district, city or unit within a legacy trait a new “Descriptor” object has to be created.

1. Select the “LegacyTraitDescriptor” collection in the project structure.
2. Add a new element by right-clicking in the inspector space and selecting “Add-> Descriptor” or simply duplicate an existing one.



After the new definition appears in the same inspector window, it's strongly recommended to rename the object.

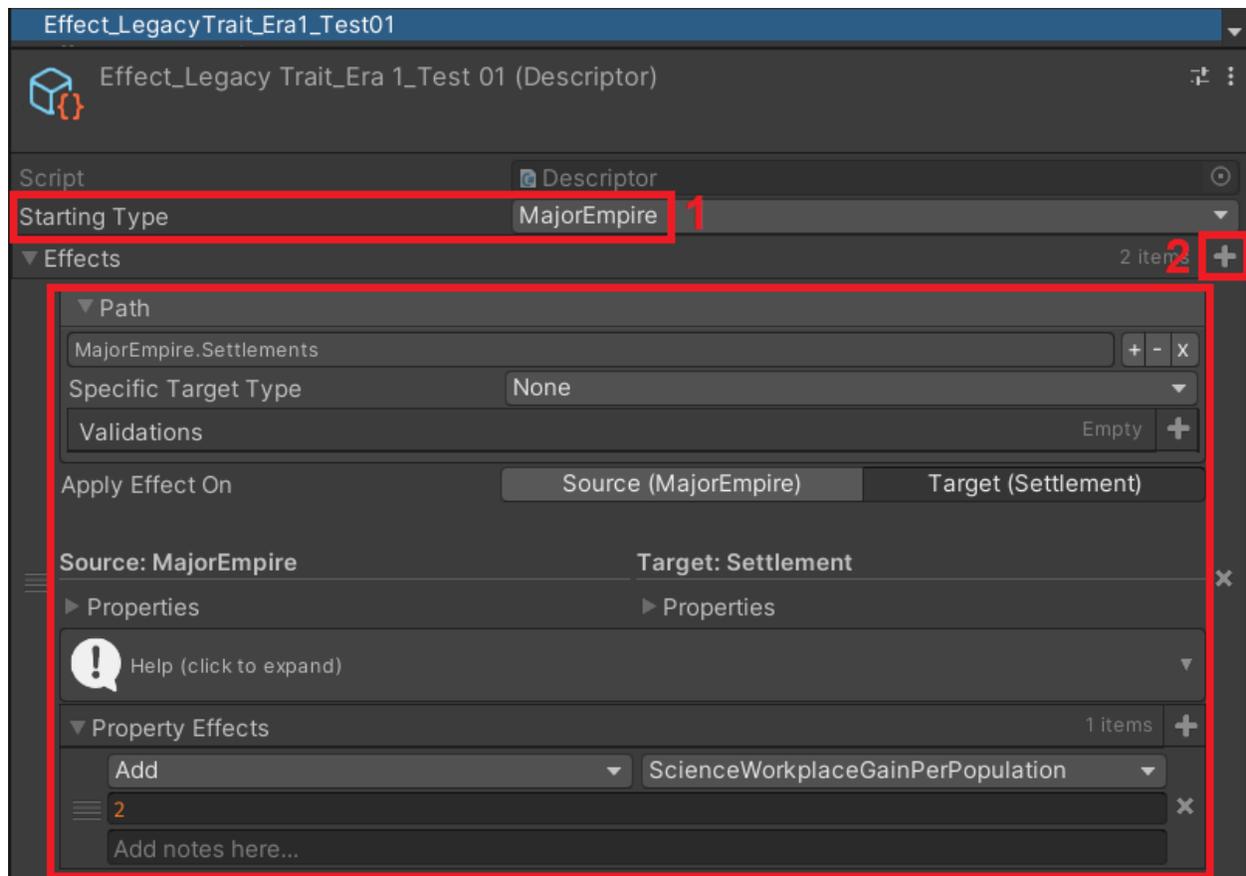


## 10.8 Setting up the legacy trait descriptor

To give some effect to the descriptor.

1. Set the "Starting Type" to "MajorEmpire".
2. Add necessary effects.

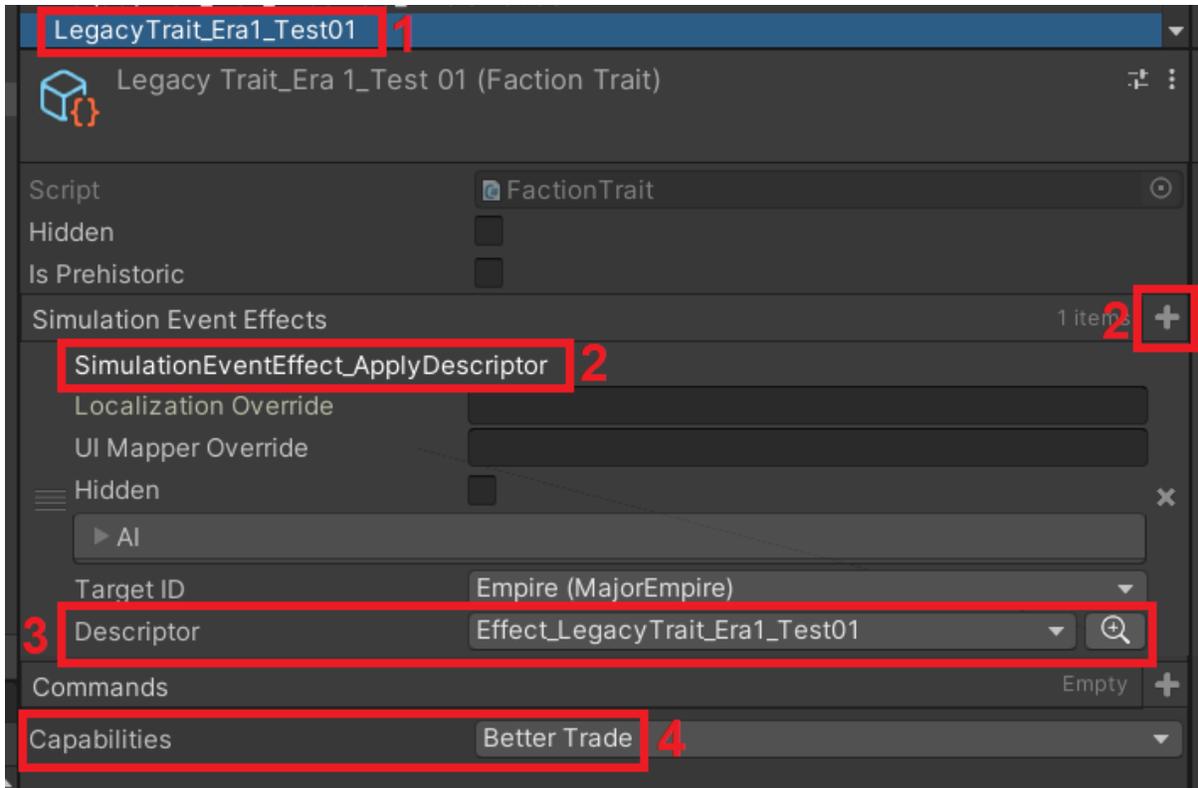
**Note!** The targeted object of the descriptor depends on the "Path" under the "Effect" section. Pick the preferable one or refer to existing traits for examples.



## 10.9 Setting up the legacy trait

To give the legacy trait some effects, the corresponding descriptor has to be assigned and other fields to be completed.

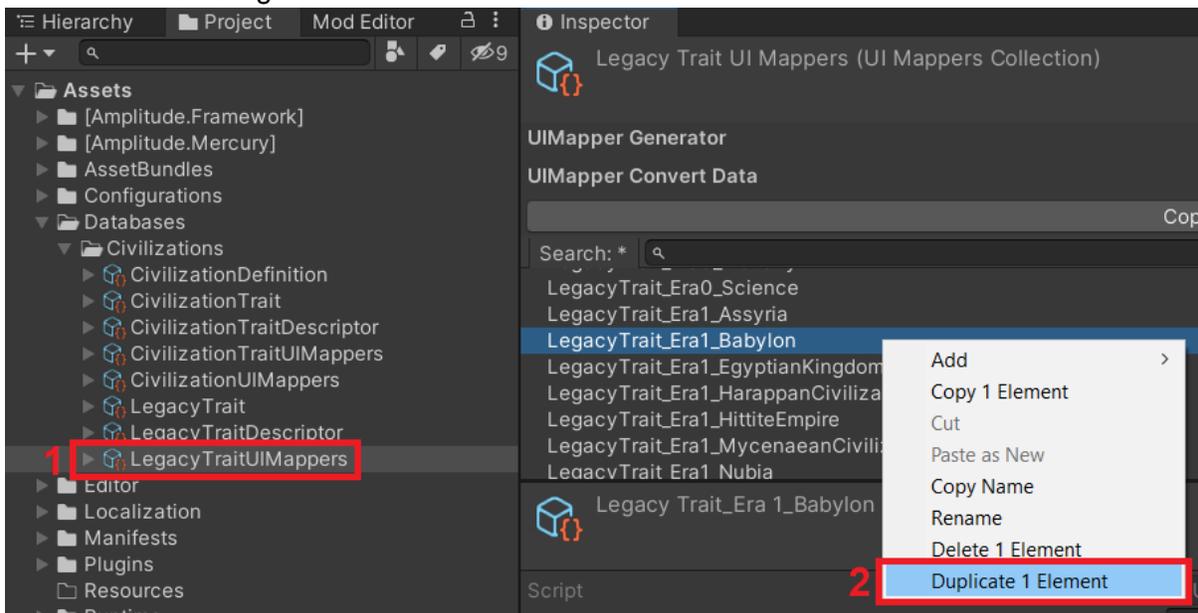
1. Select the previously created legacy trait object.
2. Add the "SimulationEventEffects\_ApplyDescriptor".
3. Set the descriptor reference.
4. Set the "Capabilities" field.



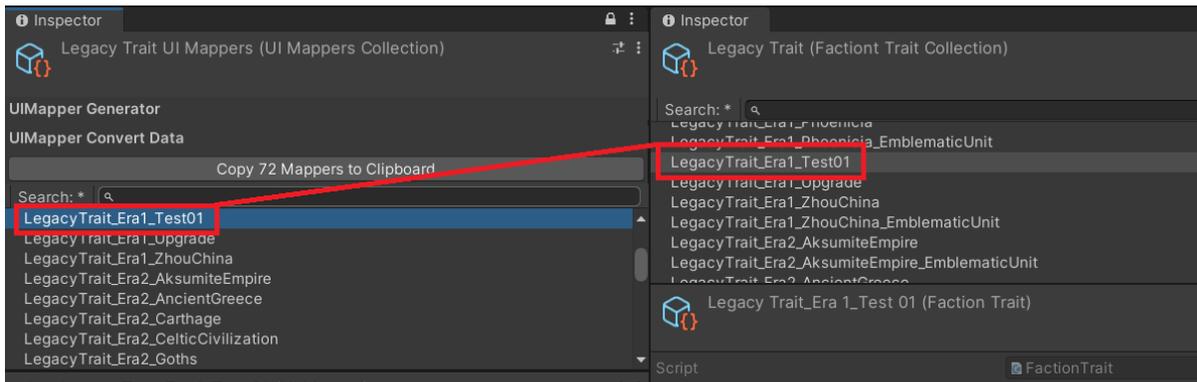
## 10.10 Adding a UI mapper for the Legacy Trait

In order to rename and display the legacy trait correctly, the UI mapper object has to be created.

1. Select the “LegacyTraitUIMappet” collection.
2. Add a new element by right-clicking in the inspector space and duplicating an existing one.



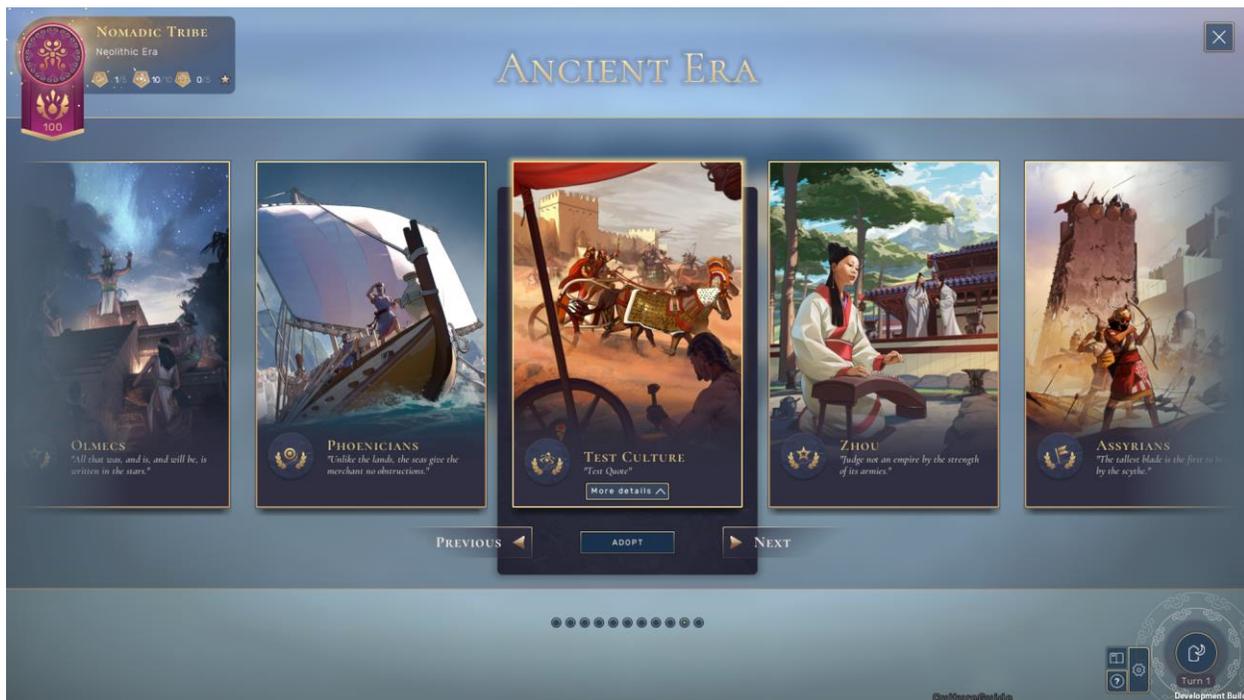
- Rename the newly created object to the same one the definition object has. Names from 2 collections MUST match.



- Fill the corresponding fields with pictures, name, description. The image must be stored in the "Resource" folder of the project.

## 10.11 Testing

After all steps were done - click "Build and Run" to check intermediate results. Start a new game and reach the era in which a newly created civilization appears (depends on prerequisites).



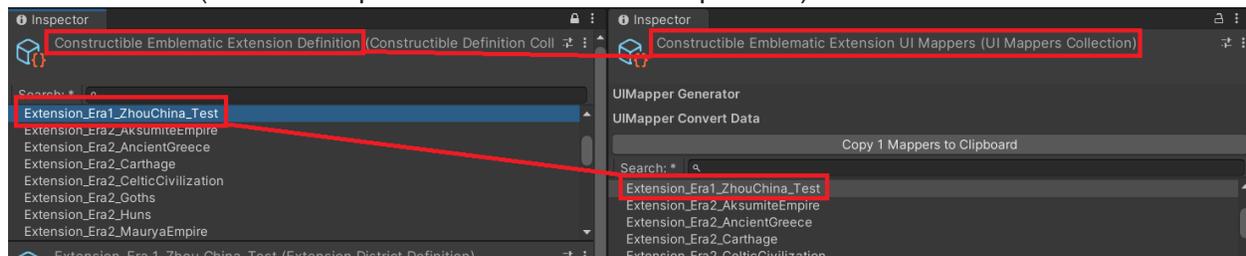


As it is shown, the new culture was added to the game. However, the fields “Emblematic Extension” and “Emblematic Unit” look wrong, because unit and district were not assigned to the newly created culture.

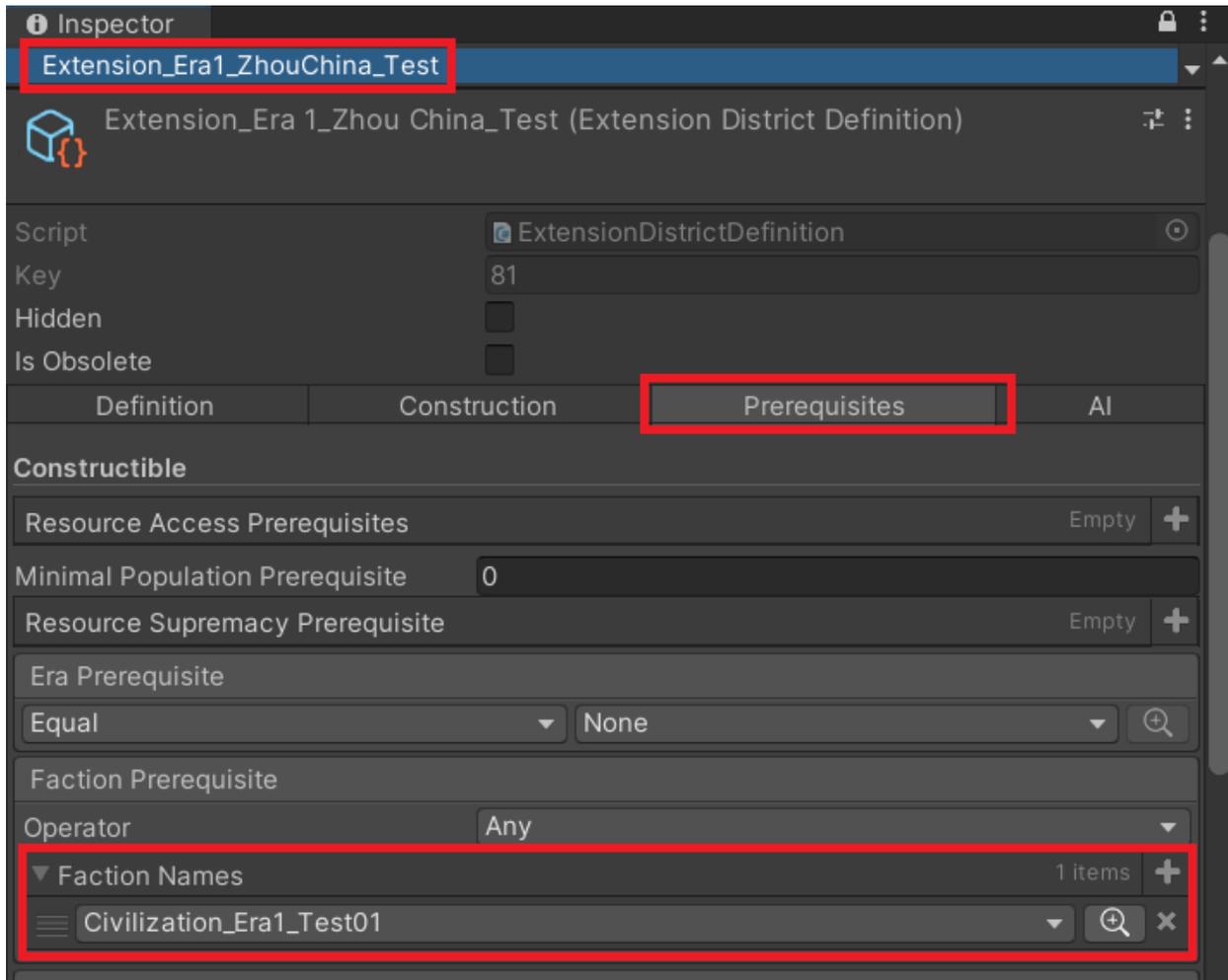
## 10.12 Adding an emblematic district

Adding the emblematic district to the culture works the same way as adding the constructible district from the previous chapter, but the important step is to set the “Faction Names” in the district prerequisites.

1. Create the necessary constructible district with definition, descriptor and UI mapper (in the example the Zhou ED will be duplicated).



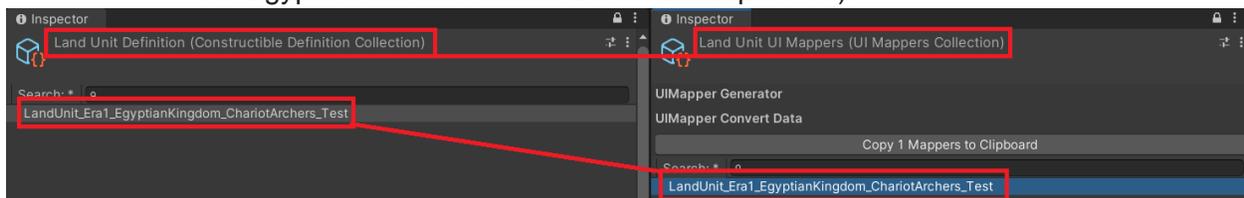
2. Select the constructible definition object, go to the “Prerequisites” tab and add the new culture under “Faction Names”.



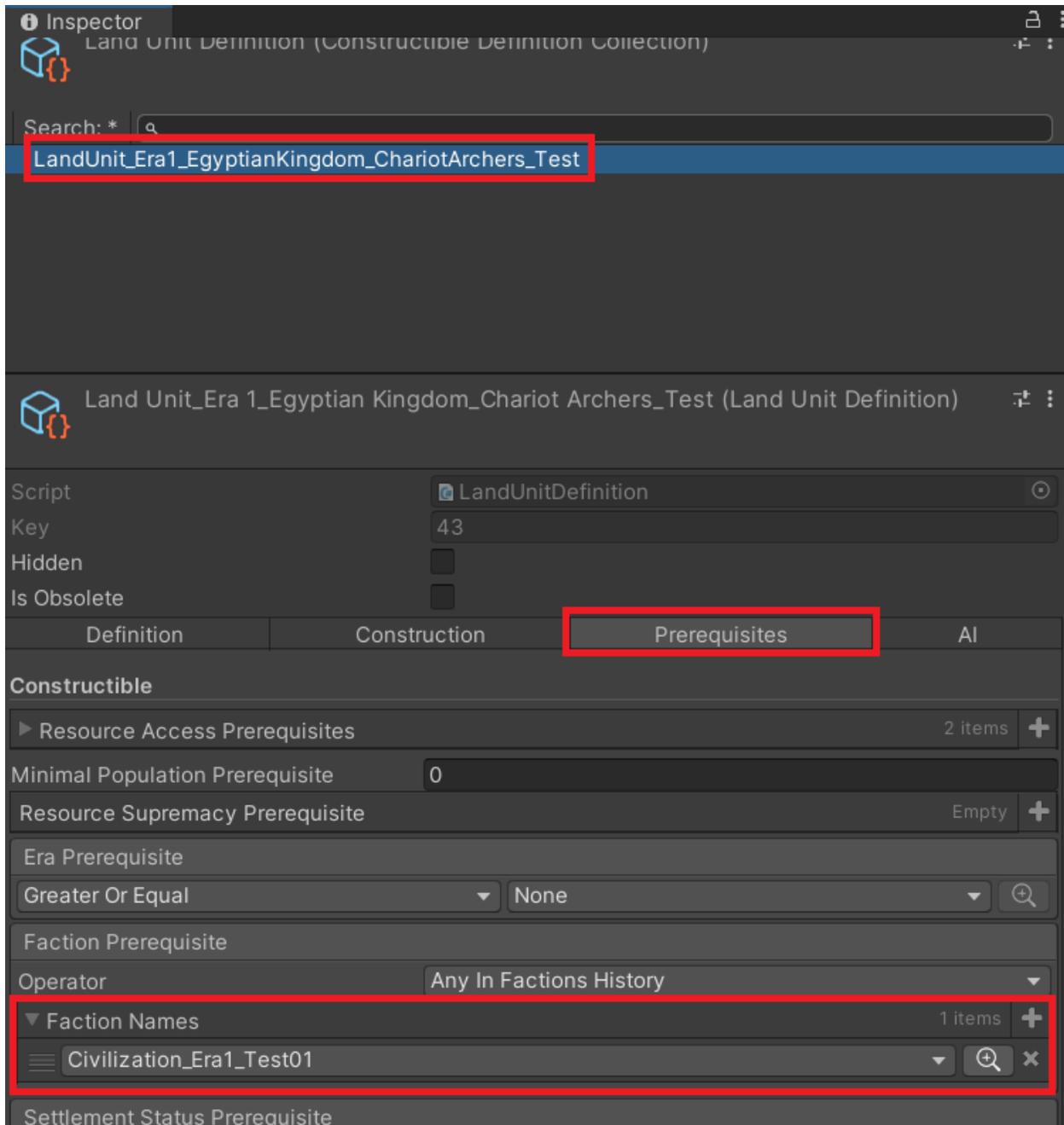
## 10.13 Adding an emblematic unit

Adding the emblematic unit to the culture works the same way as adding the unit from the previous chapter, but the important step is to set the “Faction Names” in the district prerequisites.

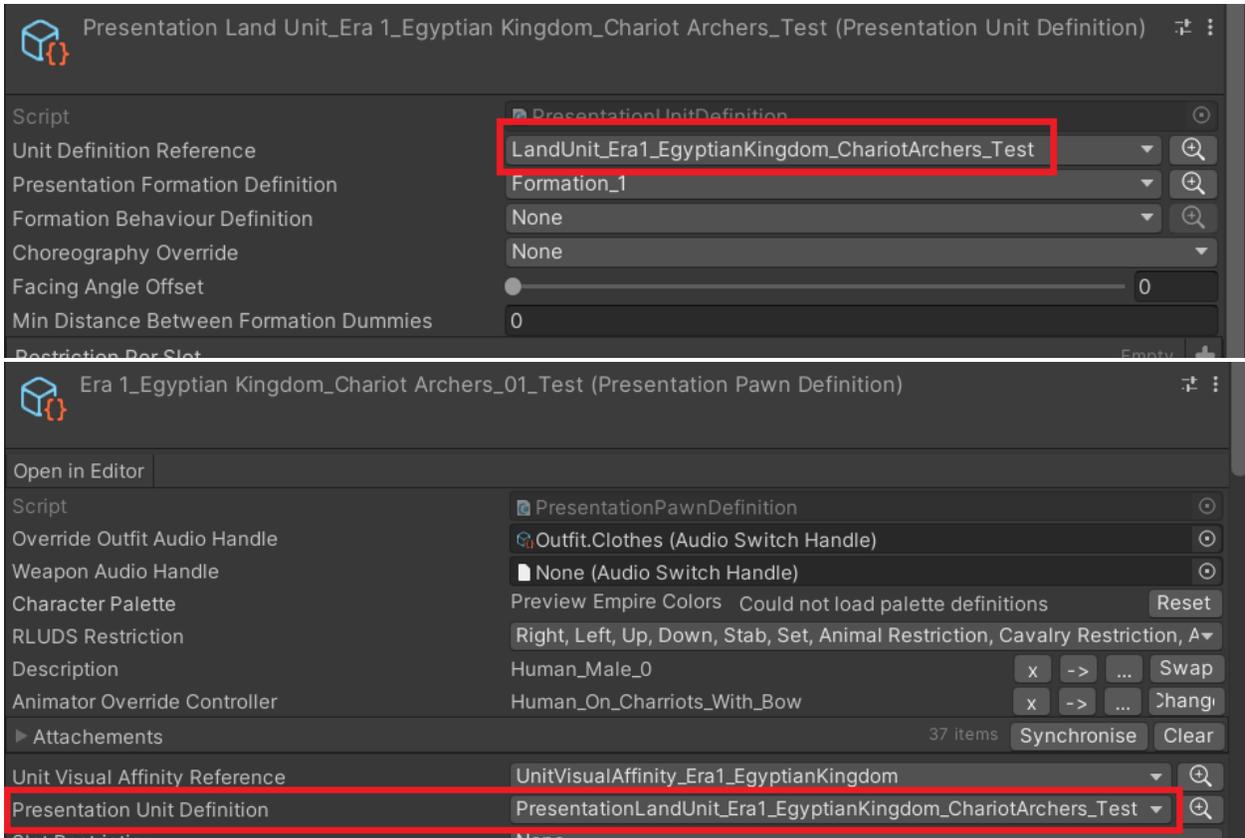
1. Create the necessary unit with definition, descriptor and UI mapper (in the example the Egyptian chariot archers EU will be duplicated).



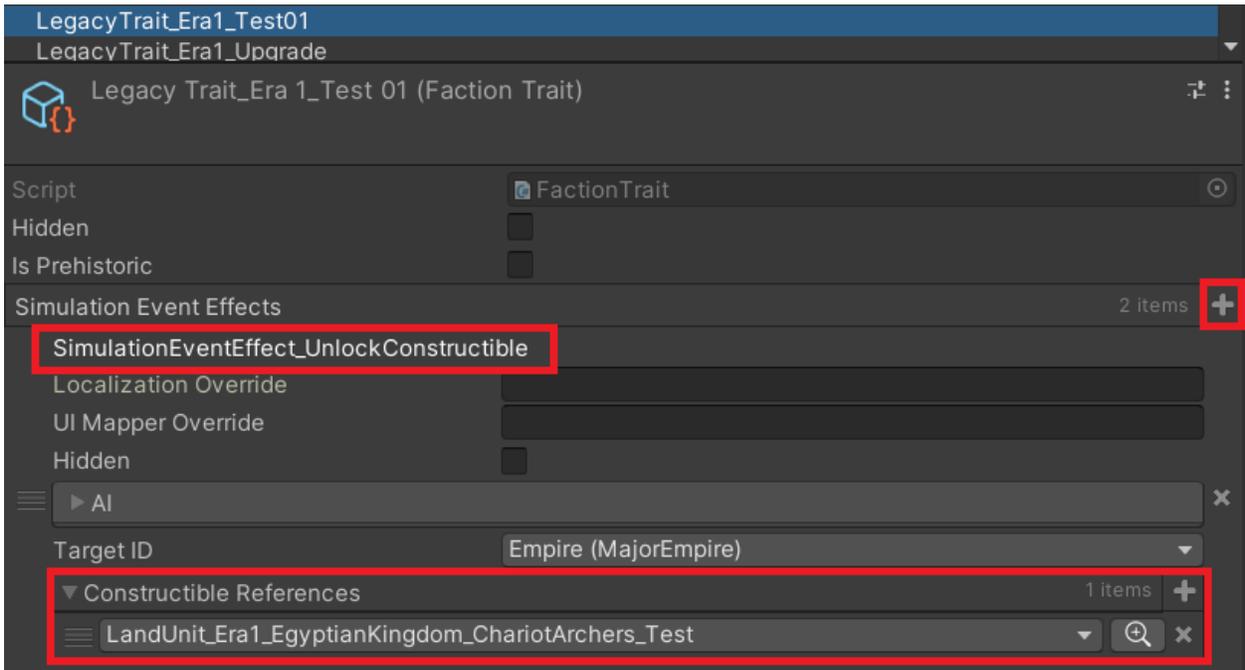
2. Select the unit definition, go to the “Prerequisites” tab and add the new culture under “Faction Names”.



**Note!** Make sure the presentation pawn and unit definitions are created for the new unit.

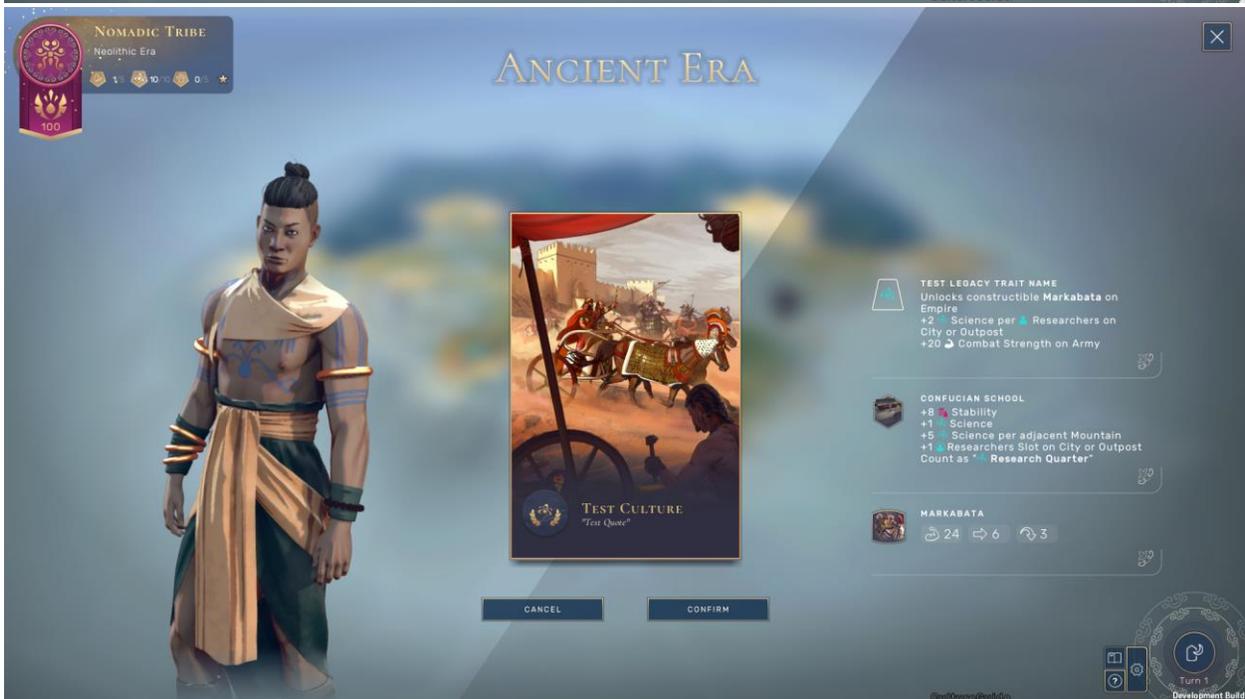
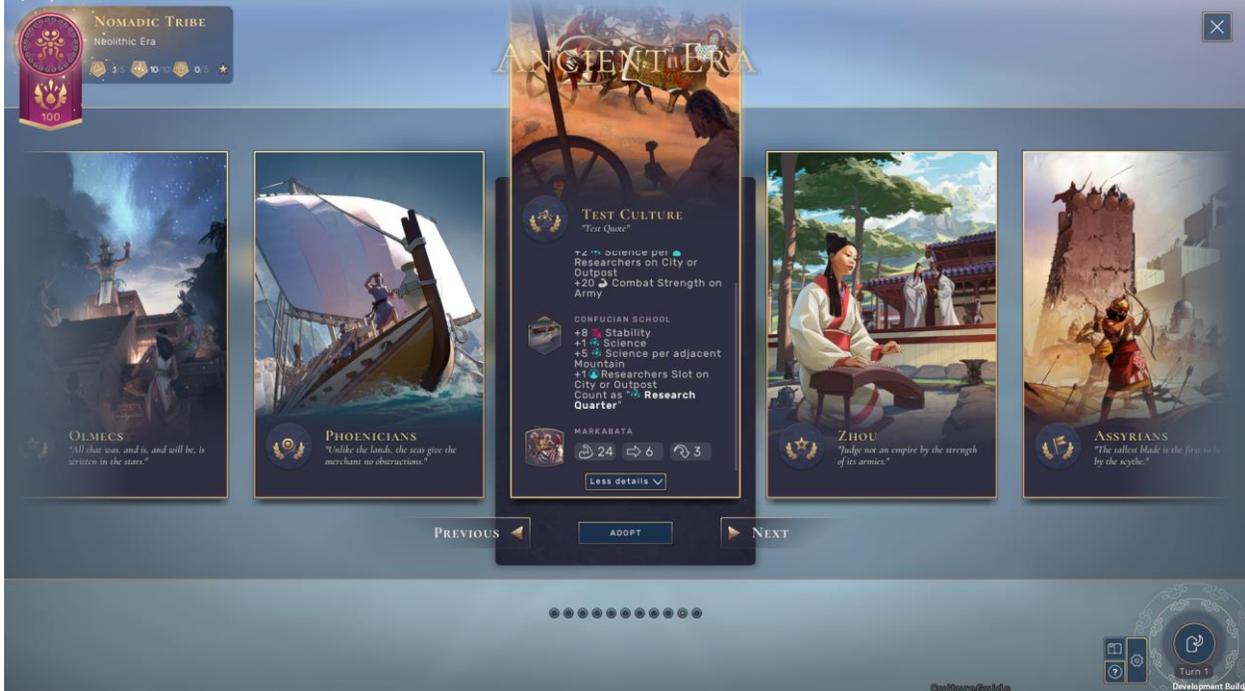


3. Go to civilization legacy trait and add a new simulation event that will unlock the unit.



## 10.14 Testing

After all steps were done - click "Build and Run" to check intermediate results. Start a new game and reach the era in which a newly created civilization appears (depends on prerequisites).



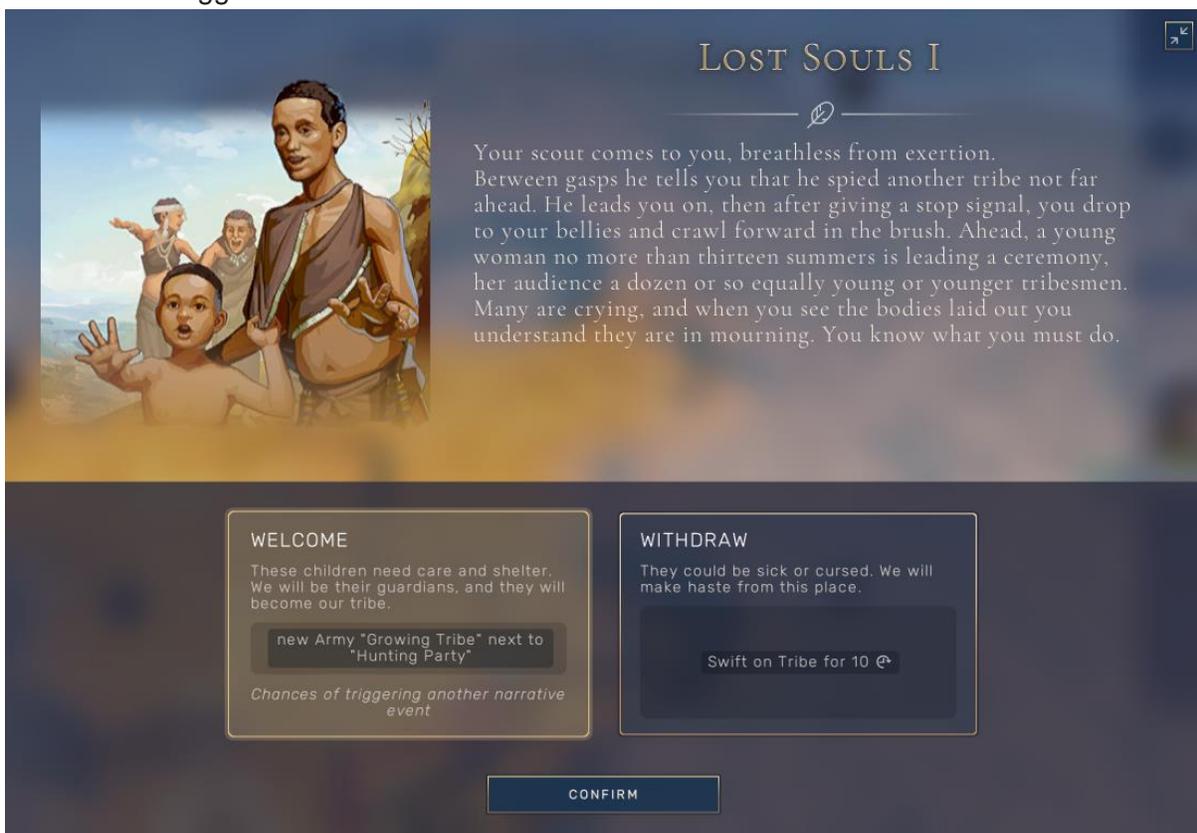
Existing emblematic unit and district were added to the culture.

# 11 Adding a new Narrative Event

Narrative events happen during the game and give short narrative stories with up to 3 decisions that affect the gameplay. Event choices may modify ideological axis, give buffs/debuffs to the Empire, spawn units, unlock civics or reduce technology cost.

Event consist of:

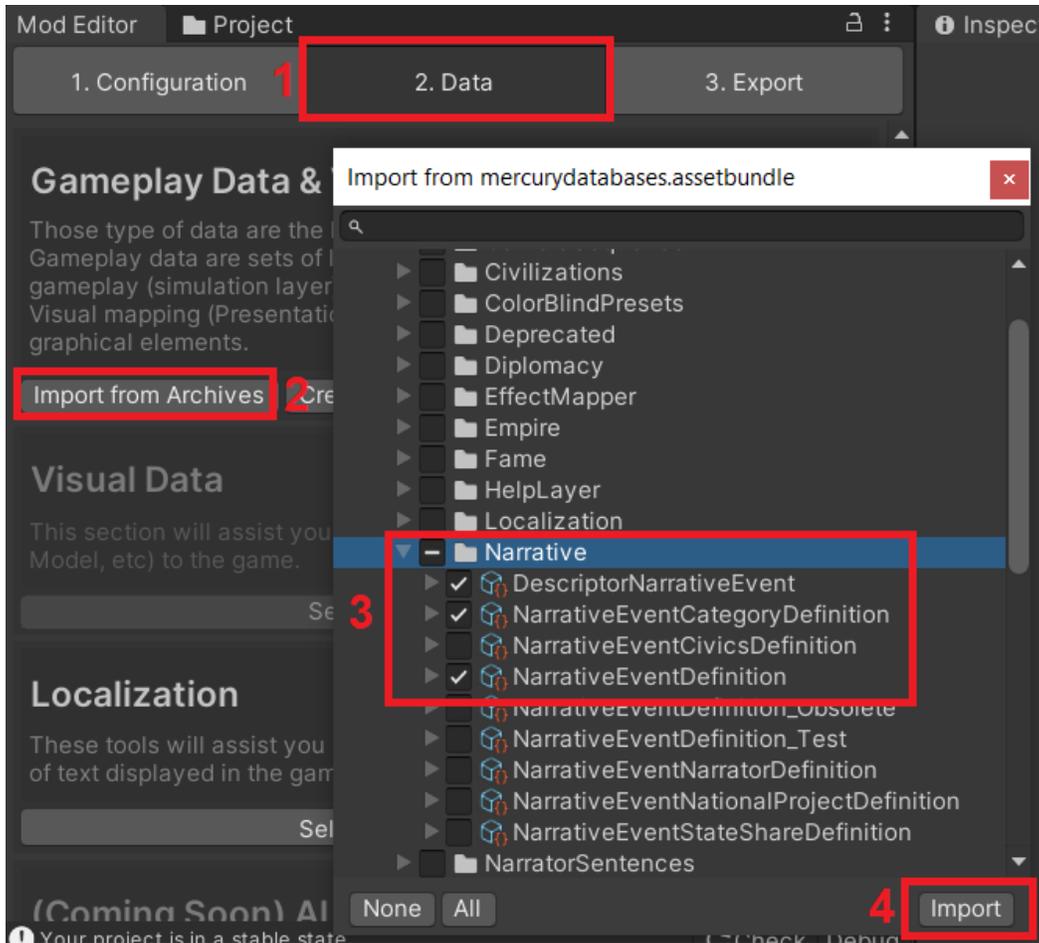
- name;
- narrative story;
- choices with some effects (including ideological axis changes);
- image;
- prerequisites;
- triggers for other events.



## 11.1 Setting up the environment

To modify or create a new event, the "Narrative" database has to be used:

1. Go to "2. Data" of the Mod Editor.
2. Select "Import from Archives".
3. Select the "Narrative" database.
4. Press "Import".

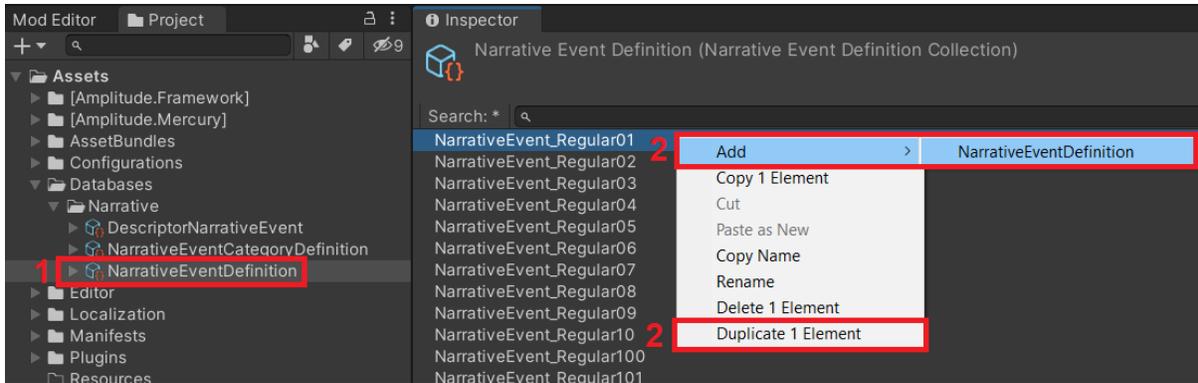


**Important!** It is not recommended to store unused objects within the mod project structure. Please either not export unused objects, or remove unnecessary objects from collections/collections after the mod is created.

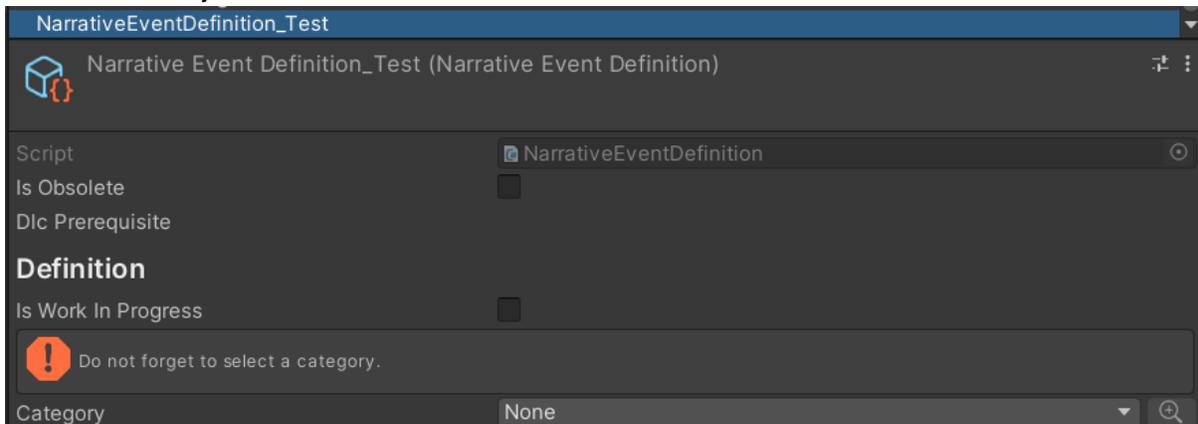
## 11.2 Create a new narrative event

To create a new narrative event into the game, the event definition object has to be created:

1. Select the "NarrativeEventDefinition" collection in the project structure.
2. Add a new element by right-clicking in the inspector space and selecting "Add-> NarrativeEventDefinition" or simply duplicate an existing one.



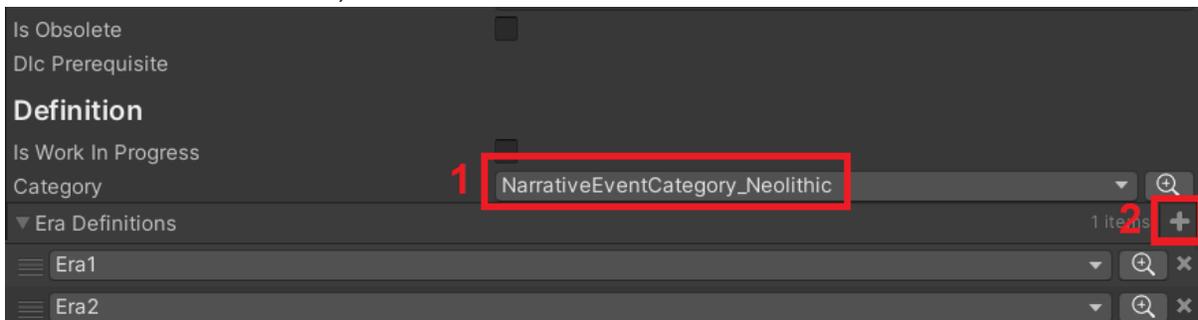
After the new definition appears in the same inspector window, it's strongly recommended to rename the object.



### 11.3 Setting up the “Definition” section

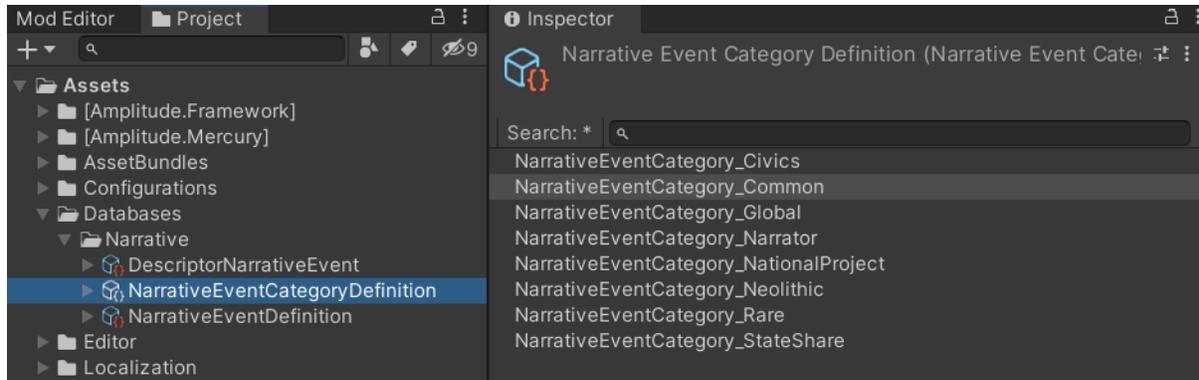
The “Definition” section allows to select the era in which the event occurs and the event's category (related to triggering process and priority).

1. Set the event category.
2. Add as many eras as required into the “Era Definitions” list (defines in which eras event occurs).



## 11.4 Event category

All event categories are stored in the “NarrativeEventCategoryDefinition” collection. It defines the priority of the event’s triggers, dead zones for triggering and some other technical stuff.



Each category has the following fields:

- The “Priority of the category” - the highest priority will be triggered upon all other categories.
- The “Global” defines if narrative events occur for AI.
- The “Narrative Event Definition Distribution” defines sorting for events upon 1 category.
- The “Need Manual Trigger” is used for linking categories to manual events like building national projects.
- The “Maximum Number Of Concurrent Instances” defines how many events from a category can happen at one turn.
- The “Consume Event Upon Trigger” blocks events from happening again.
- The “Trigger Distribution Range” defines the chance of the event happening within a specific range.
- The “Number of Turns In Dead Zone...” number of turns before the event happens after another event triggering and era changes.

NarrativeEventCategory_Neolithic	
NarrativeEventCategory_Rare	
NarrativeEventCategory_StateShare	
Narrative Event Category_Neolithic (Narrative Event Category Definition)	
Script	NarrativeEventCategoryDefinition
Is Obsolete	<input type="checkbox"/>
<b>Definition</b>	
Priority	2
Global	<input type="checkbox"/>
Narrative Event Definition Distribution	Sort Randomly
<b>Trigger</b>	
Need Manual Trigger	<input type="checkbox"/>
Maximum Number Of Concurrent Instances	1
Consume Event Upon Trigger	<input checked="" type="checkbox"/>
 Distributes the chance to trigger an event within the specified range, expressed in number of turns. Set to 0 to disable the distribution.	
Trigger Distribution Range	3
Number Of Turns In Dead Zone After Era Ch	1
Number Of Turns In Dead Zone After Trigger	3
No UI	<input type="checkbox"/>
Picto	None (Texture)
No Restriction On Timing	<input type="checkbox"/>

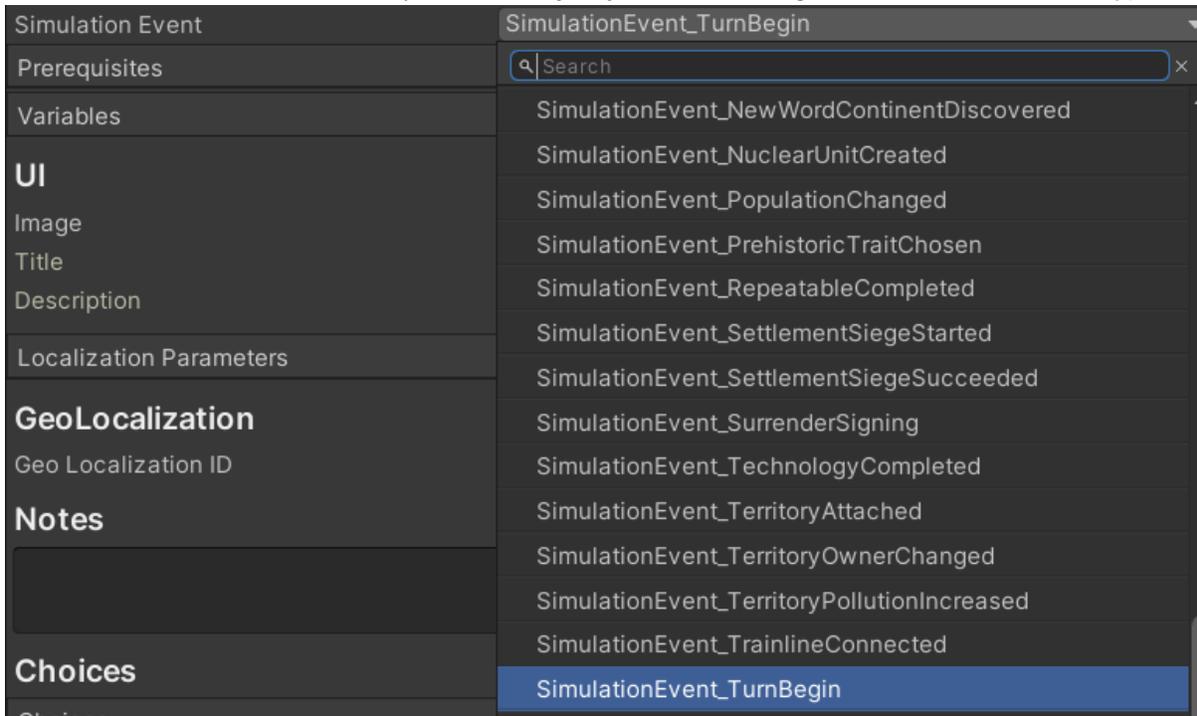
## 11.5 Setting up the “Trigger” section

The “Trigger” section defines on which conditions the event occurs.

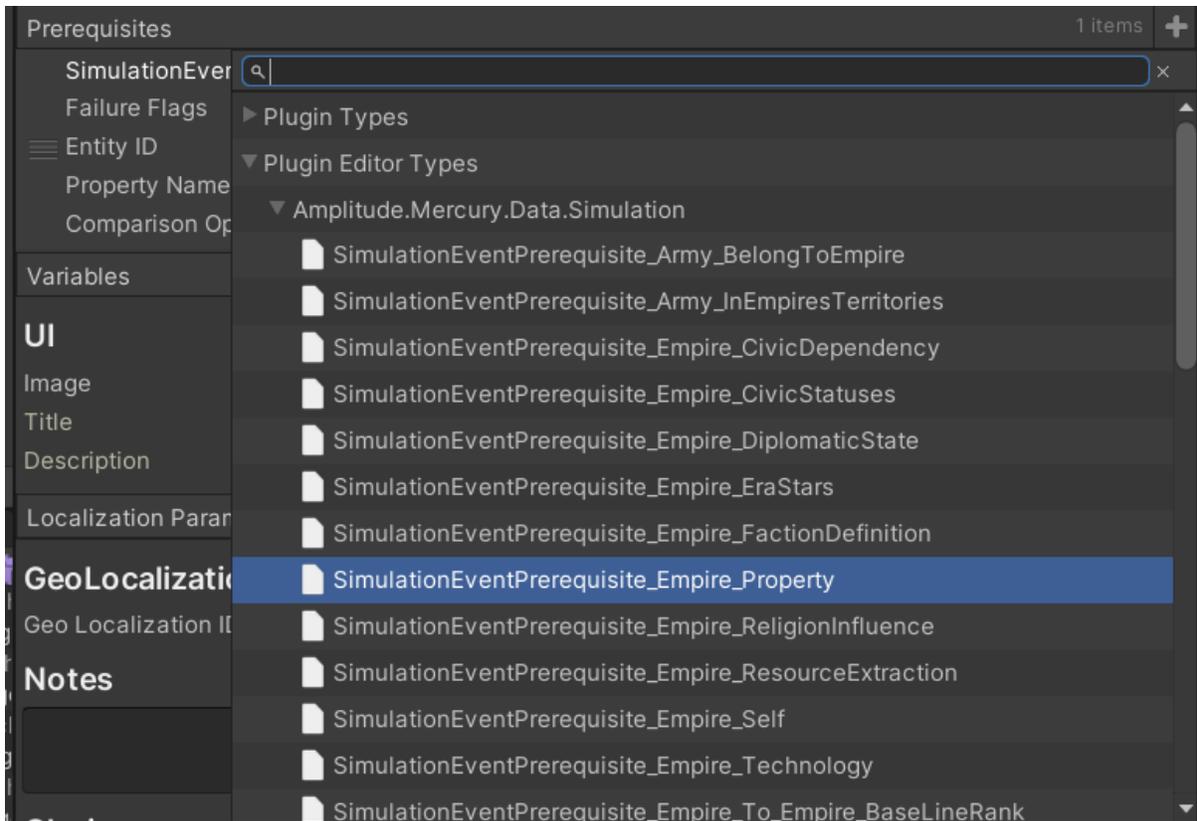
- Set some trigger options as required:
  - The “Consume Event Upon Trigger” blocks events from happening again. Same as from category. Please note that if the event with higher priority (higher category) occurred when this function is on, the event won’t appear again.
  - The “Prevent Event Removal” defines if the event can be repeated.
  - The “Is Narrative Consequence” is used for events that appear as a consequence of another event. If this option is selected, the following “Prerequisites” and “Variables” fields can be empty.

<b>Trigger</b>	
Consume Event Upon Trigger	<input checked="" type="checkbox"/>
Prevent Event Removal	<input type="checkbox"/>
Is Narrative Consequence	<input type="checkbox"/>

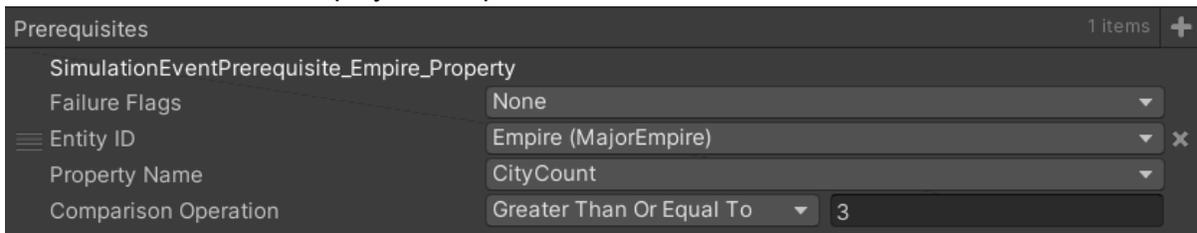
2. Select the preferable “Simulation Event”. It defines when the event occurs (after which action/on which occasion). The simplest way to work with the field is to set the “SimulationEvent\_TurnBegin” and make checks within the “Prerequisites” and “Variables” fields (the vast majority of the existing events are built this way).



3. The “Prerequisites” checks the condition on which a simulation event occurs. It is possible to have several prerequisites for 1 event. Different prerequisites have different structures, refer to existing events for more examples. Let’s check if the empire has 3 or more cities within eras 1 or 2 using “SimulationEventPrerequisite\_Empire\_Property”.

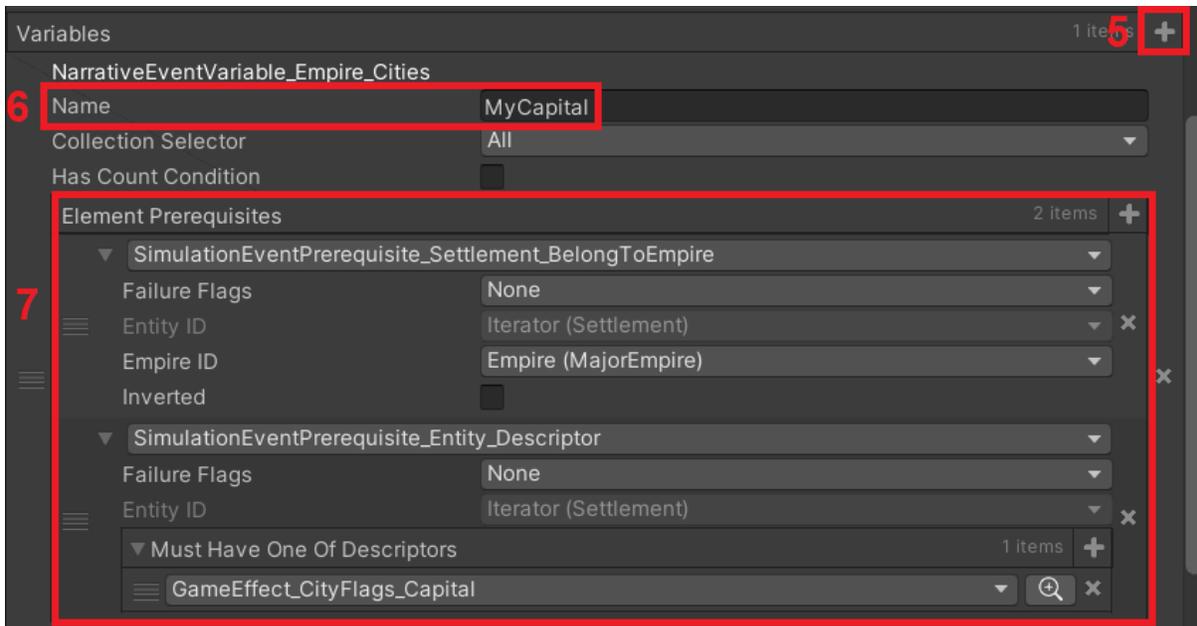


4. Fill the related fields of the selected prerequisites. The “Entity ID” has to be “Empire (MajorEmpire)” to interact with the player's one. Our prerequisite will be checking whether the player’s empire has more than 3 cities within 2 eras.



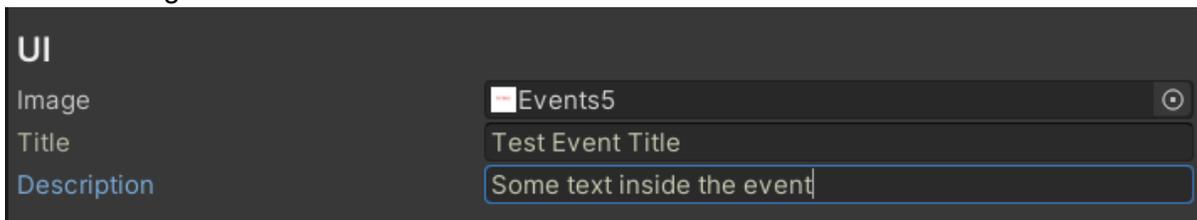
5. The “Variables” checks properties of assets for “GeoLocalization”, check statuses and properties of objects referred to in “Prerequisites”, used for localization. Add the variable “NarrativeEventVariable\_Empire\_Cities” to make the event appear in the capital.
6. Name the variable.
7. Set the preferable prerequisites for the variable. The current example checks whether the city belongs to a major empire with the “SimulationEventPrerequisite\_Settlement\_BelongToEmpire” and checks if the city has a capital flag with the “SimulationEventPrerequisite\_Entity\_Descriptor”.

**Note!** The best way to learn about all possible variables is to check the existing events.



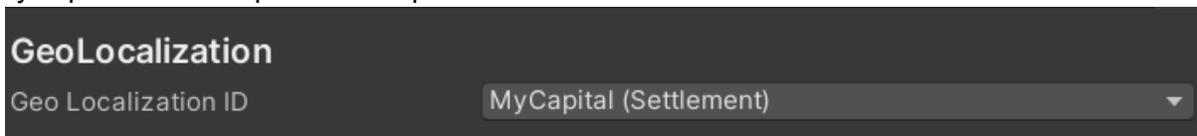
## 11.6 Setting up the “UI” section

Fill the corresponding fields with image, title and description of the event. Remember to locate the image in the “Resources” folder.



## 11.7 Setting up the “GeoLocalization” section

It defines where the event will occur. In the example it is assigned to the variable “MyCapital” from the previous step.



## 11.8 Setting up the “Choices” section

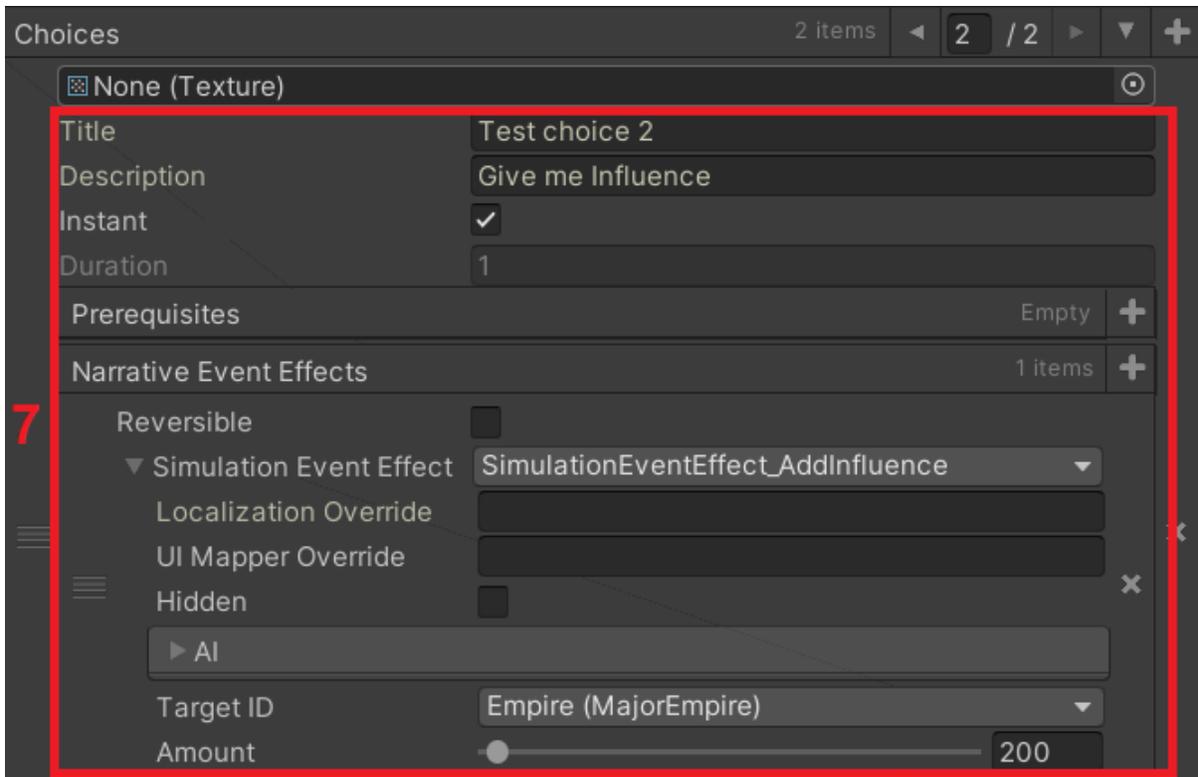
Choices allow the player to select the preferable effect from the narrative event to solve the dilemma of the event. Currently, up to 3 choices can be displayed simultaneously. To configure choices:

1. Add 2 empty choices to the event with a plus button.
2. Fill in the title and the description of the choice.

- Define the choice either instant (used for civics or unlocks) or not (for city statuses or some lasting bonuses). It is also possible to define the length of the effect from the descriptor directly.
- Add the “Prerequisites” for the choice if needed. This field defines if the specific choice will be shown to the player based on the game condition. Usually, the field is empty, but if the choice is related to civic status or religion, this field can be configured.
- Add the “Narrative Event Effects” as much as required to give rewards, bonuses, assign statuses or hidden tags.
- Fill in the required narrative effect. The dropdown list of the “Simulation Event Effect” is self-explanatory.
- Repeat steps 2-6 for the second choice.

The screenshot shows the 'Choices' configuration window. Red boxes and numbers 1 through 6 highlight the following elements:

- 1:** The top right corner containing a list view indicator '2 items', a page number '1 / 2', and navigation arrows.
- 2:** The 'Title' field containing the text 'Test choice 1'.
- 3:** The 'Description' field containing the text 'Give me Fame'.
- 4:** The 'Instant' checkbox, which is checked, and the 'Duration' field containing the value '1'.
- 5:** The 'Prerequisites' field, which is empty, and the 'Narrative Event Effects' field, which contains '1 items'.
- 6:** A large red box encompassing the 'Narrative Event Effects' configuration panel, including:
  - 'Reversible' checkbox (unchecked)
  - 'Simulation Event Effect' dropdown menu (selected: SimulationEventEffect\_AddFame)
  - 'Localization Override' text field
  - 'UI Mapper Override' text field
  - 'Hidden' checkbox (unchecked)
  - 'AI' button
  - 'Target ID' dropdown menu (selected: Empire (MajorEmpire))
  - 'Amount' slider (set to 100)

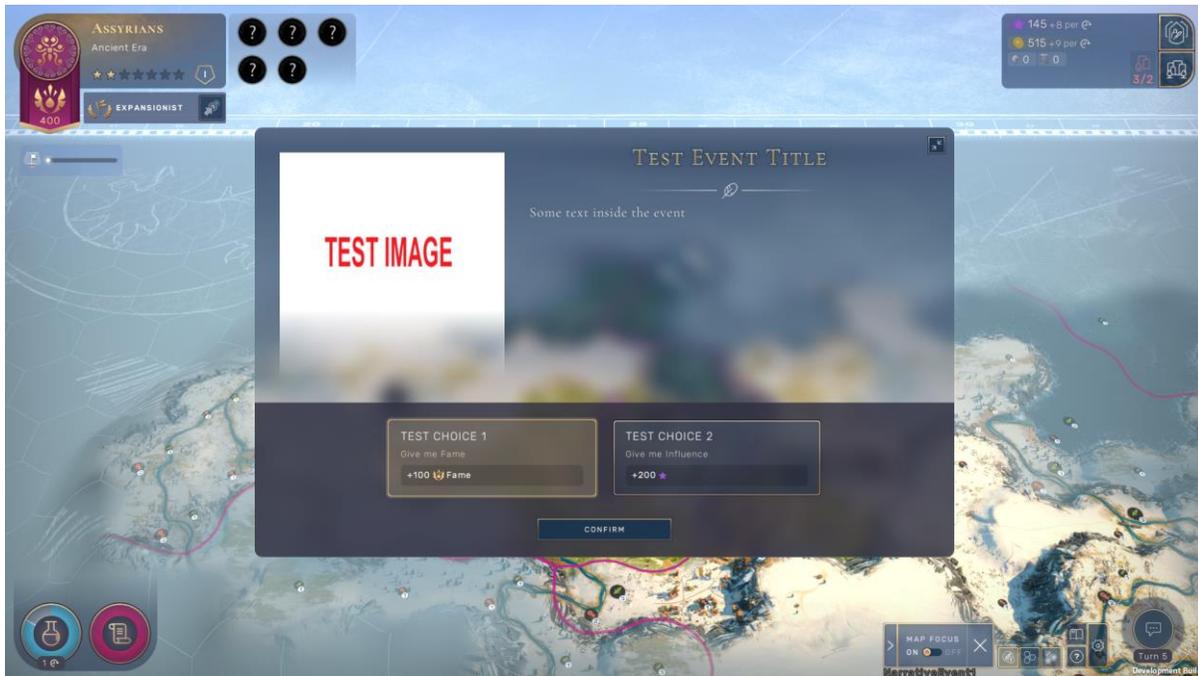


## 11.9 Testing

After all steps were done - click "Build and Run" to check intermediate results. Start a new game, go to the needed era and complete the prerequisite.



3 cities were built and as of the next turn the event occurred.



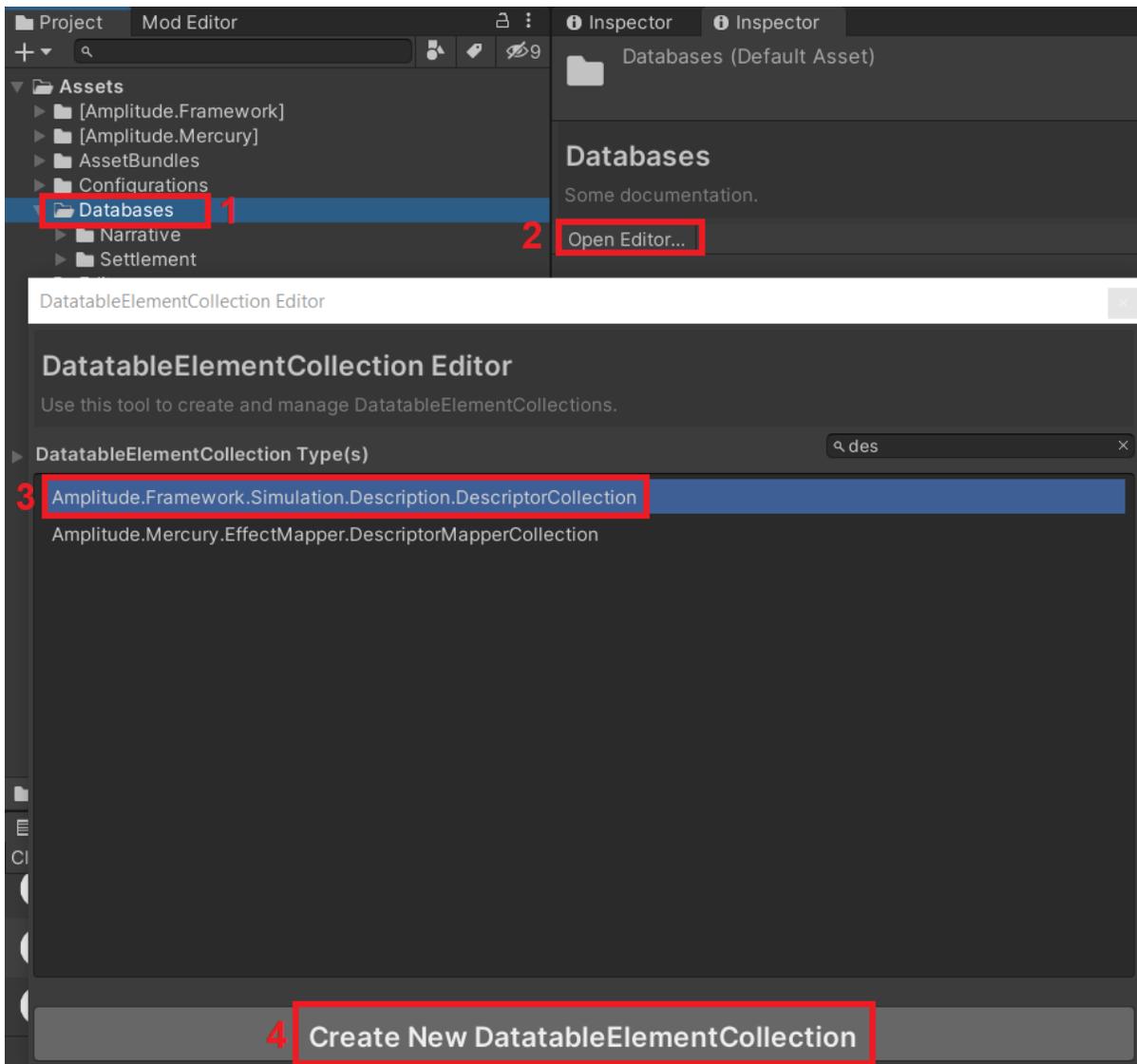
## 11.10 Adding a lasting effect

To give the effect/bonus which lasts for several turns, the corresponding definition and descriptor should be created and assigned to the choice. Current lasting effects/bonuses are stored in different databases and splitted by objects which receive the effect.

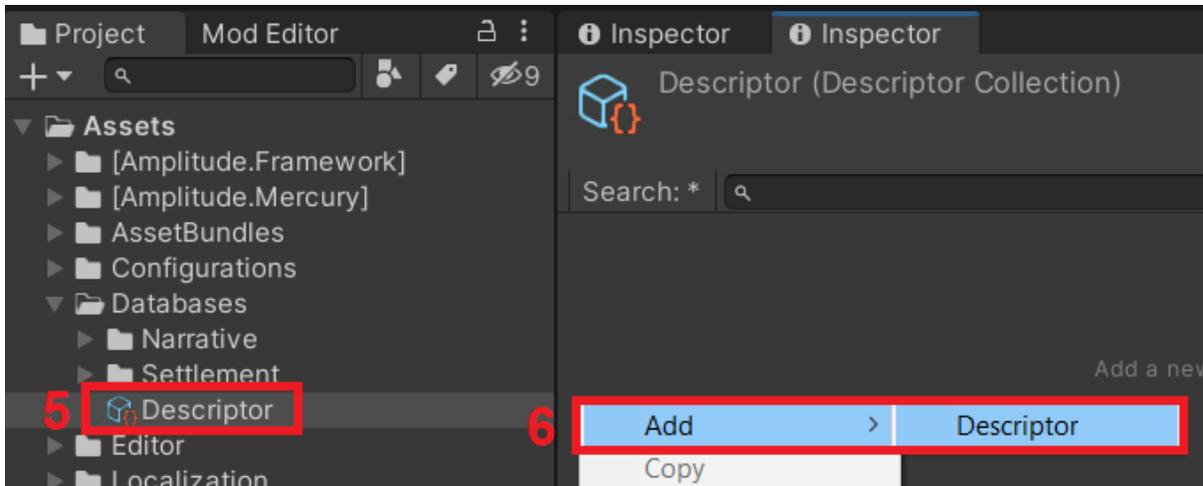
### 11.10.1 Adding a descriptor

To give the effect/bonus which lasts for several turns, the corresponding descriptor should be created and assigned to the choice. Current lasting effects/bonuses are stored in different collections and splitted by objects which receive the effect.

1. To add the descriptor and the collection press in the “Databases” folder in the project structure. It is also possible to create a descriptor in the existing collection, export the necessary one and skip steps 2-3.
2. Press “Open Editor...”.
3. Select the “...DescriptorCollection”.
4. Press the button to create a new. Rename collection if needed.

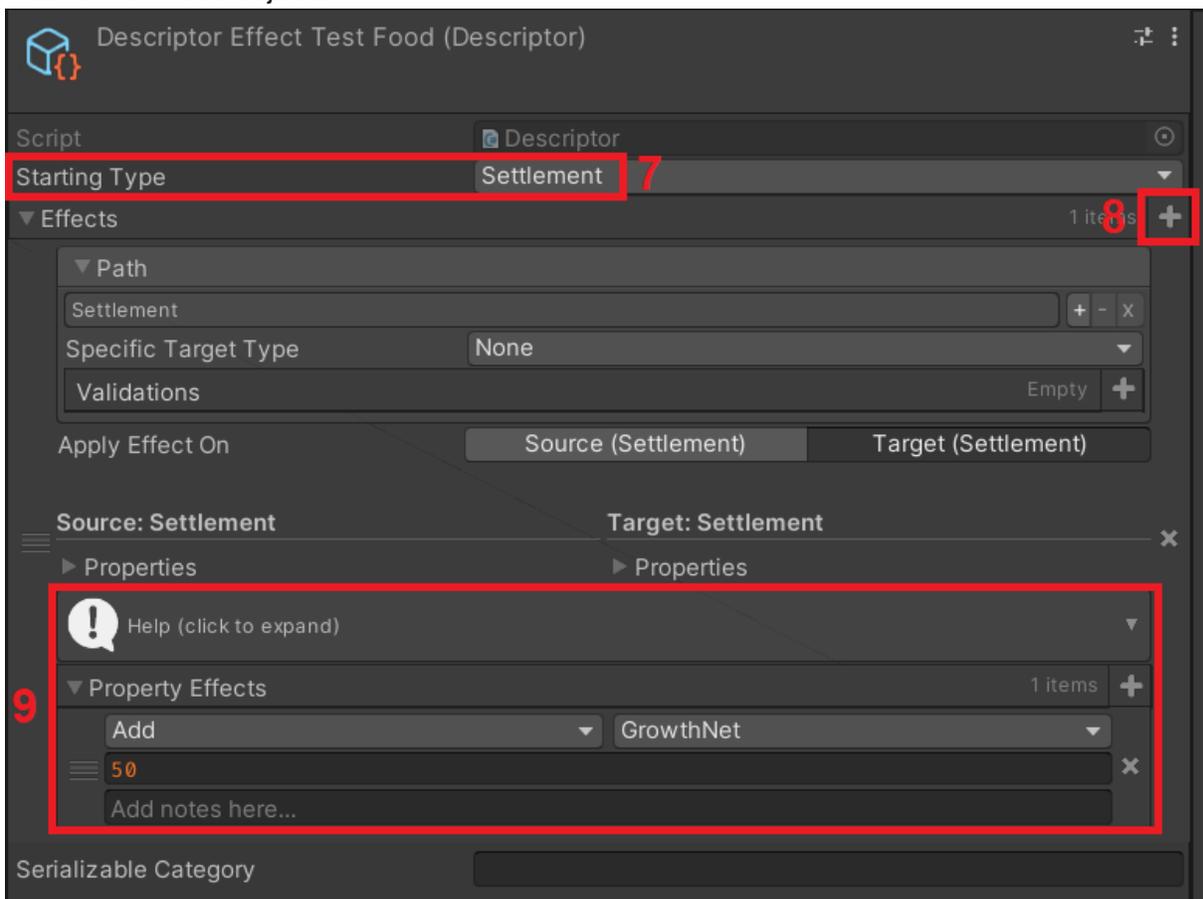


5. Select the new collection.
6. Add a new descriptor to the collection for the effect for choice 1. Rename object if needed.

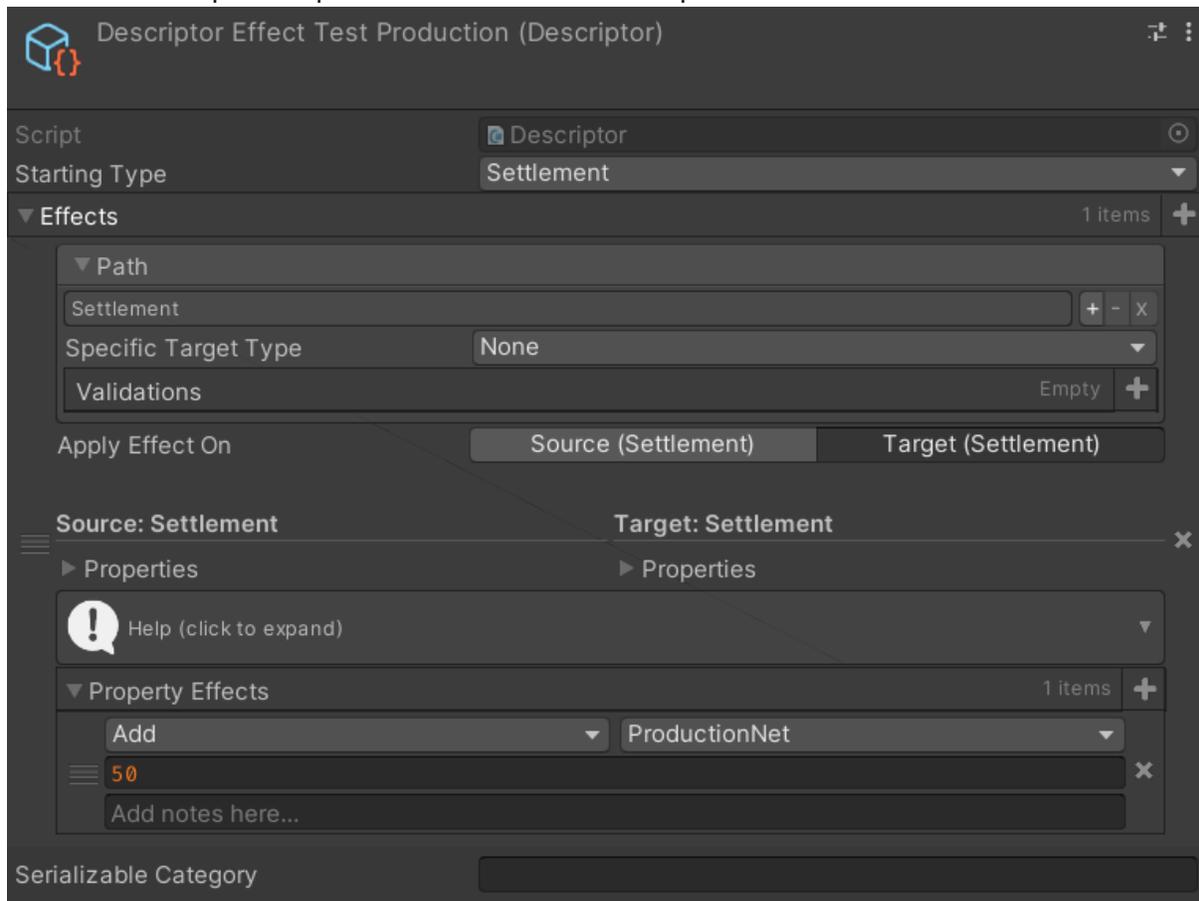


7. Select the “Starting Type” for the descriptor. It defines where the effect is applied.
8. Add the new effect.
9. Add the “Property Effects” as needed and complete the related fields. The example descriptor will be giving the food in the city.

**Note!** It is possible to assign a more specific path for precise effect on objects and set validation for these objects.



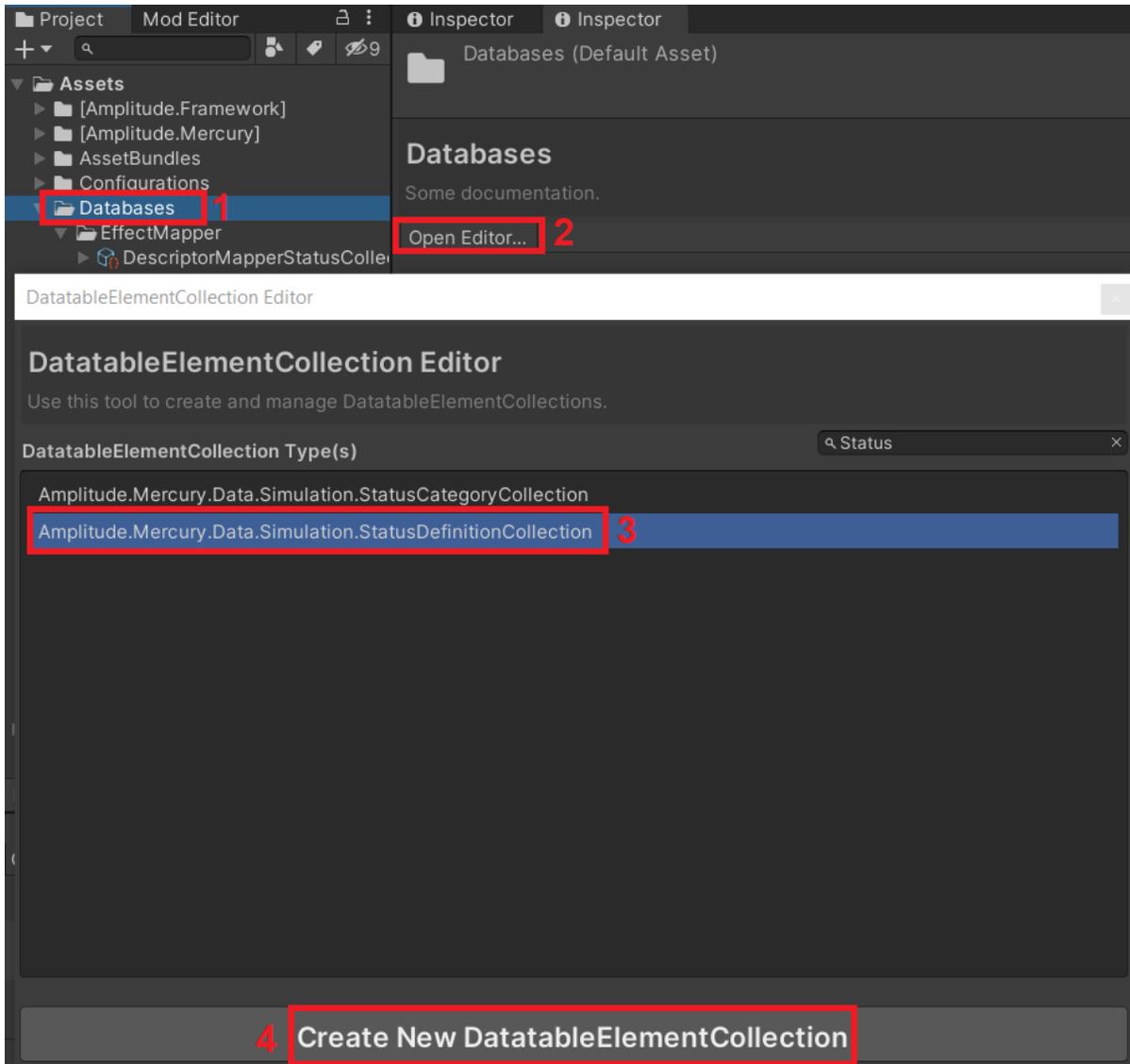
10. Repeat steps 6-9 to add another descriptor for another effect for choice 2.



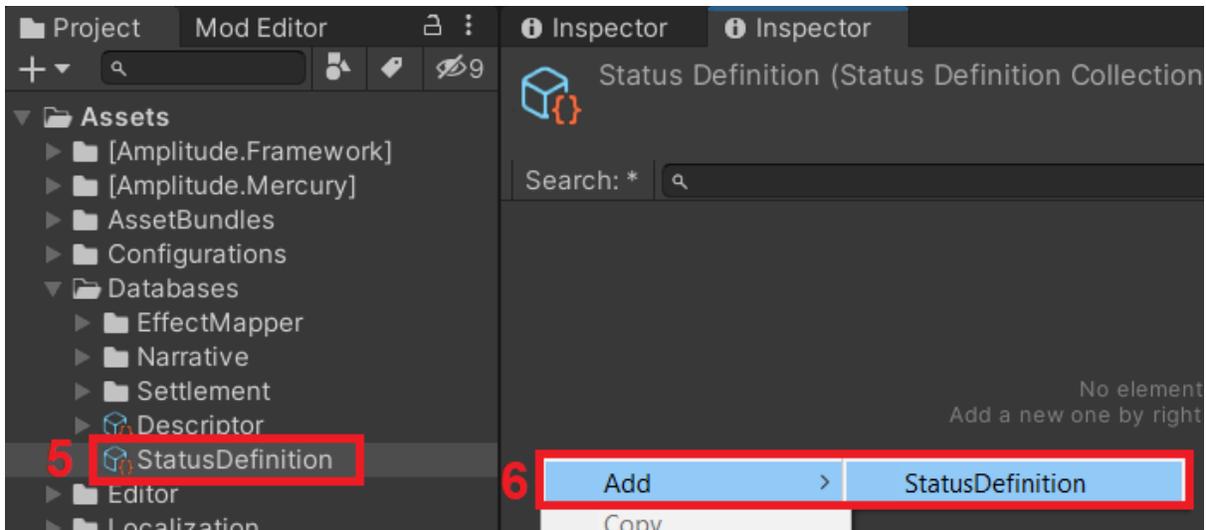
### 11.10.2 Adding a status definition

To assign the descriptor effect properly, the status definition (works as placeholder of the descriptor) must be created.

1. To add the status definition and the collection press in the “Databases” folder in the project structure. It is also possible to create a status definition in the existing collection, export the necessary one and skip steps 2-3.
2. Press “Open Editor...”
3. Select the “...StatusDefinitionCollection”.
4. Press the button to create a new one. Rename collection if needed.

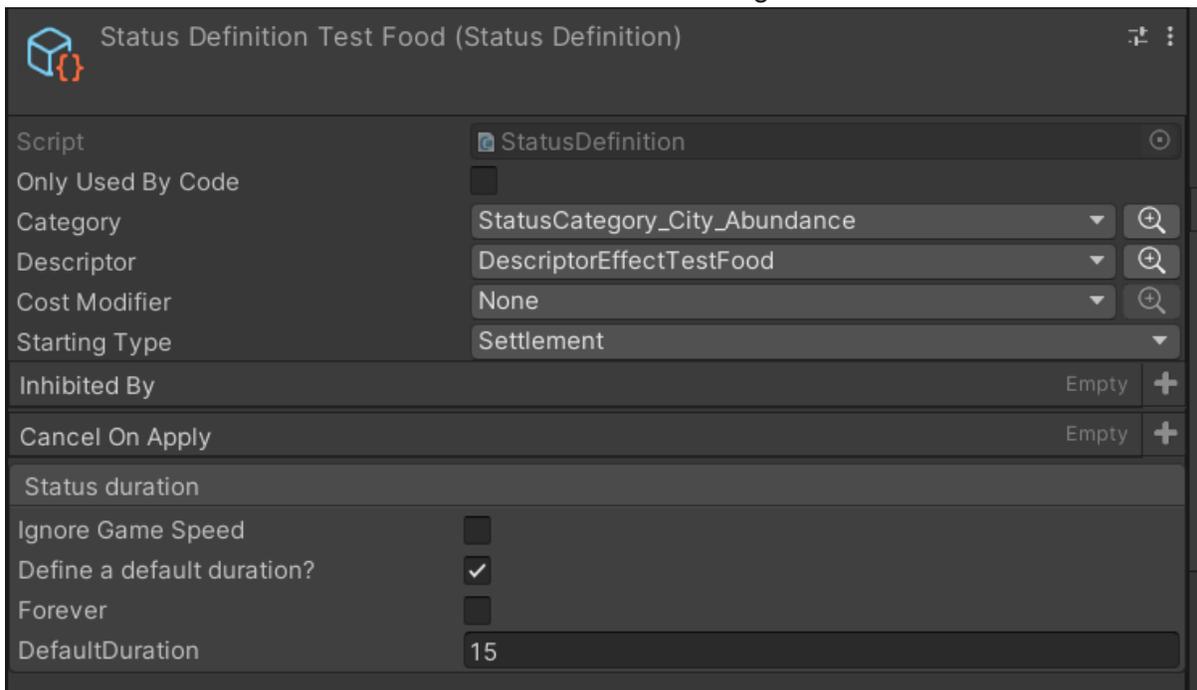


5. Select the new collection.
6. Add a new definition to the collection. Rename object if needed.



7. Complete all the fields according to the need:

- The “Category” defines some audio/visual effects.
- The “Descriptor” assigns the effect to the status. Created in previous steps.
- The “Cost Modifier” changes the cost of something.
- The “Starting Type” defines the objects types on which effects will be applied.
- The “Inhibited By” defines if the status can be canceled by another category.
- The “Cancel on Apply” defines which category will be canceled if the current status is applied.
- The “Status duration” defines how long the effect lasts.



- Repeat steps 6-7 to create another status for the second choice and the second descriptor.

Script: StatusDefinition

Only Used By Code:

Category: StatusCategory\_City\_Overproduction

Descriptor: DescriptorEffectTestProduction

Cost Modifier: None

Starting Type: Settlement

Inhibited By: Empty

Cancel On Apply: Empty

Status duration

Ignore Game Speed:

Define a default duration?:

Forever:

DefaultDuration: 15

**Note!** Refer to existing statuses and status categories for more information.

### 11.10.3 Applying a lasting effect to choices

To apply the new choices to the correct objects (all cities in the example) the variable has to be created in the narrative event. The “Collection Selector” stands for the number of affected cities.

NarrativeEventVariable\_Empire\_Cities

Name: AllCities

Collection Selector: All

Has Count Condition:

Element Prerequisites: 1 items

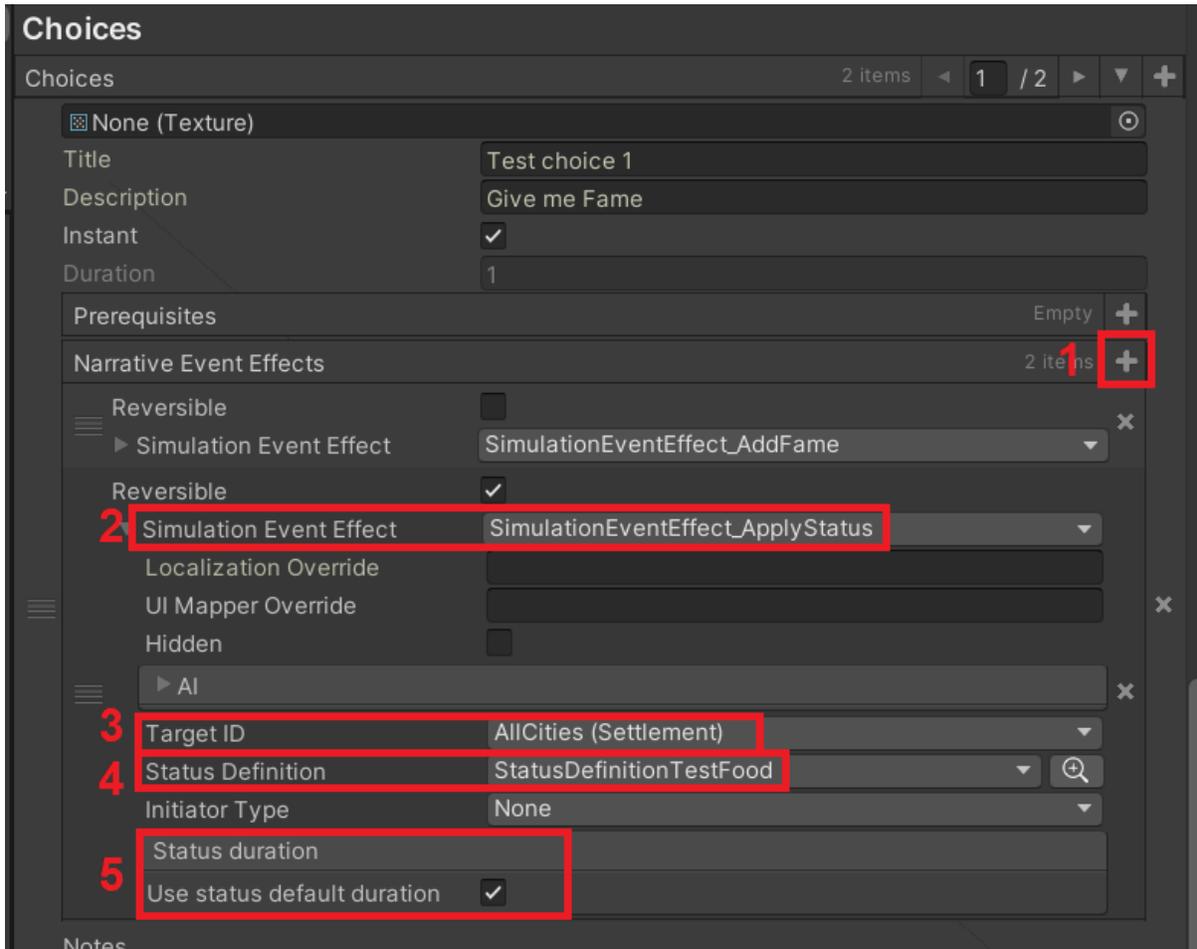
- SimulationEventPrerequisite\_Settlement\_BelongToEmpire
  - Failure Flags: None
  - Entity ID: Iterator (Settlement)
  - Empire ID: Empire (MajorEmpire)
  - Inverted:

Sort Property Name: None

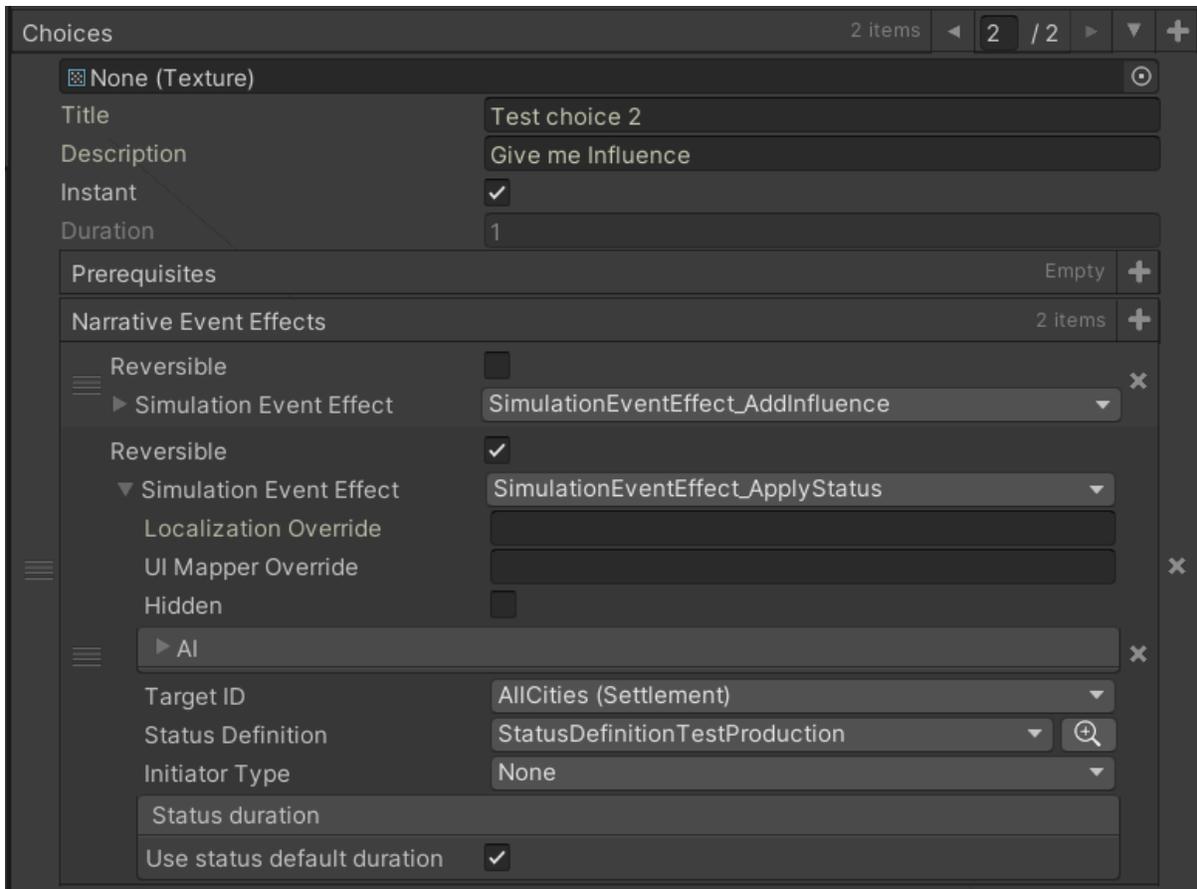
Empire ID: Empire (MajorEmpire)

- Go to choices and add a new “Narrative Event Effect” to choice 1.
- Set the “Simulation Event Effect” to “...ApplyStatus”.

3. Set the “Target ID” to the newly created variable of cities.
4. Set the “Status Definition” to the recently created status.
5. Set the “Status duration” to the default one.



6. Repeat steps 1-5 for the second choice.

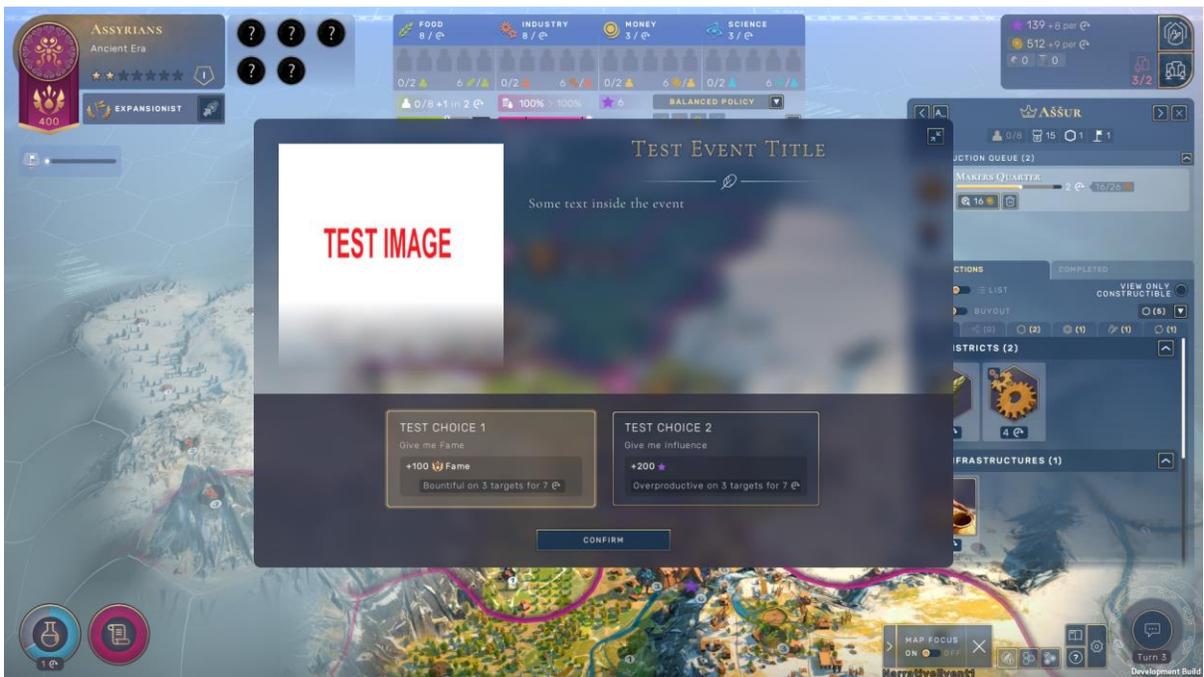


## 11.11 Testing

After all steps were done - click "Build and Run" to check the result. Start a new game, go to the needed era and complete the same prerequisites (having 3 cities in the example).



3 cities were built and as of the next turn the event occurred.



Effect on different cities:



## 11.12 Adding narrative consequences

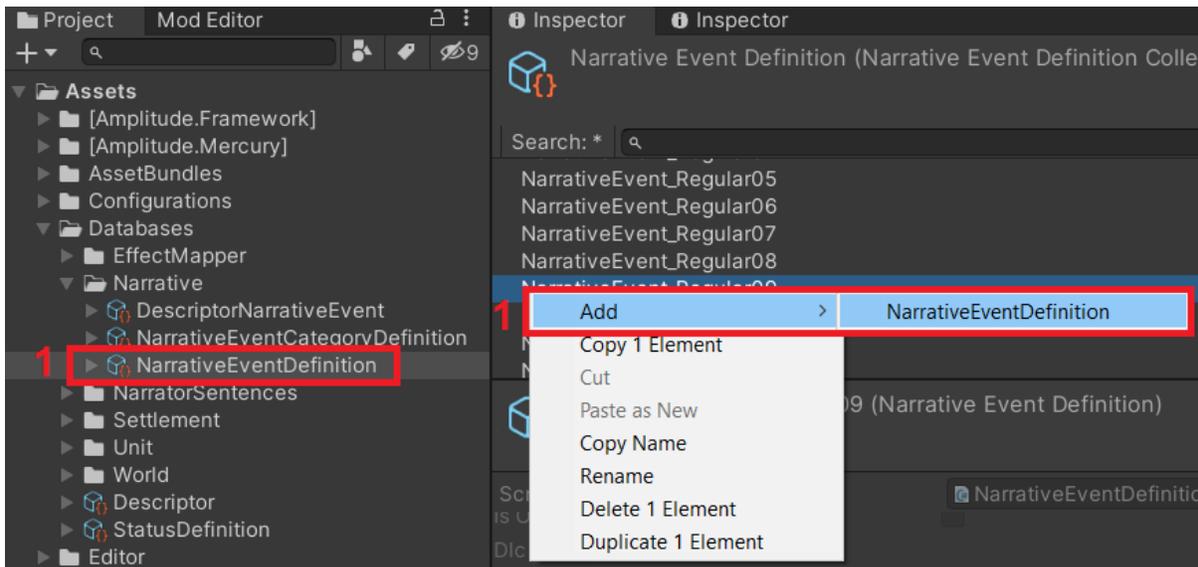
There are 2 main ways to connect different events between each other:

1. Use the narrative consequence option.
2. Make an invisible tag.

### 11.12.1 Narrative consequence

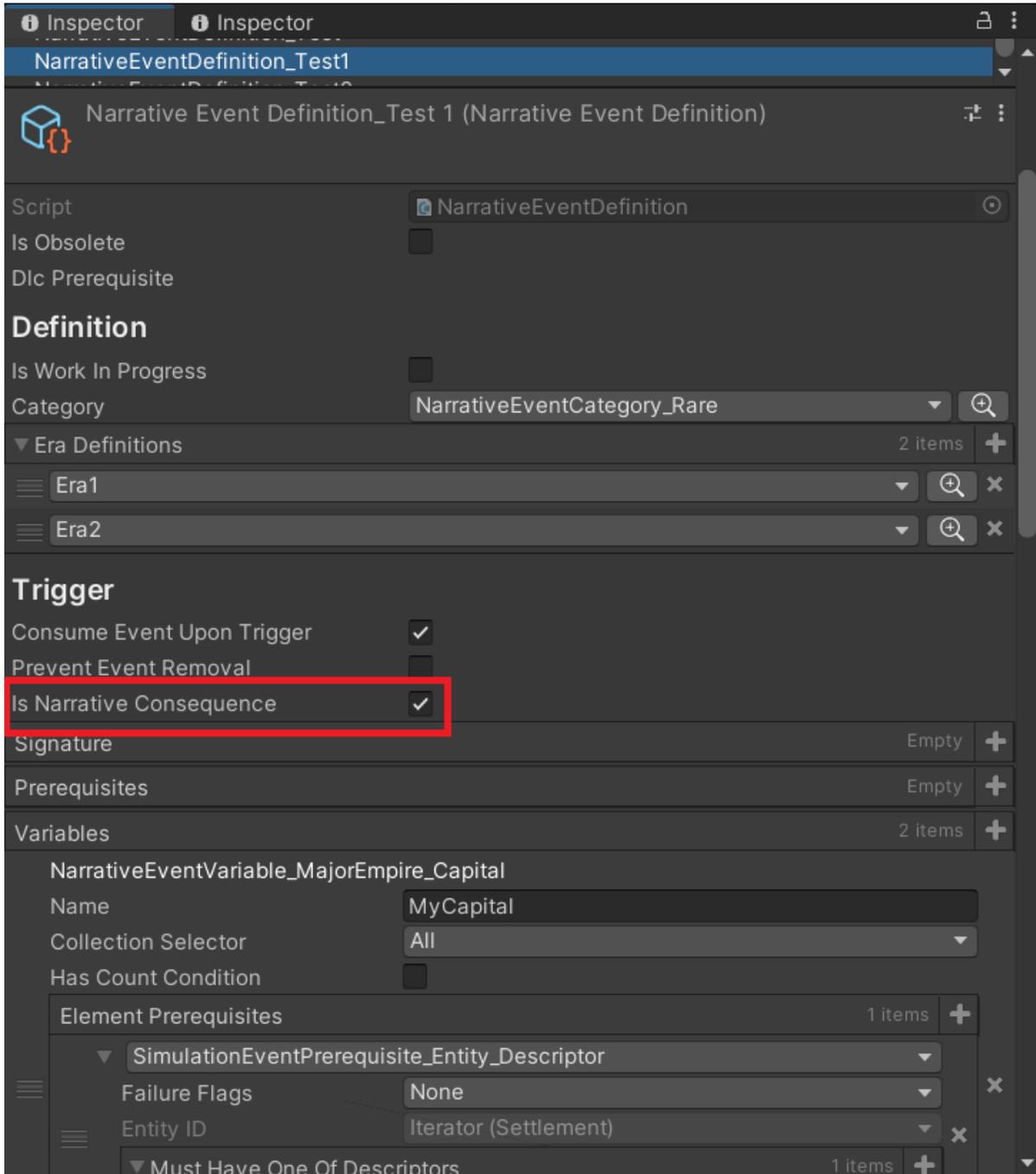
To connect 1 event to another as a consequence:

1. Create a new event that will be used as a consequence in the existing narrative collection.

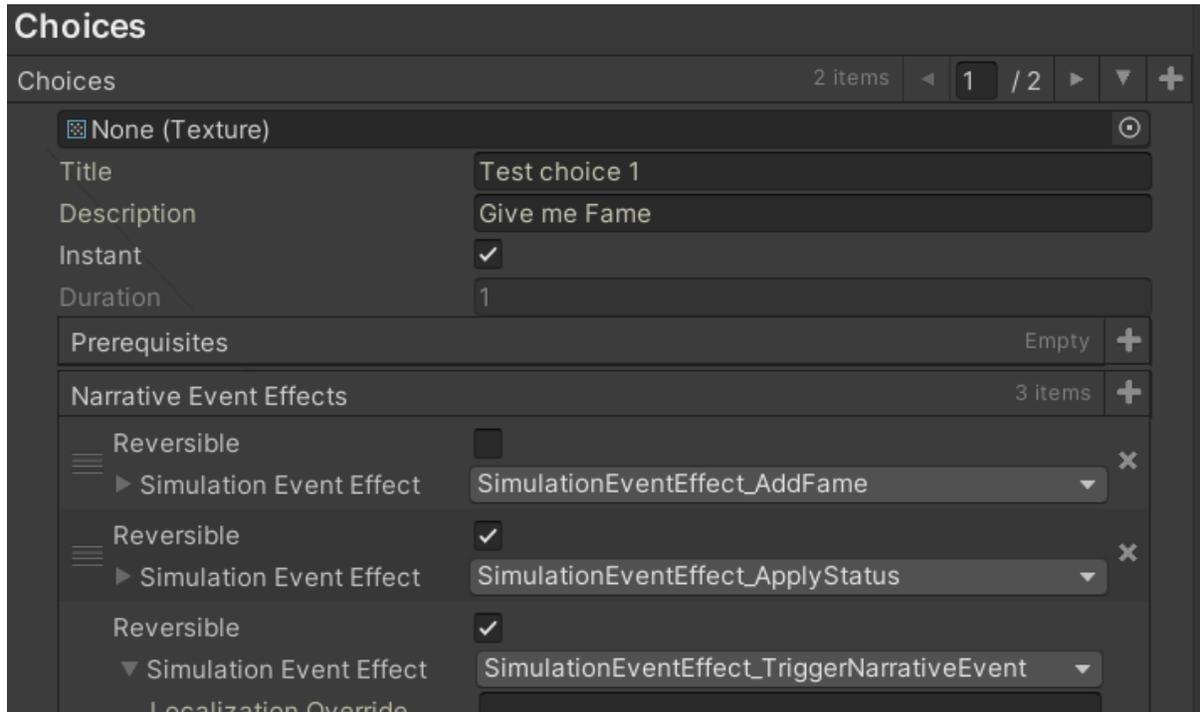


2. Fill the event as it was done in previous steps according to the preferable result. Make sure the "Is Narrative Consequence" is checked. The example event will give additional speed to units.

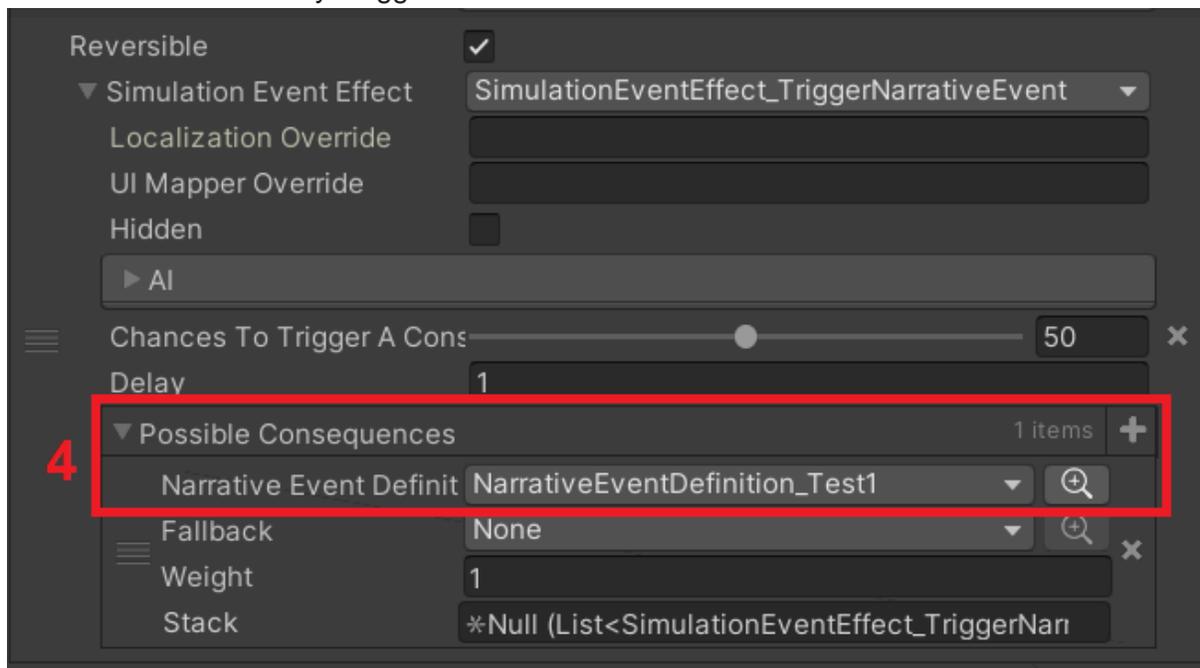
Once the “Is Narrative Consequence” is checked, the new “Signature” section appears. It can be manually filled with the variables from the initial event.



- Get back to the initial event and add the new narrative event effect “SimulationEventEffect\_TriggerNarrativeEvent” to one of the choices. It will trigger the consequence event.

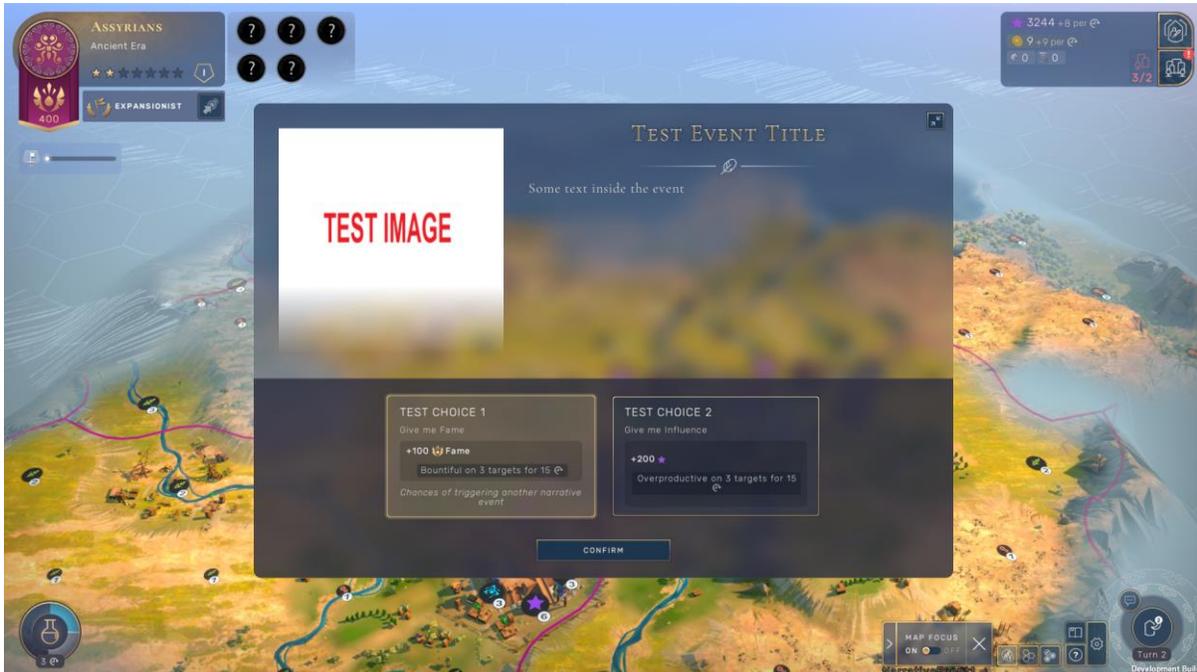


- Set the event which will be a consequence and complete fields. The “Fallback” is used to trigger another event if the first one cannot be triggered for some reason. **Note!** Field “Delay” won’t give the delay in turns to the consequence event. The event always triggers the next turn.

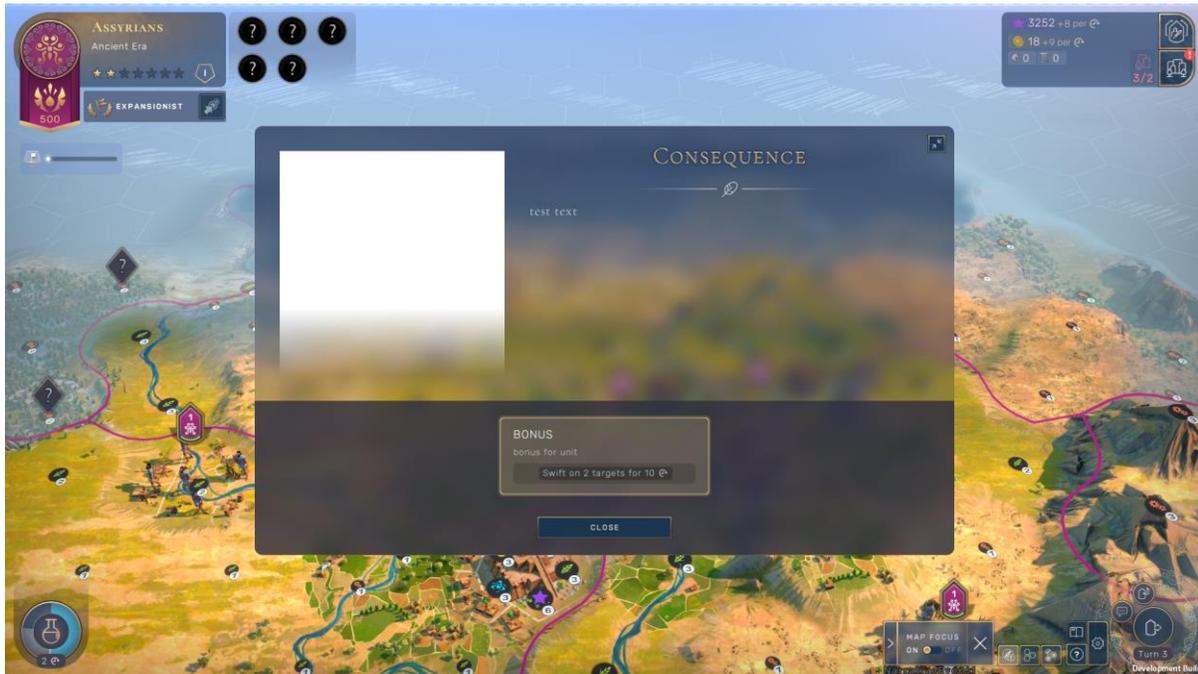


## 11.12.2 Testing

After all steps were done - click “Build and Run” to check the results. Start a new game, go to the needed era and complete the same prerequisites (having 3 cities in the example). Pick the choice which has a narrative consequence and finish the turn.



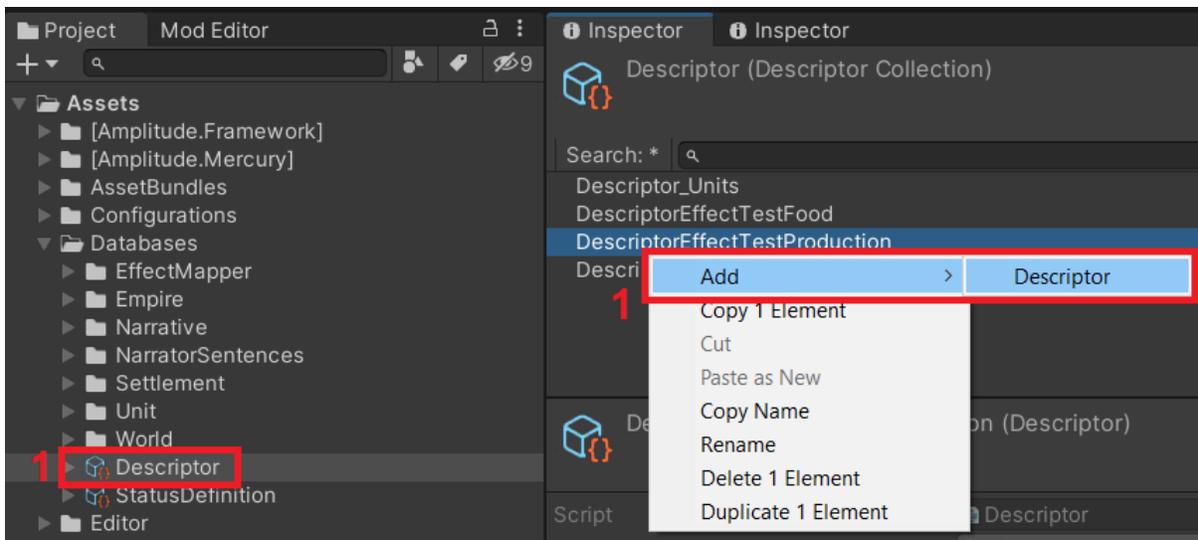
On the next turn the consequence event will be triggered (or not if the chance is not equal to 100).



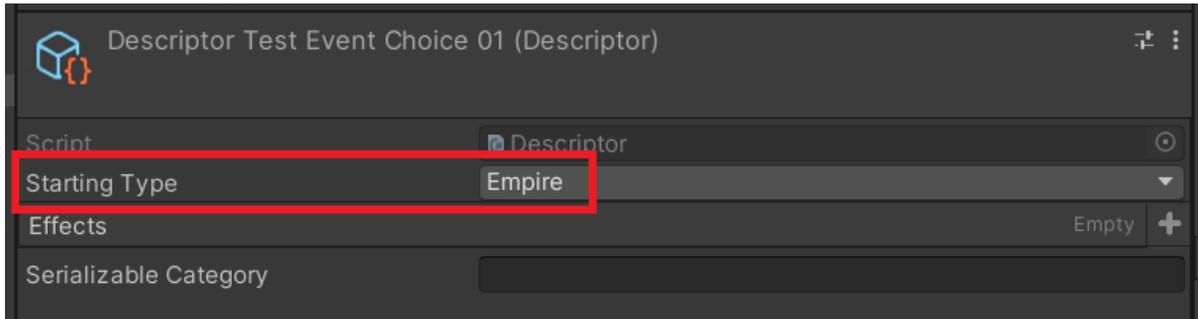
### 11.12.3 Narrative consequence tag

This way allows to trigger an event as a consequence with very different prerequisites and much later in the game. The main goal is to assign a hidden descriptor (tag) with a made choice and check it as a prerequisite.

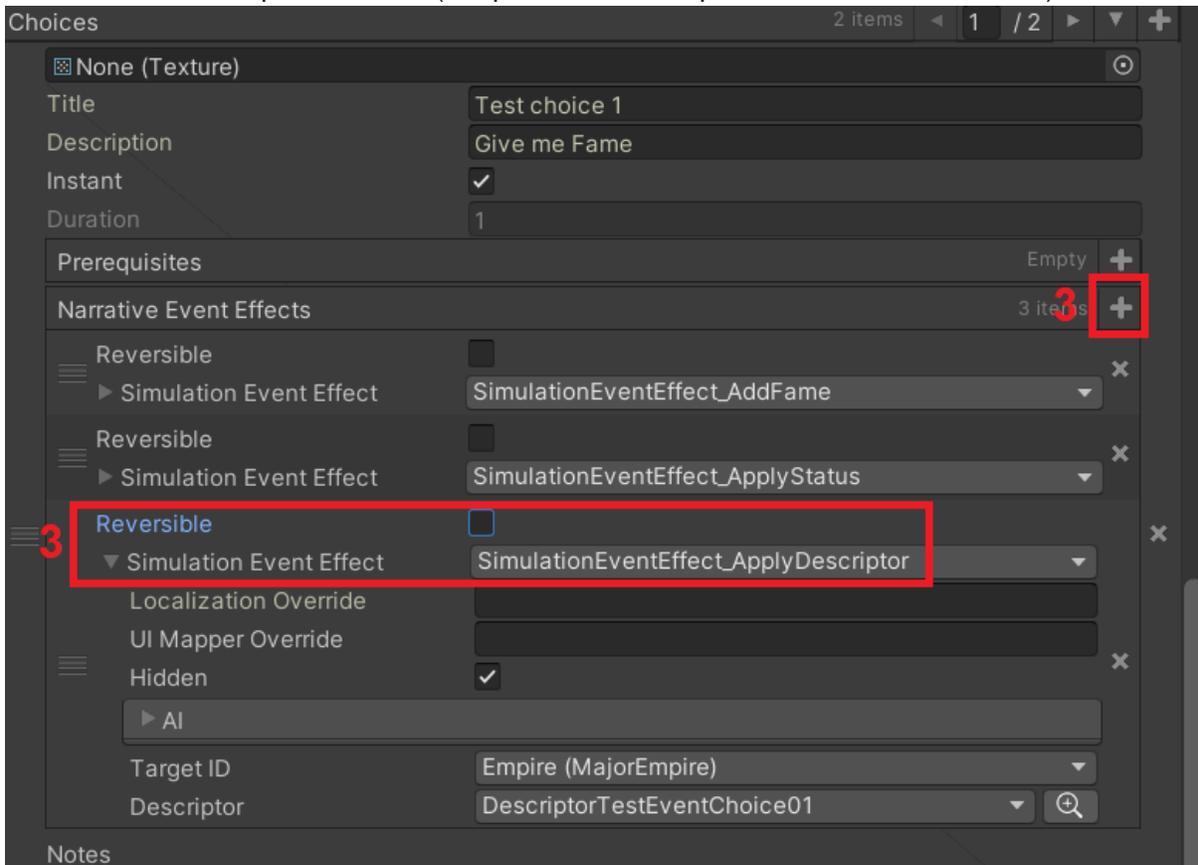
1. Create a new empty descriptor of type "Empire". The existing collection from city statuses can be used.



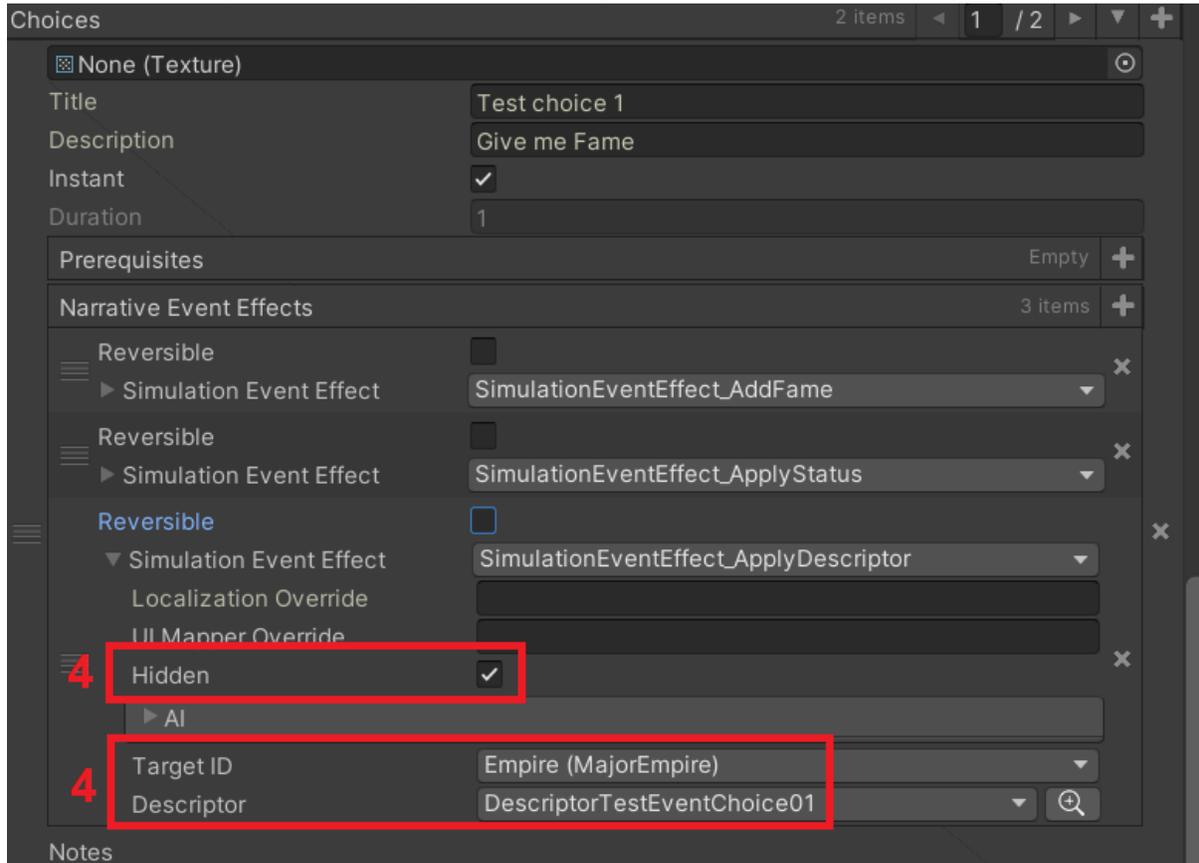
2. Rename the descriptor and set the “Starting Type” to “Empire”. Other fields will be empty.



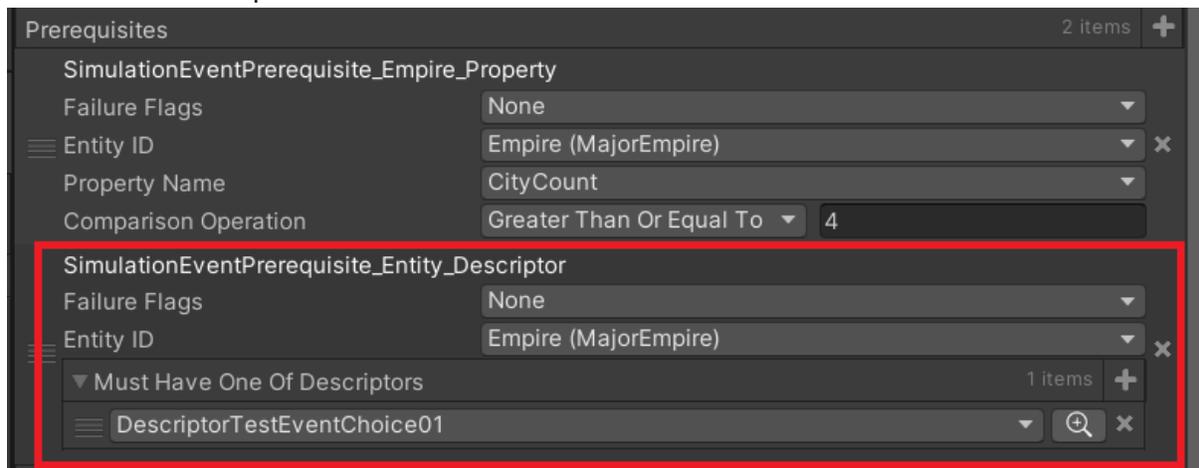
3. Get back to the initial event and add the new narrative event effect “SimulationEventEffect\_ApplyDescriptor” to one of the choices. It will trigger the consequence event (the previous consequence effect was removed).



4. Tick the “Hidden” box to hide the reward from the screen, set the “Target ID” to “Empire (MajorEmpire)” and assign the newly created descriptor.



5. Create a new consequence event and fill the event as needed. In the “Prerequisites” the simulation effect “...Entity\_Descriptor” must be added with a descriptor from the initial event’s choice.

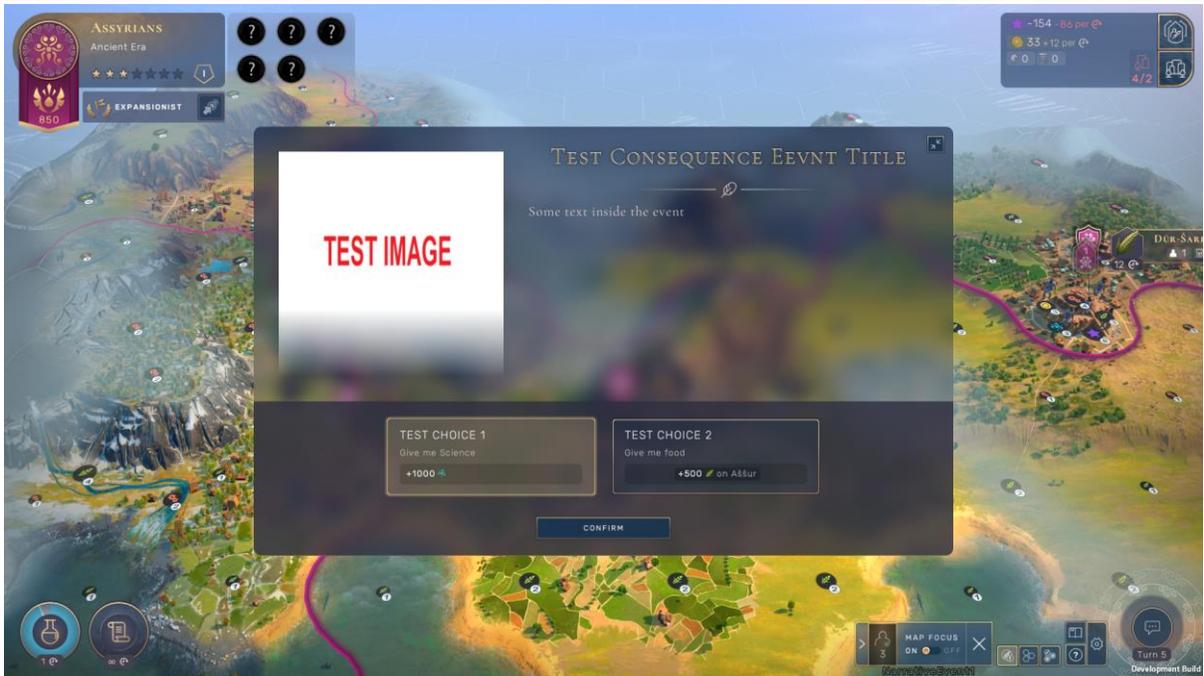


## 11.12.4 Testing

After all steps were done - click "Build and Run" to check the results. Start a new game, go to the needed era and complete the prerequisites (having 3 cities). Pick the event which has a narrative consequence and finish the turn.



Once the city amount reaches 4, the new event will be triggered.



## 12 Adding a new Civic

A civic is an in-game unlock of societal matters on which the player can position the Empire by making a choice between propositions, and thus modifying the Ideological Axis.



In the game civic consists of:

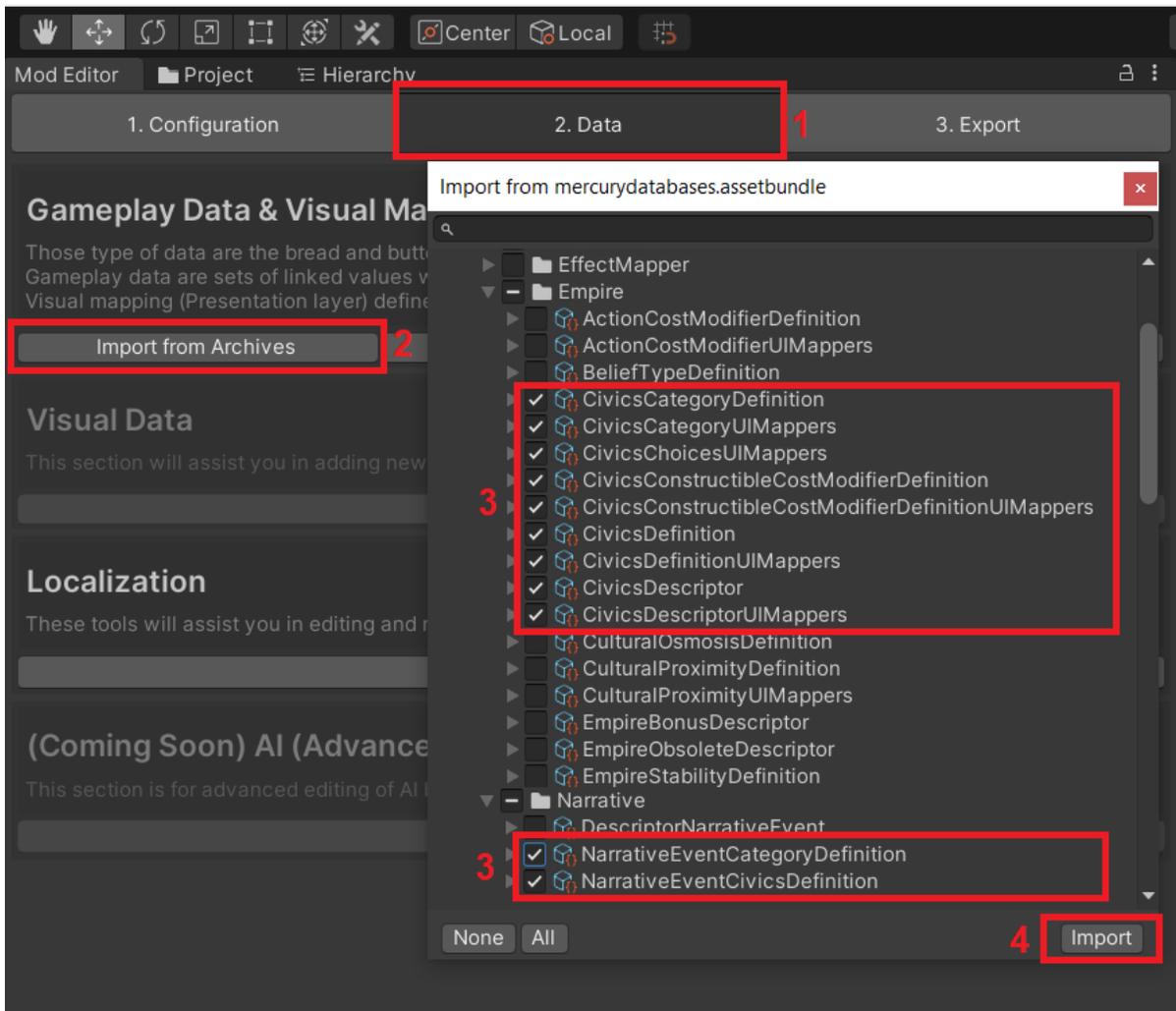
- a name;
- a description;
- an image;
- 2 decisions with ideological changes and gameplay effects;
- a cost;
- prerequisite requirements.

To unlock the Civic some prerequisites must be met and the narrative event triggered.

### 12.1 Setting up the environment

To modify or create a new civic, the related collections have to be exported:

1. Go to "2. Data" of the Mod Editor.
2. Select "Import from Archives".
3. Select the related civic collections (from "Empire" and "Narrative" databases).
4. Press "Import".



The following collections were exported:

1. The “CivicsCategoryDefinition” defines the civic’s category to which it is related.
2. The “CivicsCategoryUIMappers” defines the civic’s category name, description, etc.
3. The “CivicsChoicesUIMappers” defines the civic’s choices names, description, etc.
4. The “CivicsConstructibleCostModifierDefinition” defines constructible cost modifier effects assigned to the civic (e.g. “-20% on Attach Outpost cost”).
5. The “CivicsConstructibleCostModifierDefinitionUIMappers” defines names, descriptions, etc for “CivicsConstructibleCostModifierDefinition” objects.
6. The “CivicsDefinition” defines the civic’s settings.
7. The “CivicsDefinitionUIMappers” defines names, descriptions, locations on the civic screen, etc for the “CivicsDefinition” assets.
8. The “CivicsDescriptor” defines the civic’s non-constructible effects.
9. The “CivicsDescriptorUIMappers” defines names, descriptions, etc for the “CivicsDescriptor” assets.
10. The “NarrativeEventCivicsDefinition” defines narrative events to unlock civics.

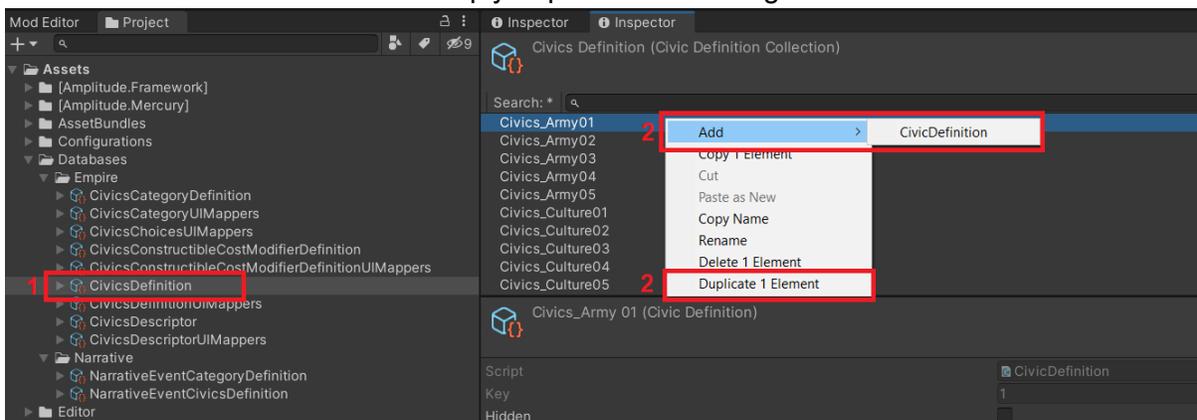
11. The “NarrativeEventCategoryDefinition” defines the narrative event category.

**Important!** It is not recommended to store unused objects within the mod project structure. Please either not export unused objects, or remove unnecessary objects from collections/collections after the mod is created.

## 12.2 Creating the new civic

To create a new civic, a new “CivicDefinition” asset must be created:

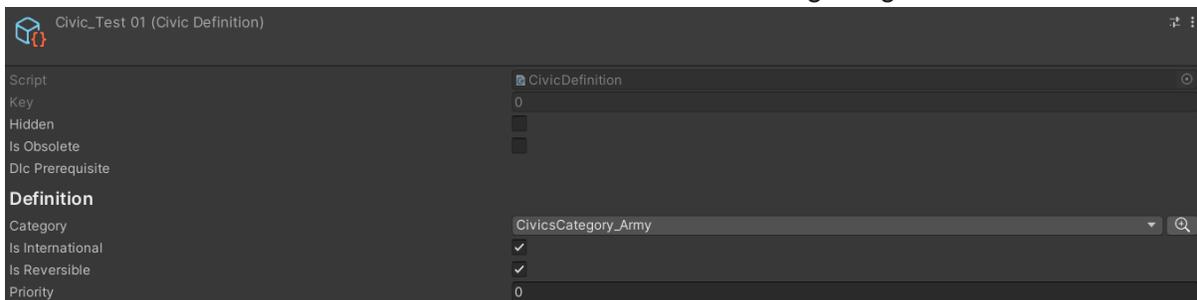
1. Select “CivicsDefinition” collection in the project structure.
2. Add a new element by right-clicking in the inspector space and selecting “Add->CivicsDefinition” or simply duplicate an existing one.



## 12.3 Setting up the civic definition

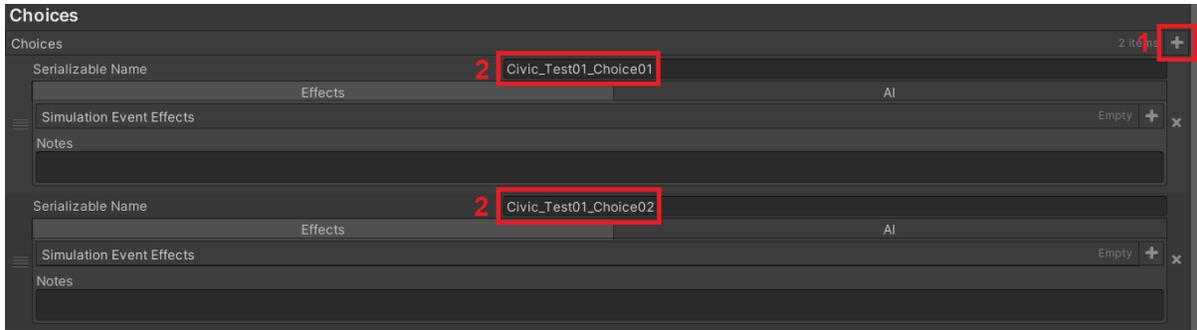
The “Definition” section defines:

1. The category for the new civic.
2. The “Is International” allows other civilizations to see the made choice and can lead to a special grievance for the influencing foreign empires.
3. The “Is Reversible” allows to reset the Civic during the game.



## 12.4 Setting up the empty choices

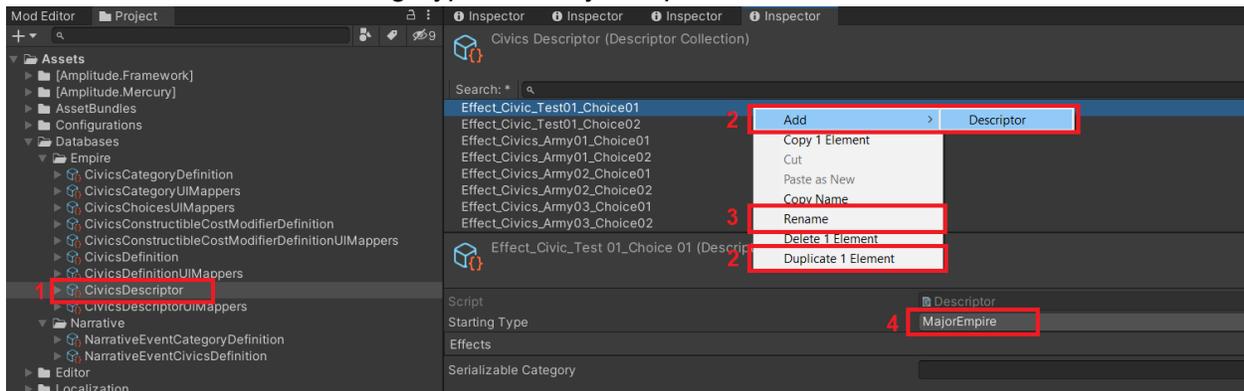
1. Add two empty choices to the new civic to select one of them during the game.
2. Name both choices in a simple way.



## 12.5 Setting up the civic descriptors

Civic descriptors usually apply non-constructible effects on “MajorEmpire”. Each civic definition must contain related descriptors even if the descriptors are empty and have no effect on the game:

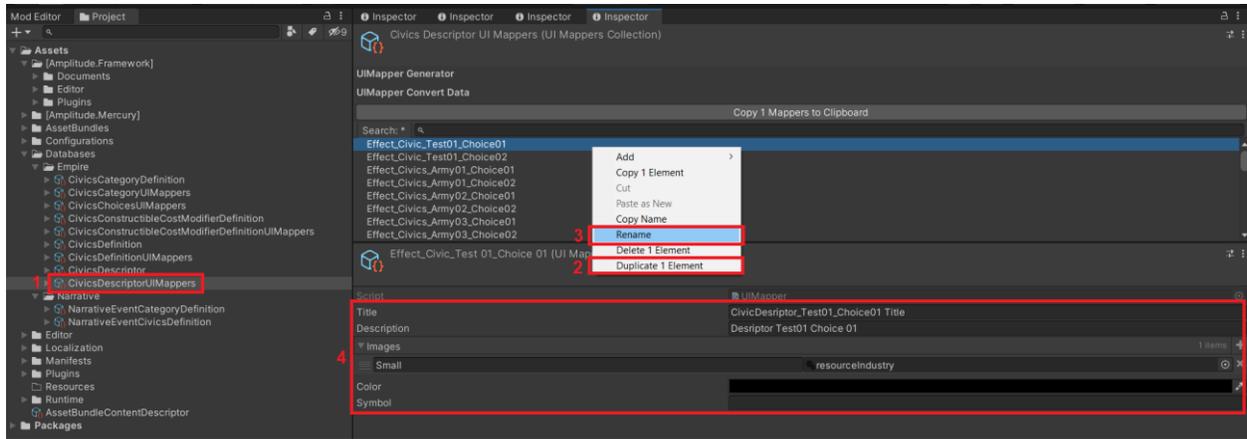
1. Select the “CivicsDescriptor” collection in the project structure.
2. Add 2 new elements by right-clicking in the Inspector space and selecting “Add-> Descriptor” or simply duplicate an existing one.
3. Rename both elements in the following manner: “Effect” + CivicDefinition Name “Choice01” (or Choice02).
4. Set the “Starting Type” as “MajorEmpire”.



## 12.6 Setting up the civic descriptor UI mapper

To display the civic descriptor correctly:

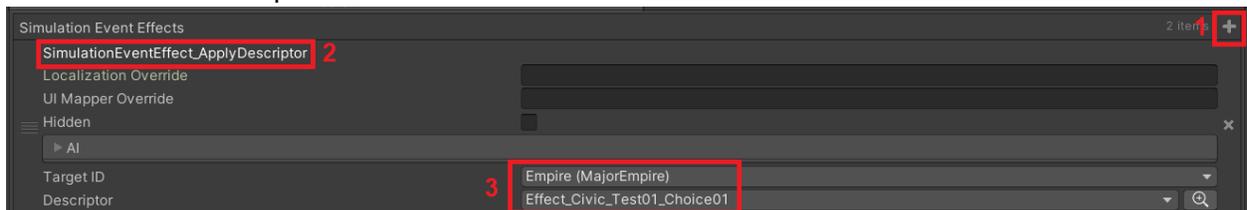
1. Select “CivicsDescriptorUIMappers” collection in the project structure.
2. Add 2 new elements by right-clicking in the Inspector space duplicating an existing one.
3. Give the same name as to “CivicsDescriptor” objects : “Effect” + CivicDefinition Name “Choice01” (or Choice02).
4. Fill the necessary fields for both newly created objects.



## 12.7 Fulfilling the empty choices

Add the newly created descriptors to each choice:

1. Add the “SimulationEventEffects”.
2. Pick the “SimulationEventEffects\_AppllyDescriptor” in the “Plugin Editor Types”.
3. Set the “TargetID” to “Empire (MajorEmpire)” and select the newly created descriptor for the choice.



Fulfill each choice with some in-game effects:

1. Add the “SimulationEventEffects”.
2. Pick the preferable effect in “Plugin Editor Types” (“ModifyIdeologicalAxis” for the example).
3. Set the “TargetID” to “Empire (MajorEmpire)”, pick Ideological Axis and set the delta.



**Hint!** Ideological Axis References:



Current example view for both choices:

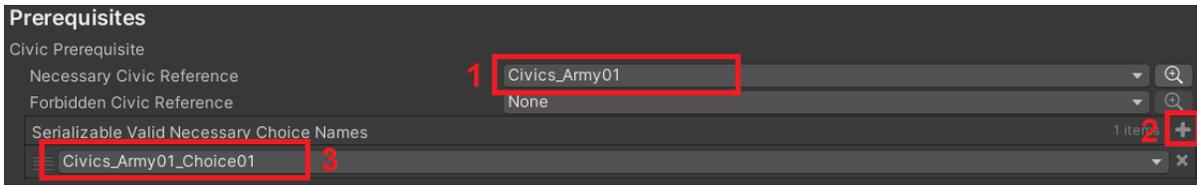
Serializable Name	Civic_Test01_Choice01
Effects	
AI	
Simulation Event Effects <span style="float: right;">2 Items +</span>	
SimulationEventEffect_ApplyDescriptor	
Localization Override	
UI Mapper Override	
Hidden	<input type="checkbox"/>
▶ AI	
Target ID	Empire (MajorEmpire)
Descriptor	Effect_Civic_Test01_Choice01
SimulationEventEffect_ModifyIdeologicalAxis	
Localization Override	
UI Mapper Override	
Hidden	<input type="checkbox"/>
▶ AI	
Target ID	Empire (MajorEmpire)
Ideological Axis Reference	IdeologicalAxis_Economic
Delta	8
Notes	
Test Choice 01	

Serializable Name	Civic_Test01_Choice02
Effects	
AI	
Simulation Event Effects <span style="float: right;">2 Items +</span>	
SimulationEventEffect_ApplyDescriptor	
Localization Override	
UI Mapper Override	
Hidden	<input type="checkbox"/>
▶ AI	
Target ID	Empire (MajorEmpire)
Descriptor	Effect_Civic_Test01_Choice02
SimulationEventEffect_ModifyIdeologicalAxis	
Localization Override	
UI Mapper Override	
Hidden	<input type="checkbox"/>
▶ AI	
Target ID	Empire (MajorEmpire)
Ideological Axis Reference	IdeologicalAxis_Economic
Delta	-8
Notes	
Test Choice 02	

## 12.8 Setting up the “Prerequisites” section

The “Prerequisites” area defines if any other civic choice must be made before opening the current one or if the civic is blocked by selecting the other one.

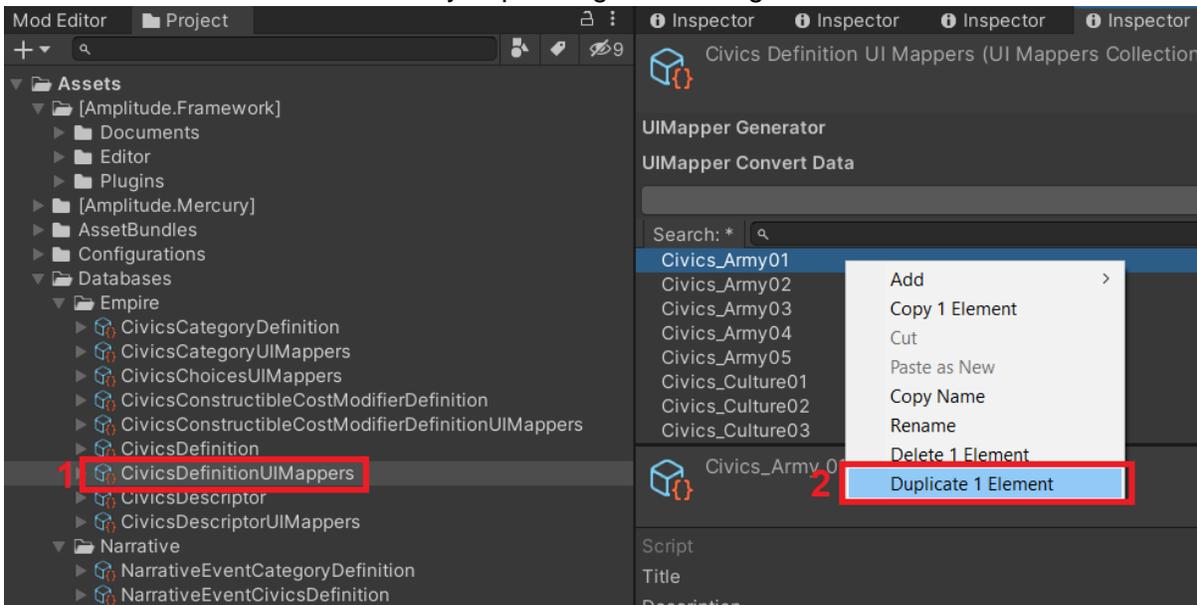
1. Select the referenced civic.
2. Click on the “Serializable Valid Necessary Choice Name”.
3. Select the necessary choice from other civic.



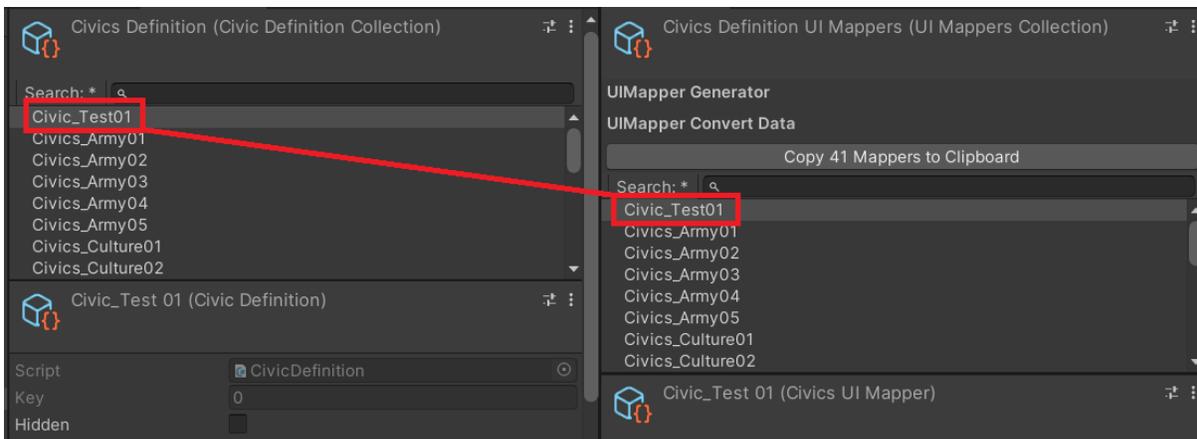
## 12.9 Mapping the new civic to the Civics screen

To display the new civic on the Civics screen correctly and assign the correct naming/description/location, the new object must be created in the “CivicsDefinitionUIMappers” collection.

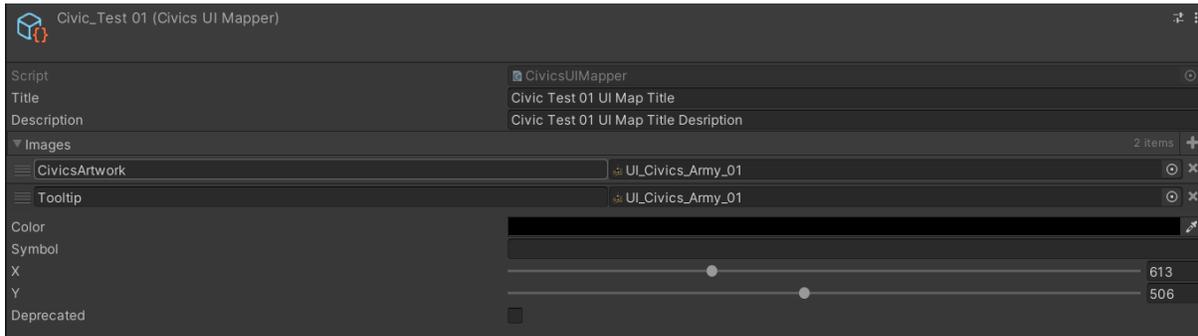
1. Go to the “CivicsDefinitionUIMappers” collection.
2. Add a new element by duplicating the existing one.



3. Rename the newly created object to the same name the “CivicsDefinition” object has. Names from 2 collections MUST match.



- Fill the corresponding fields with pictures, name, description and the location on the Civics Screen.



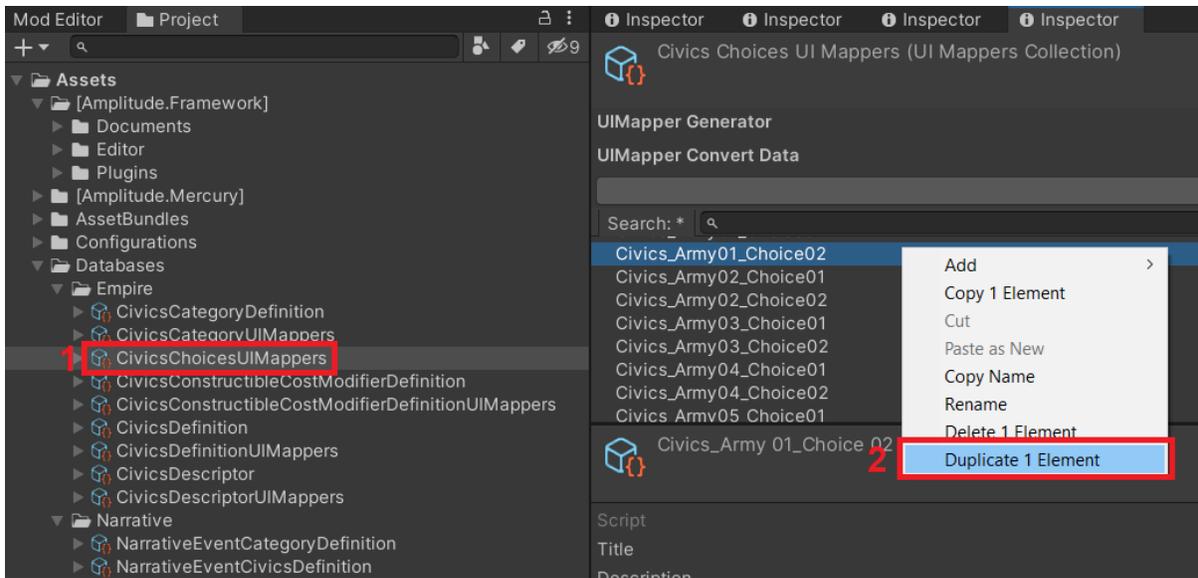
**Hint!** The X axis goes from left to right, the Y axis goes from top to bottom.



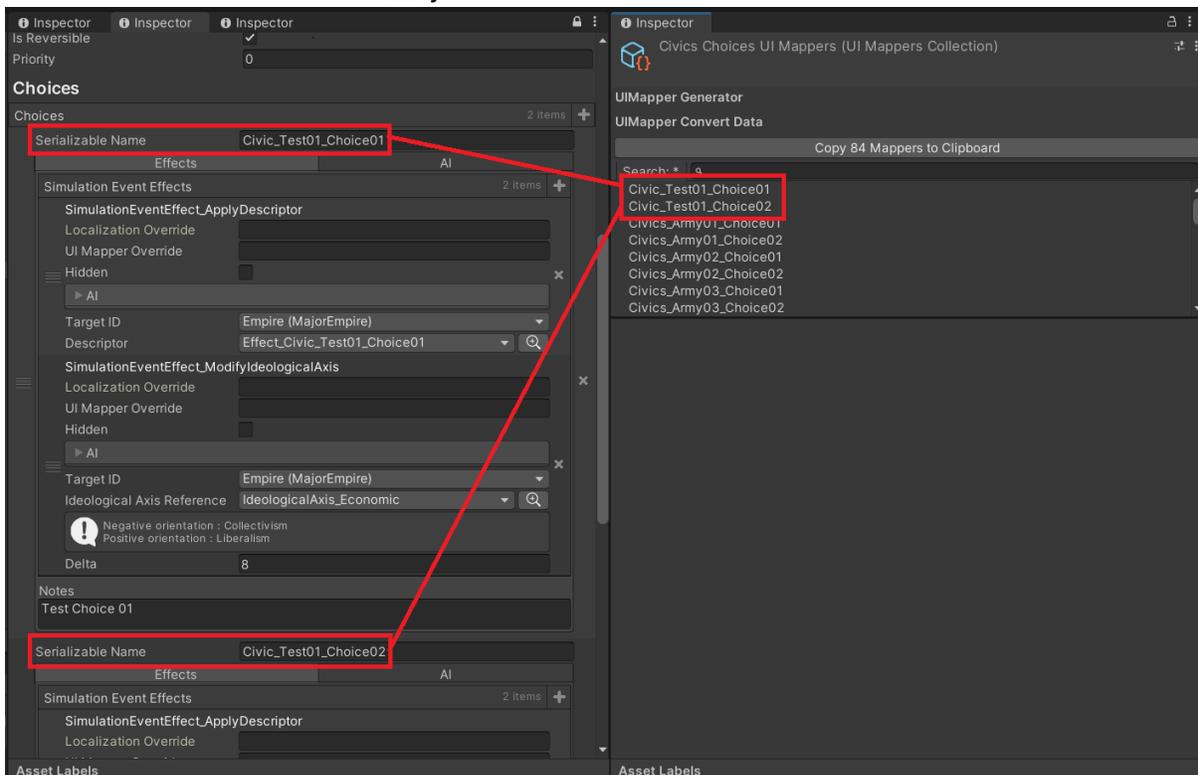
## 12.10 Setting the new civic choices UI mappers

To display the new civic choices correctly and assign the correct naming/description/location, the new object must be created in the “CivicsChoicesUIMappers” collection.

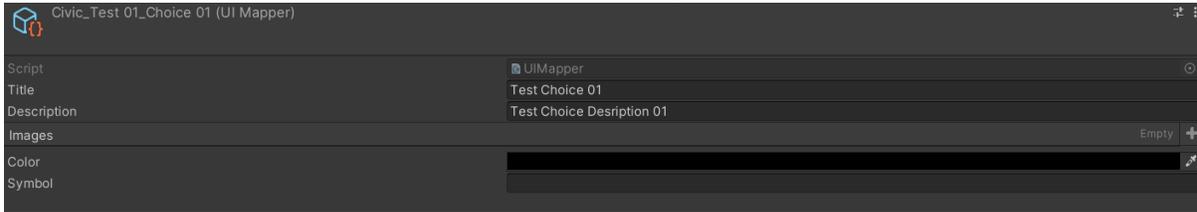
- Go to the “CivicsChoicesUIMappers” collection.
- Add 2 new elements (1 per each new civic choice) by duplicating the existing one.



3. Rename the newly created objects to the same names the choices have in the "CivicsDefinition" object. Names from 2 collections MUST match.



#### 4. Fill the corresponding fields for both objects.

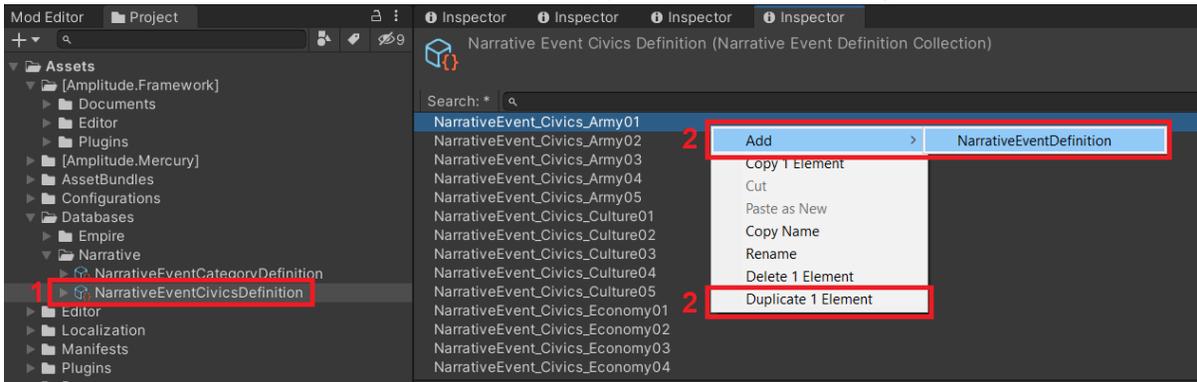


## 12.11 Creating a new civic narrative event

Unlocking the civics in-game works the same way as triggering narrative events by fulfilling the civic prerequisites. Each existing civic is connected with the corresponding narrative event of a special category and becomes available after the event was triggered.

To create a new civic narrative event, a new object must be created:

1. Select “NarrativeEventCivicsDefinition” collection in the project structure.
2. Add a new element by right-clicking in the inspector space and selecting “Add-> NarrativeEventDefinition” or simply duplicate an existing one.

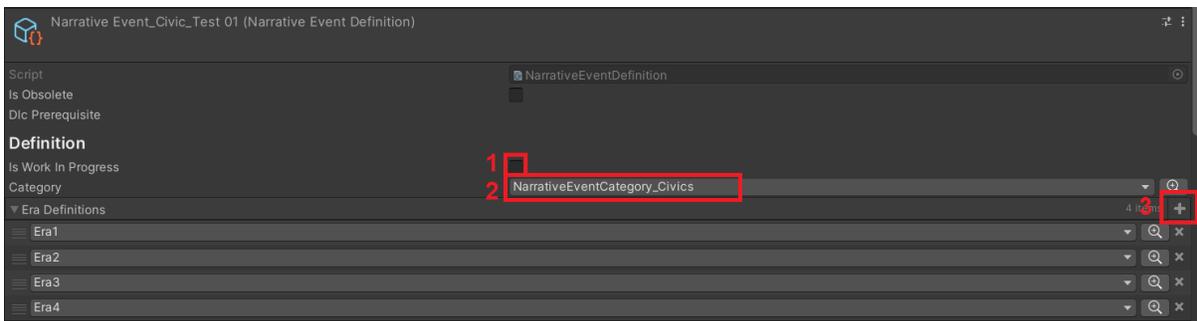


## 12.12 Setting up the “Definition” section

In the “Definition” area select:

1. Uncheck the “Is Work In Progress” box to make it available during the game.
2. The event category is responsible for events priority and dead zones. The default category for civics is the “NarrativeEventCategory\_Civics”.
3. Add the “Era Definitions” to show the civic event in different eras.

**Note!** Era0 cannot be selected.

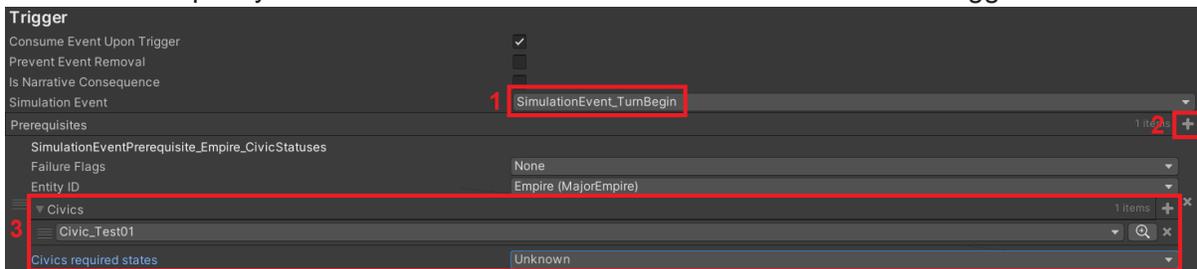


## 12.13 Setting up the “Trigger” section

To unlock the civic within a game the specific triggers must be configured in the narrative event.

Add a prerequisite to check whether the newly created civic is unlocked or not:

1. Select the preferable “Simulation Event”. It is highly recommended to use “SimulationEvent\_TurnBegin” as a safe choice.
2. Add the prerequisite object “SimulationEventPrerequisite\_Empire\_CivicStatuses” which checks if the newly created civic is unlocked or not.
3. Specify the civic which should be checked before the event triggers.



Add another prerequisite to check whether the first civic and the civic “Army 1” were selected:

1. Add the prerequisite object “SimulationEventPrerequisite\_Entity\_Descriptors” which checks whether the first civic was selected.
2. Add 2 new descriptors.
3. Select the “Tag\_MajorEmpire\_EventCivicsGovernment01\_Justice01” and the “Tag\_MajorEmpire\_EventCivicsArmy01”.

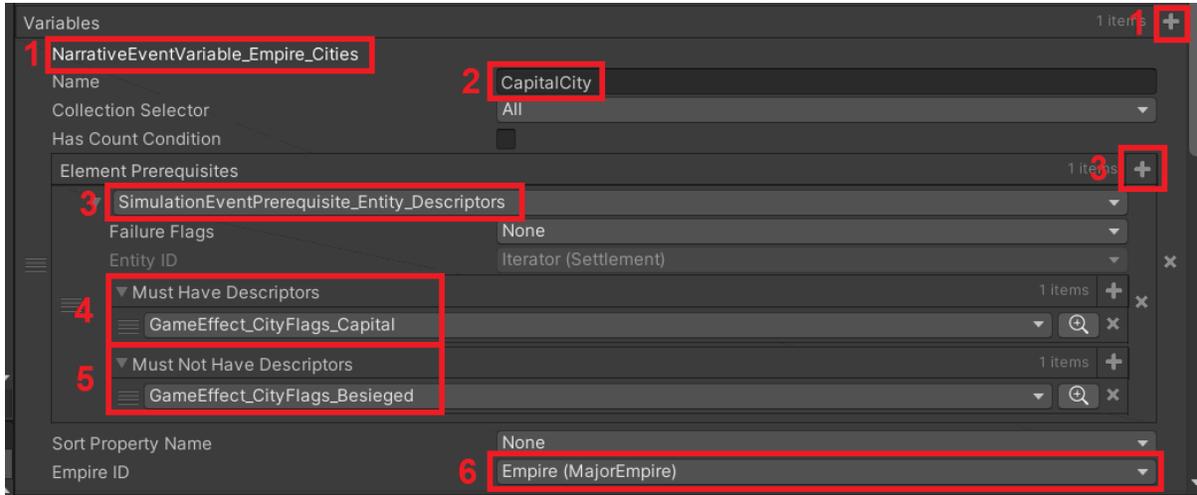


## 12.14 Setting up the “Variables” section

Within variables it is possible to check the assets statuses and if assets meet specific conditions. Let’s check whether the city is the capital of the empire and whether it is besieged or not.

1. Add a new variable of type “NarrativeEventVariable\_Empire\_Cities”.
2. Rename the variable.
3. Add a prerequisite element of type “SimulationEventPrerequisite\_Entity\_Descriptor” to check whether the city is the capital and is not besieged.

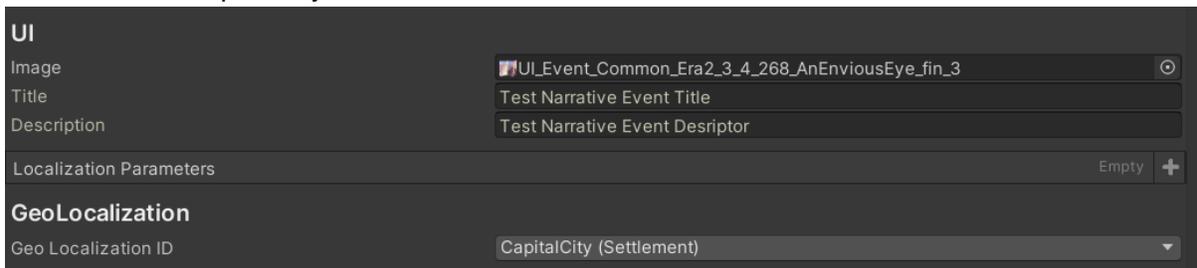
4. Assign a related descriptor “GameEffect\_CityFlags\_Capital” to the “Must Have” field.
5. Assign a related descriptor “GameEffect\_CityFlags\_Besieged” to the “Must Not Have” field.
6. Specify the “Empire ID” as “Empire (MajorEmpire)”.



## 12.15 Setting up the “UI” and “GeoLocalization” sections

In the UI area it is possible to set the appearing image of the event, its text and description.

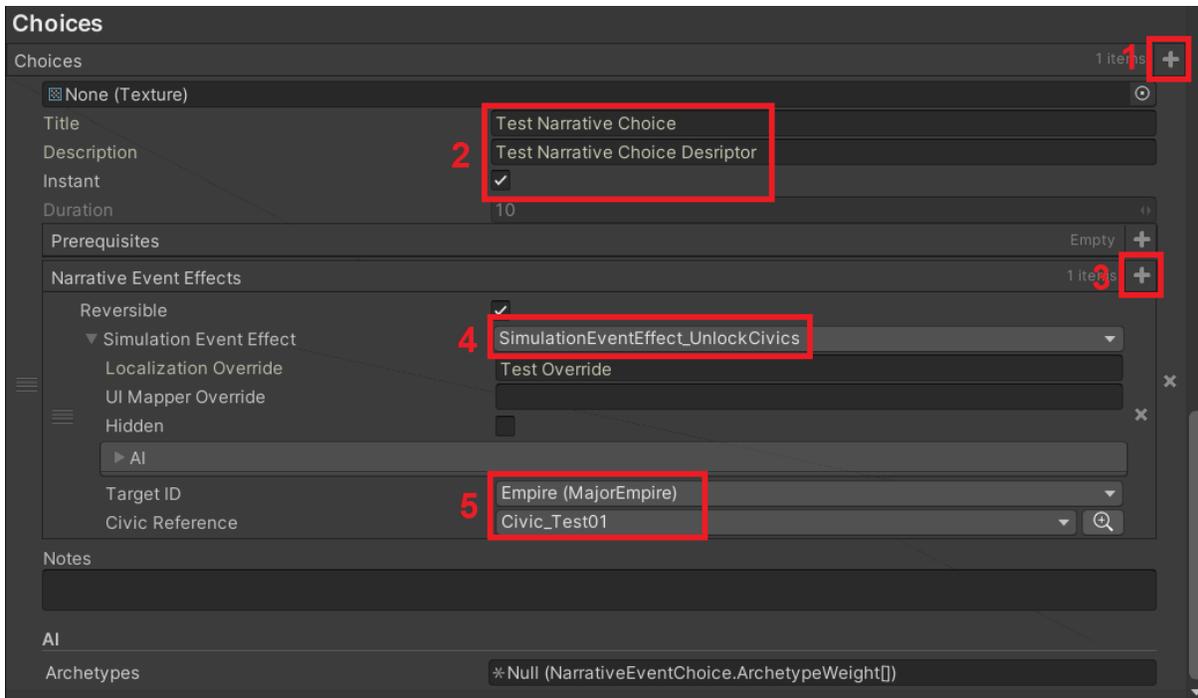
The “GeoLocalization” area identifies where the event appears. The current example event will occur in the capital city from the variable created before.



## 12.16 Setting up the choice

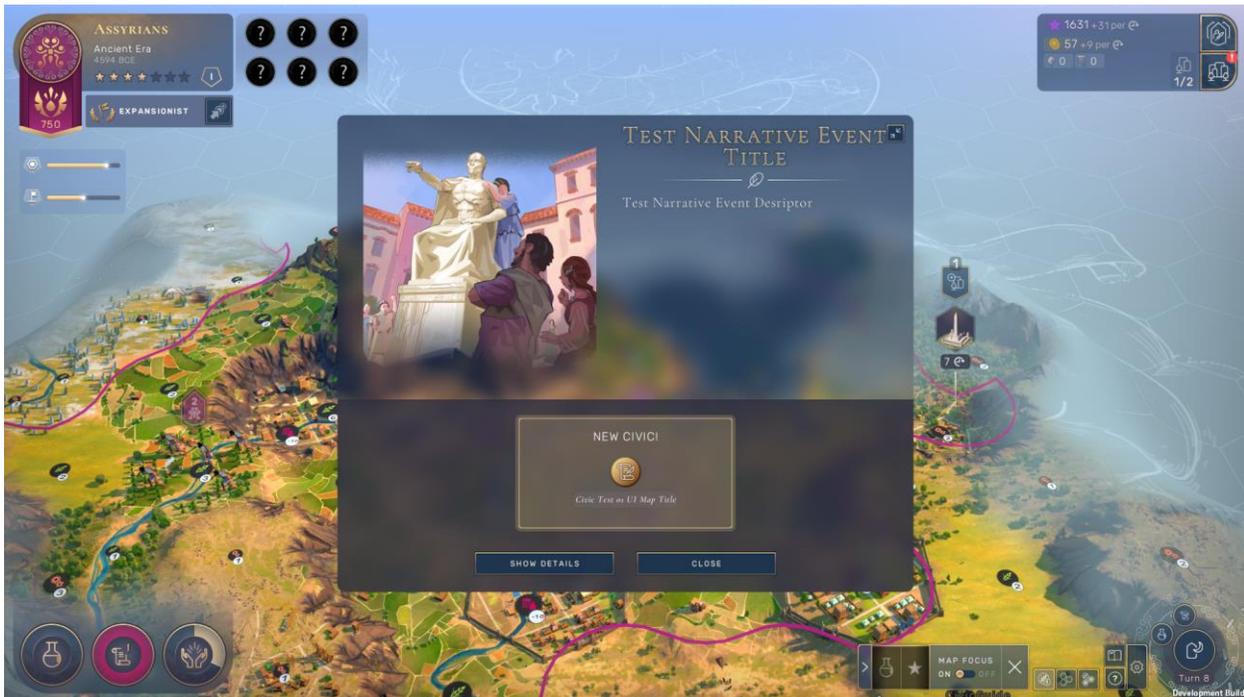
Events that unlock civics usually have only 1 choice to select. To create a choice:

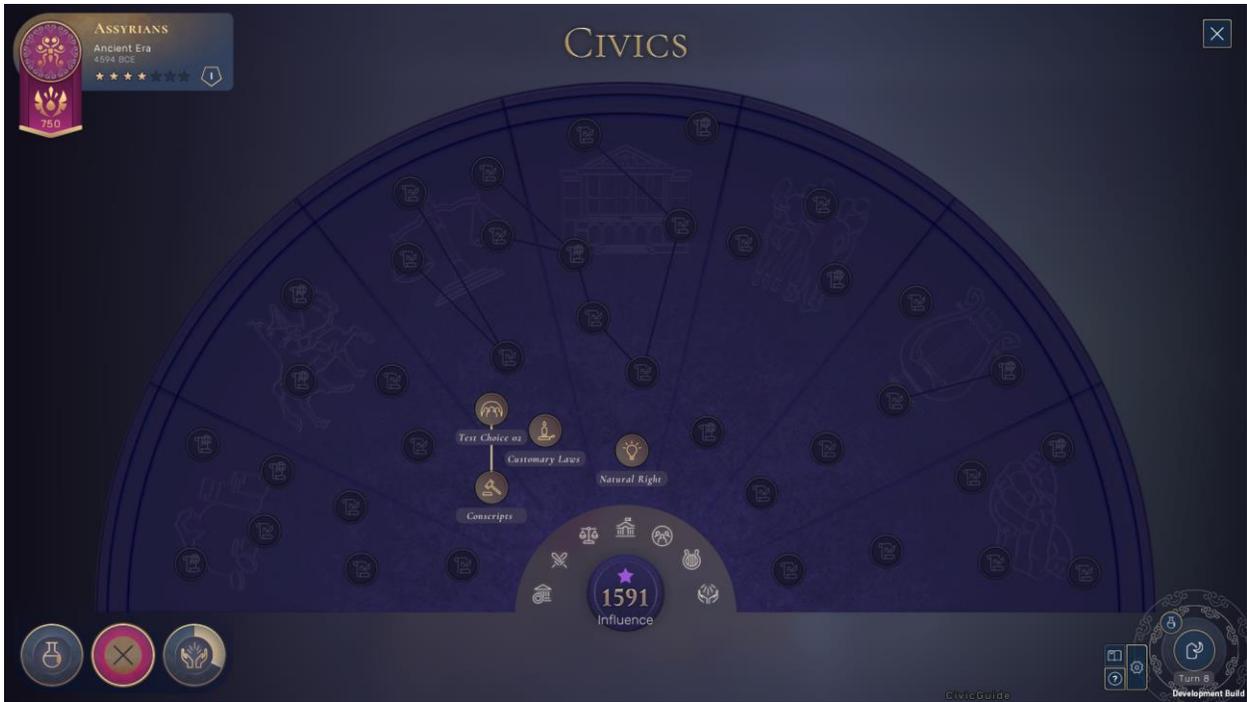
1. Add the new choice.
2. Fill the related fields and make sure the civic narrative event is “Instant”.
3. Add the “Narrative Event Effect”.
4. Pick the “SimulationEventEffect\_UnlockCivics” which will unlock civic during the in-game event.
5. Select the “TargetID” and the “Civic Reference”. The “Civic Reference” specifies which civic will be unlocked with the civic narrative event (newly created civic in the current example).



## 12.17 Testing

After all steps were done - click "Build and Run" to check the results. Start a new game, go to the needed era and complete the prerequisites to trigger the event (have selected civic for Army01 and Justice/Government01).



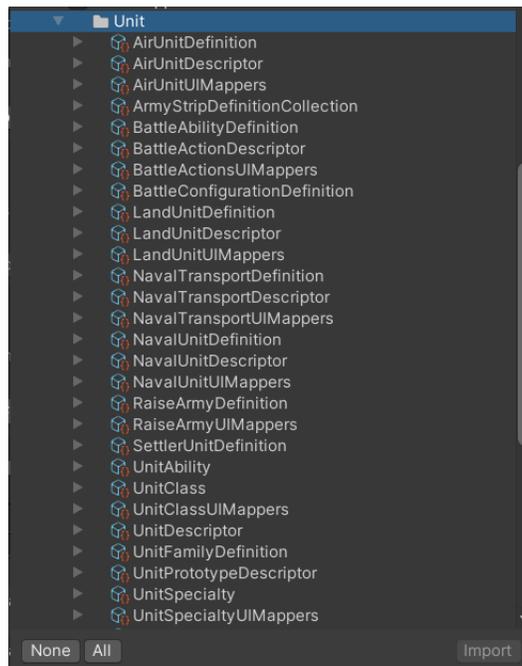


# 13 Balancing Battle

Battle is an important part of the game, with the modding tool it's possible to modify different aspects of it (from simple combat strength, to complicated battle actions).



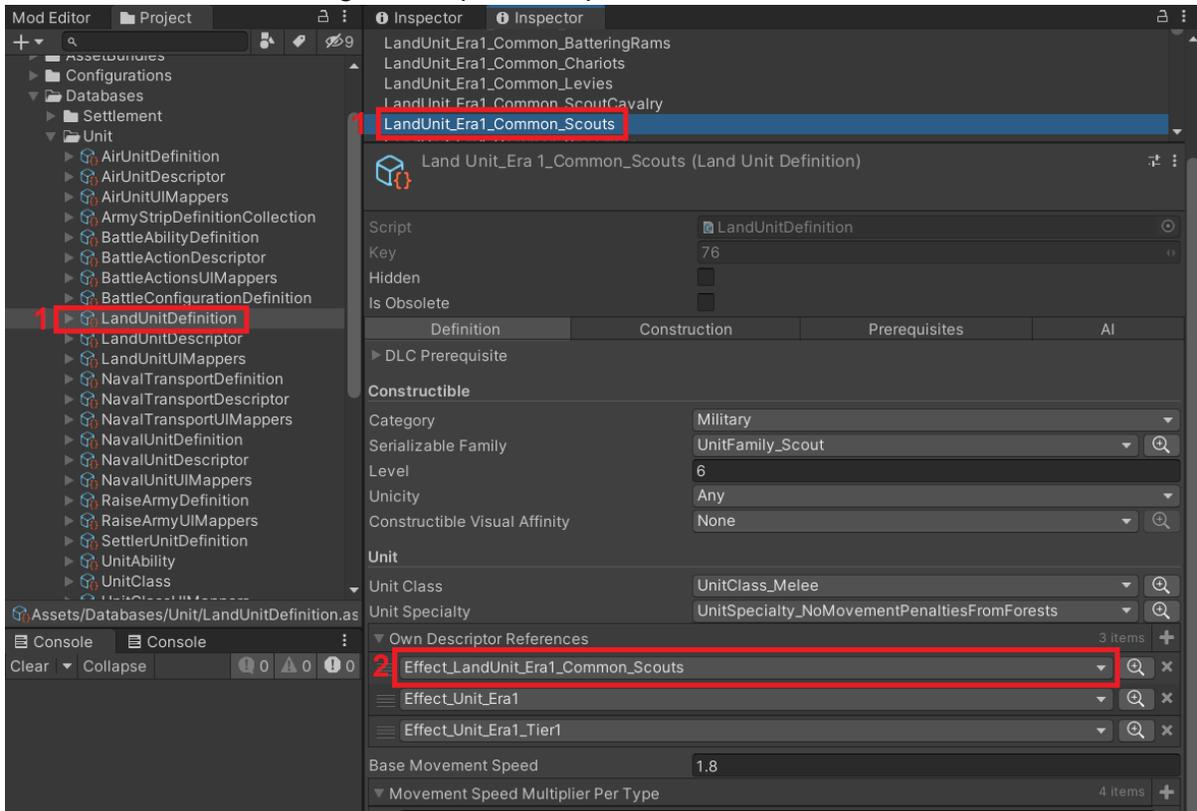
All the necessary collections related to units or battle are stored in the "Unit" database.



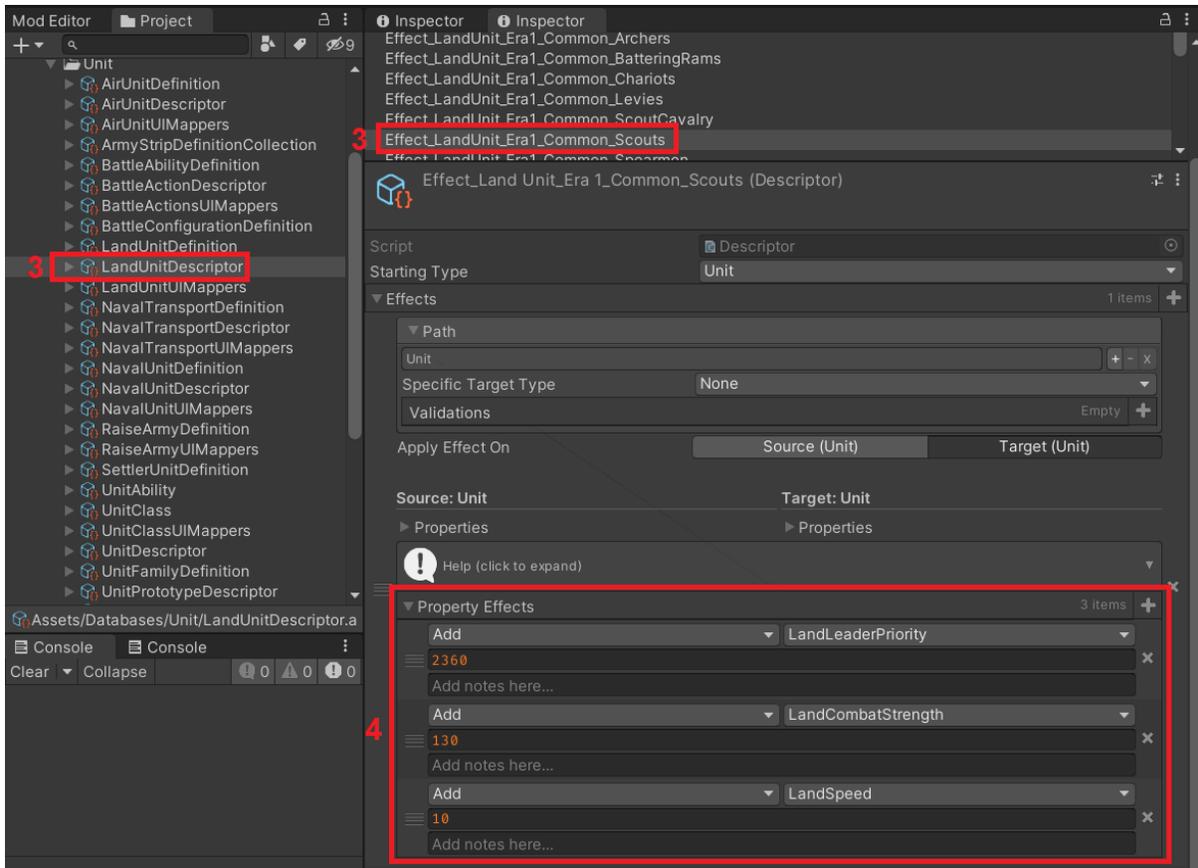
## 13.1 Changing unit's basic stats

Changing some basic unit stats is pretty simple:

1. Find the necessary unit in the database definition collection.
2. Find the assigned unique descriptor to the unit under the "Definition" tab.

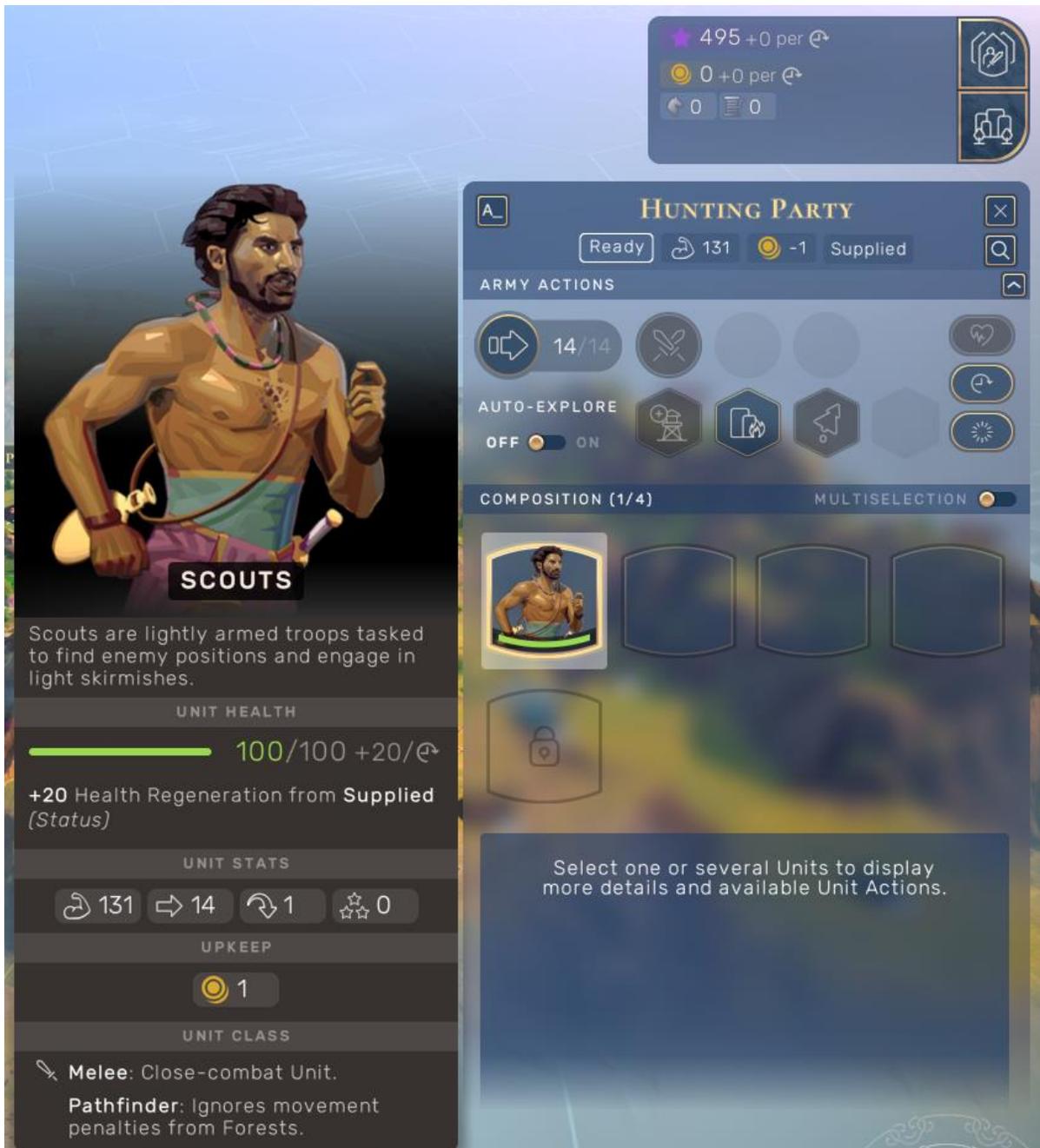


3. Search for this descriptor in the corresponding collection.
4. Modify the combat strength, priority or add the necessary effect (speed for example).



## 13.2 Testing

Let's check the result of the change - click "Build and Run" and start a new game. Once the Ancient era is reached, the modified scouts will appear.



### 13.3 The unit abilities

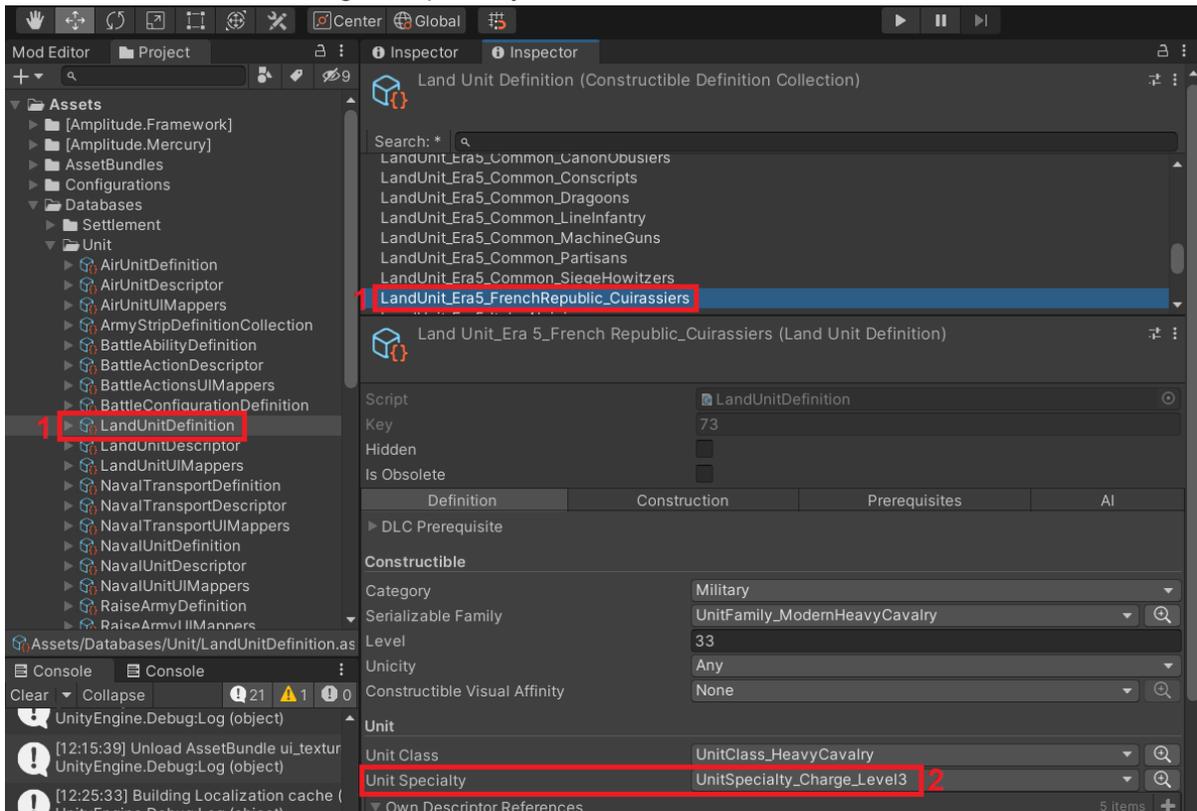
Unit ability gives bonuses to the unit in combat or while moving, and ability itself. Unit ability consists of the following sections which can be modified:

- The “Tag As Ability” defines if the tag works as an ability.
- The “Movement Ability” defines if the ability is related to movement on water or ground.
- The “Pathfinding Flags” defines if the ability allows or ignores some in-game zones/situations.

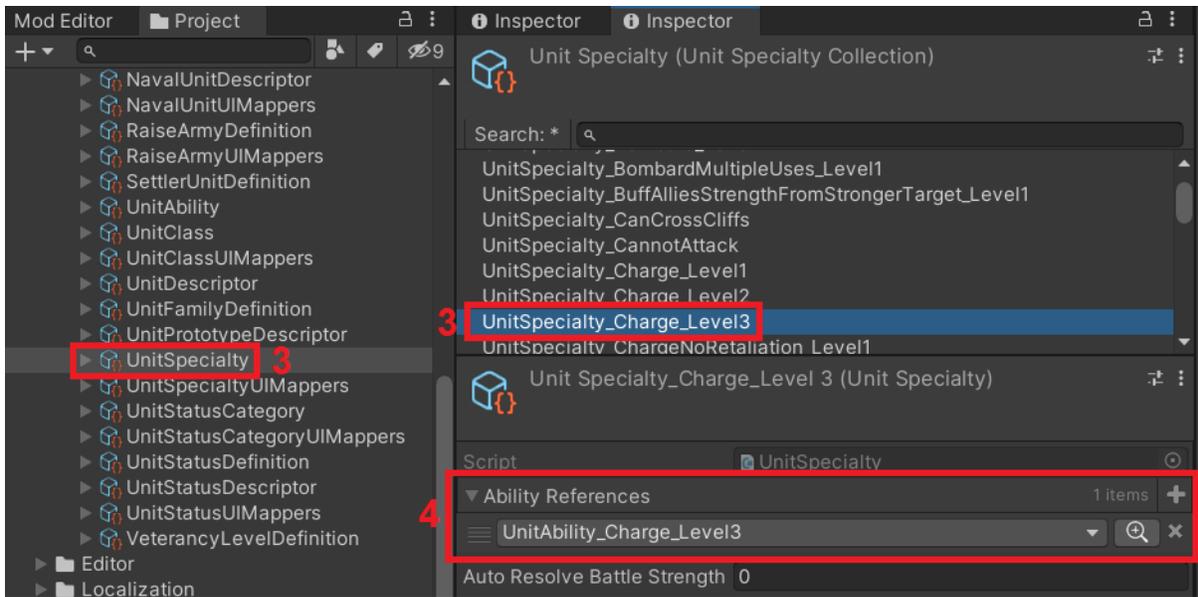
- The “Descriptor Reference” gives some effects/bonuses (like specialty level) on the unit/target.
- The “Battle Ability Reference” defines how the unit behaves during the battle and assigns the ability formula.

To find the unit ability:

1. Find the necessary unit in the corresponding collection.
2. Find the assigned specialty to the unit under the “Definition” tab.

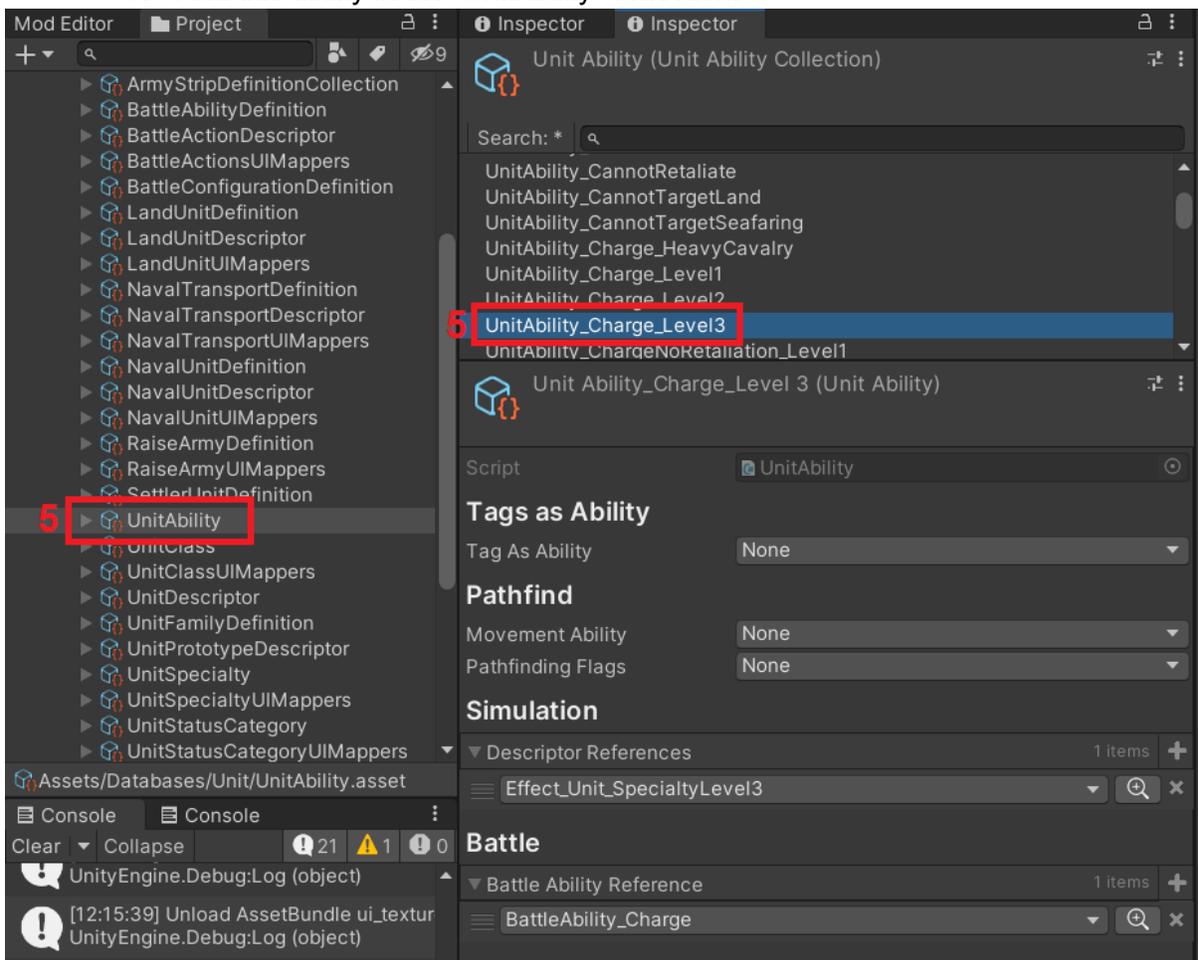


3. Find this specialty in the “UnitSpecialty” collection.
4. Find the ability under the “Ability References” section.



**Note!** Despite the fact that specialties mostly work as placeholders for abilities and only give a bonus for auto resolve, they are displayed in the game under the unit’s description.

5. Find this ability in the “UnitAbility” collection.

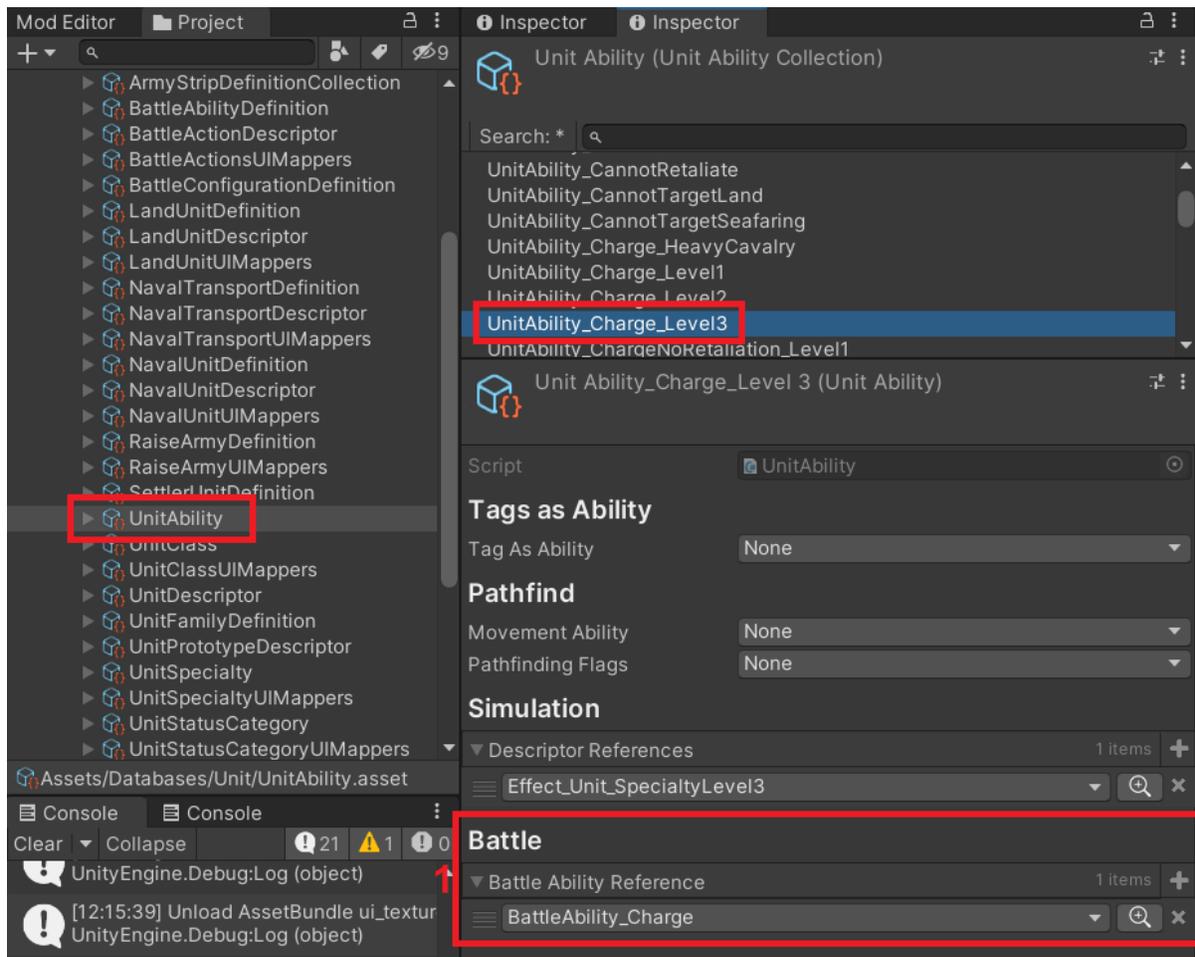


## 13.4 The battle abilities

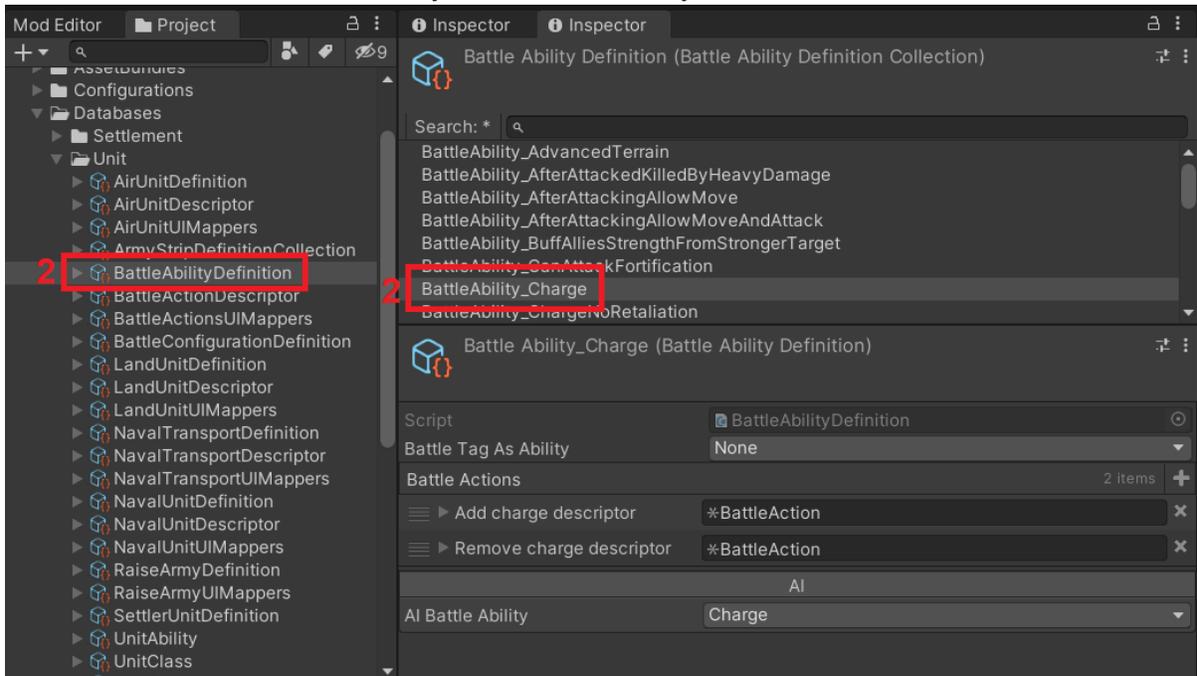
Battle abilities are assigned to unit abilities and define how the unit behaves during the battle and the assigned ability formula. Some battle abilities are common (“BattleAbility\_Common”) for units in the game, while some are unique.

To find the battle ability of the unit:

1. Find the battle ability under the “Battle Ability Reference” section in the unit ability asset.

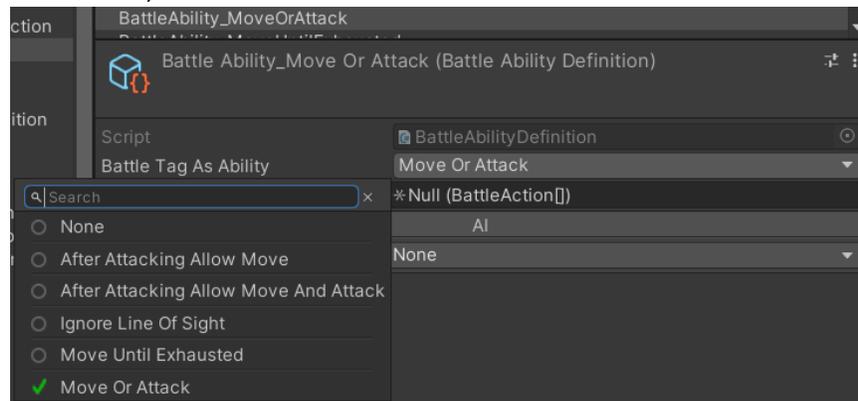


## 2. Find this battle ability in the “BattleAbility” collection.

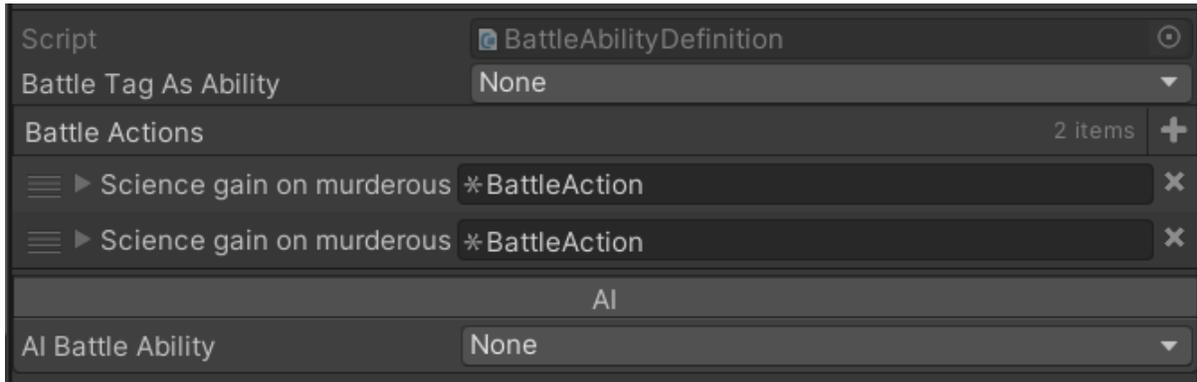


There are 2 main types of battle abilities:

- Based on battle tags. If the “Battle Tag As Ability” is selected, the “Battle Action” section is empty. This type is not modifiable from the modding tool (actions are hardcoded).



- Based on battle actions. The “Battle Tag As Ability” is set to “None” and the “Battle Action” section is filled.



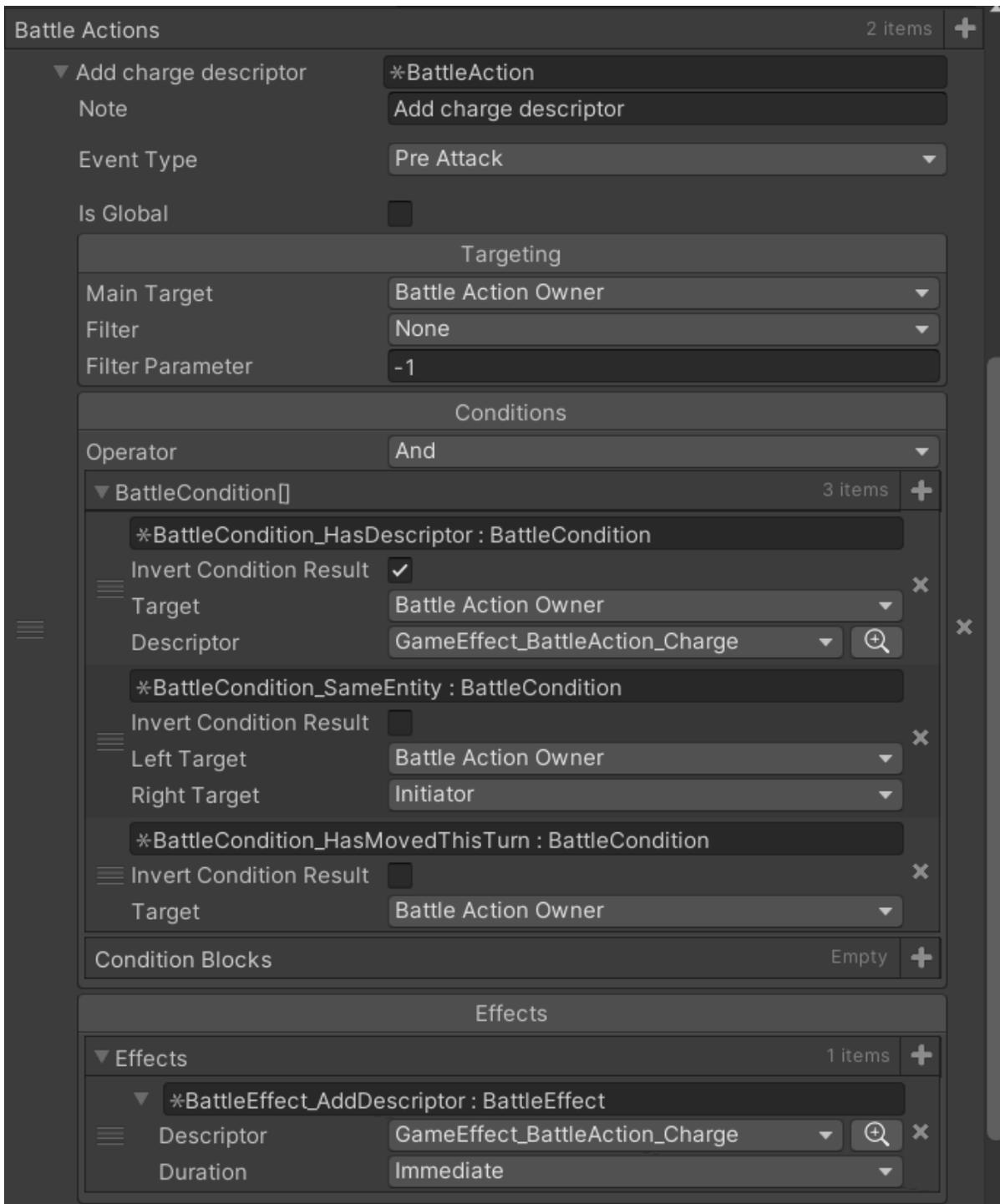
The “Battle Action” section is where all the checks and assignments are done. The main approach on this is to assign some descriptor and then remove it.

Explanation on some fields:

- The “Note” is used as a name or a note to the battle action.
- The “Event Type” is a predefined field which is used as a flag in the battle flow.
- The “Is Global” action is owned by a unit but applied to everyone in the battle.
- The “Main Target” defines an entity on which action/effect is applied:
  - The “Battle Action Owner” - unit that has been given the battle action.
  - The “Initiator” - unit that has triggered the event that caused battle action to fire.
  - The “Target” - target of the battle action.

**Note!** Targeting is a very tricky part. If the bonus didn’t work correctly, maybe the target was selected incorrectly.

- The “Filter” isn’t used in the game right now.
- The “Operator” defines if all (AND) or only one (OR) condition must be fulfilled.
- The “Condition Block” used for a more detailed approach on the battle conditions and used to fire the battle action only for some specific assets.
- The “Invert Condition Result” is used to prevent effects from sticking on the unit.

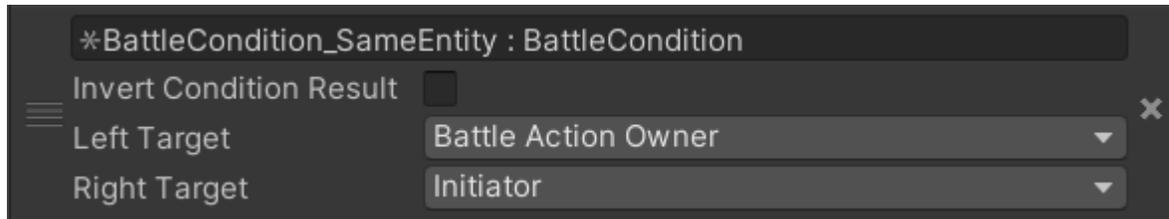


Explanation of the presented conditions:

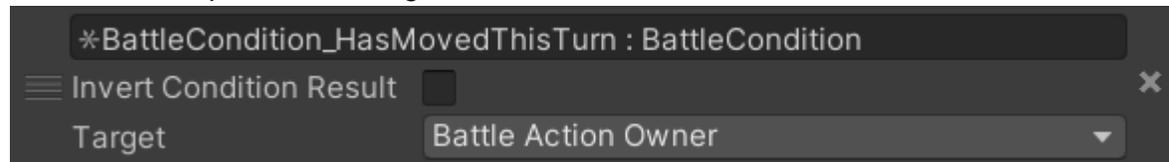
- The first one checks whether the target has the assigned game effect descriptor.



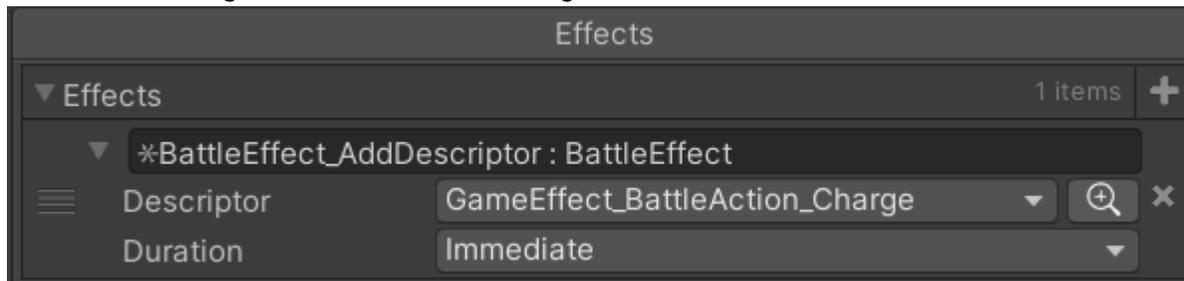
- The second one checks whether the targeted unit (this unit will receive the bonus) is the one who started the action. This is very common and widely used across all actions.



- The third one checks if the unit has already moved this turn. This condition is unique for the charge.



The “Effects” section assigns the game effect to the target and gives the bonus strength for the unit for charge. It also defines how long the effect will last.

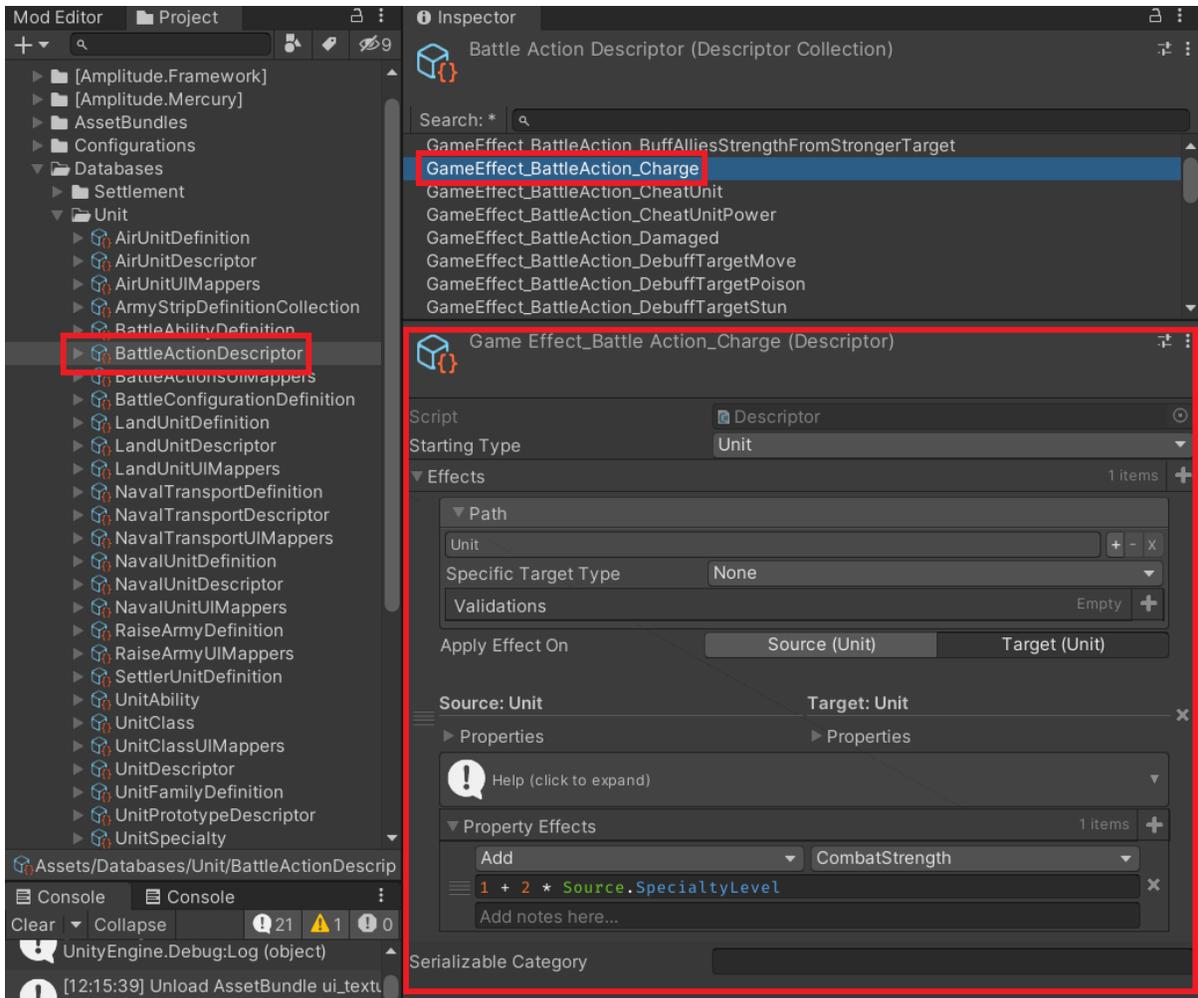


## 13.5 The game effect

The game effect is a simple descriptor stored in the separate collection that gives a bonus to a unit. It has the same structure as other descriptors.

To find the game effect:

1. Find the effect in the “Effects” section of the battle ability.
2. Find this descriptor in the “BattleActionDescriptor” collection.

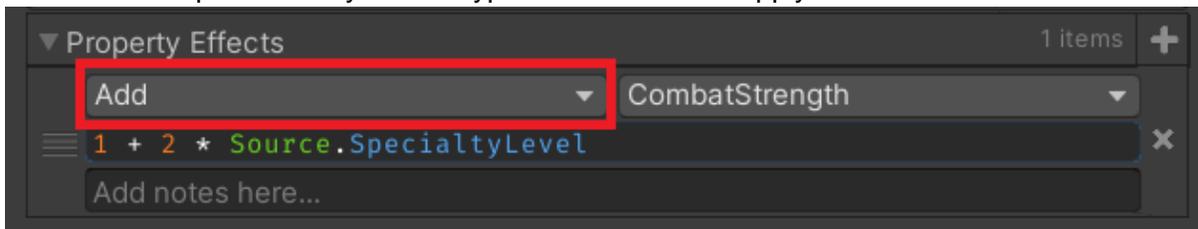


As it's shown, in the battle action descriptor the actual ability calculations are stored.

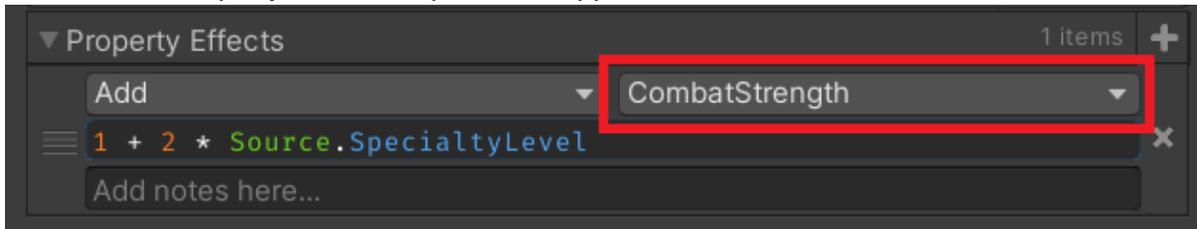
## 13.6 Property effects calculations

Each descriptor has the same property effect section which consists of:

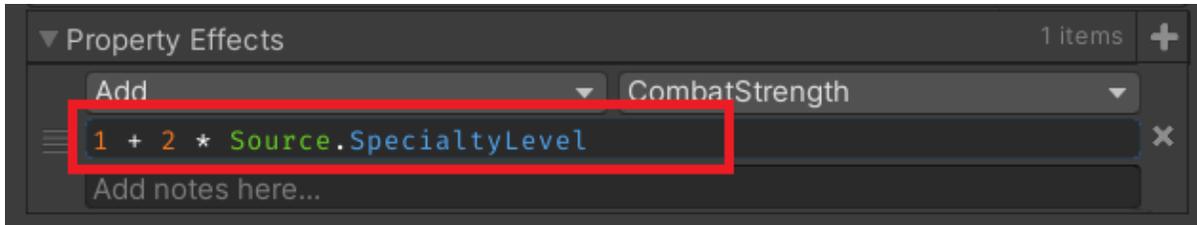
1. Operator - says which type of calculation to apply.



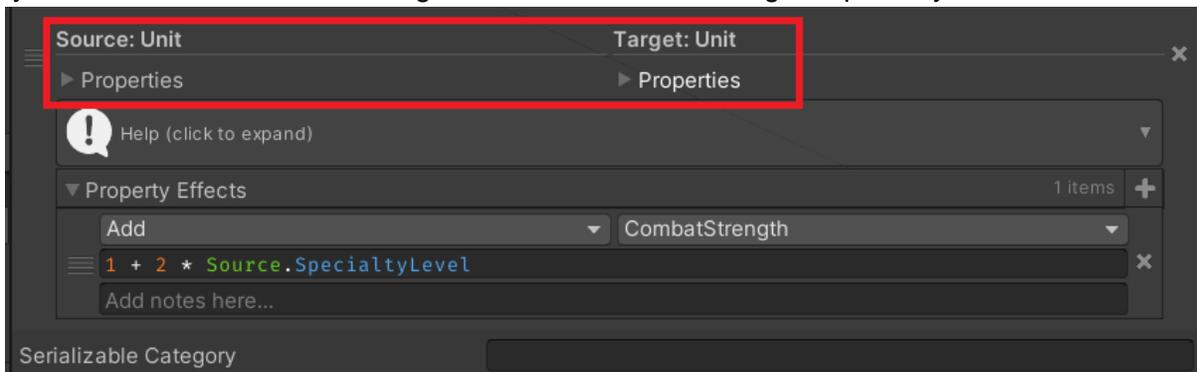
2. Property on which operator is applied.



3. Formula bar defines the value applied to the property.



In all formula bars it is possible to use not only constant numbers, but also the keywords ("Source", "Target", "Value") and properties. The list of the properties which can be used with keywords are listed in the modding editor for each source/target separately.



Source: Unit	Target: Unit
<p>▼ Properties</p> <ul style="list-style-type: none"> <li>AntiAirCombatStrength</li> <li>AttackRange</li> <li>BaseStrength</li> <li>BombardCenterStrengthBonus</li> <li>BombardSplashStrengthBonus</li> <li>CombatStrength</li> <li>DetectionRange</li> <li>DistrictCombatStrengthBonus</li> <li>ExperienceGainMultiplier</li> <li>HealthRatio</li> <li>HealthRegen</li> <li>HealthRegenAfterBattle</li> <li>HitPoints</li> <li>LandAttackRange</li> <li>LandCombatStrength</li> <li>LandLeaderPriority</li> <li>LandSiegeWorksNet</li> <li>LandSpeed</li> <li>LandVisionRange</li> <li>LeaderPriority</li> <li>LostAtSeaDamageRatio</li> <li>MovementRatio</li> <li>NavalSpeed</li> <li>RansackCombatStrengthBonus</li> <li>SiegeWorksNet</li> <li>SpecialtyLevel</li> <li>SpoilOfWarGains</li> <li>StrikeRadius</li> <li>TransportAttackRange</li> <li>TransportCombatStrength</li> <li>TransportLeaderPriority</li> <li>TransportSiegeWorksNet</li> <li>TransportVisionRange</li> <li>TrespassingDamageRatio</li> <li>UnitsKilled</li> <li>UnitUpkeep</li> <li>VeterancyLevel</li> <li>VisionHeight</li> <li>VisionRange</li> <li>ZoneOfControlDistance</li> </ul>	<p>▼ Properties</p> <ul style="list-style-type: none"> <li>AntiAirCombatStrength</li> <li>AttackRange</li> <li>BaseStrength</li> <li>BombardCenterStrengthBonus</li> <li>BombardSplashStrengthBonus</li> <li>CombatStrength</li> <li>DetectionRange</li> <li>DistrictCombatStrengthBonus</li> <li>ExperienceGainMultiplier</li> <li>HealthRatio</li> <li>HealthRegen</li> <li>HealthRegenAfterBattle</li> <li>HitPoints</li> <li>LandAttackRange</li> <li>LandCombatStrength</li> <li>LandLeaderPriority</li> <li>LandSiegeWorksNet</li> <li>LandSpeed</li> <li>LandVisionRange</li> <li>LeaderPriority</li> <li>LostAtSeaDamageRatio</li> <li>MovementRatio</li> <li>NavalSpeed</li> <li>RansackCombatStrengthBonus</li> <li>SiegeWorksNet</li> <li>SpecialtyLevel</li> <li>SpoilOfWarGains</li> <li>StrikeRadius</li> <li>TransportAttackRange</li> <li>TransportCombatStrength</li> <li>TransportLeaderPriority</li> <li>TransportSiegeWorksNet</li> <li>TransportVisionRange</li> <li>TrespassingDamageRatio</li> <li>UnitsKilled</li> <li>UnitUpkeep</li> <li>VeterancyLevel</li> <li>VisionHeight</li> <li>VisionRange</li> <li>ZoneOfControlDistance</li> </ul>

The shown charge ability is calculated in the following way:

1. It takes the unit specialty level (which is assigned within another descriptor).
2. Multiply specialty by 2.
3. Adds 1 to the previous result.
4. The result is added to the unit combat strength during the charge.

If we check the existing unit (Cuirassiers specialty level is 3) in the battle, we can find a bonus result ( $1+2*3=7$ ).

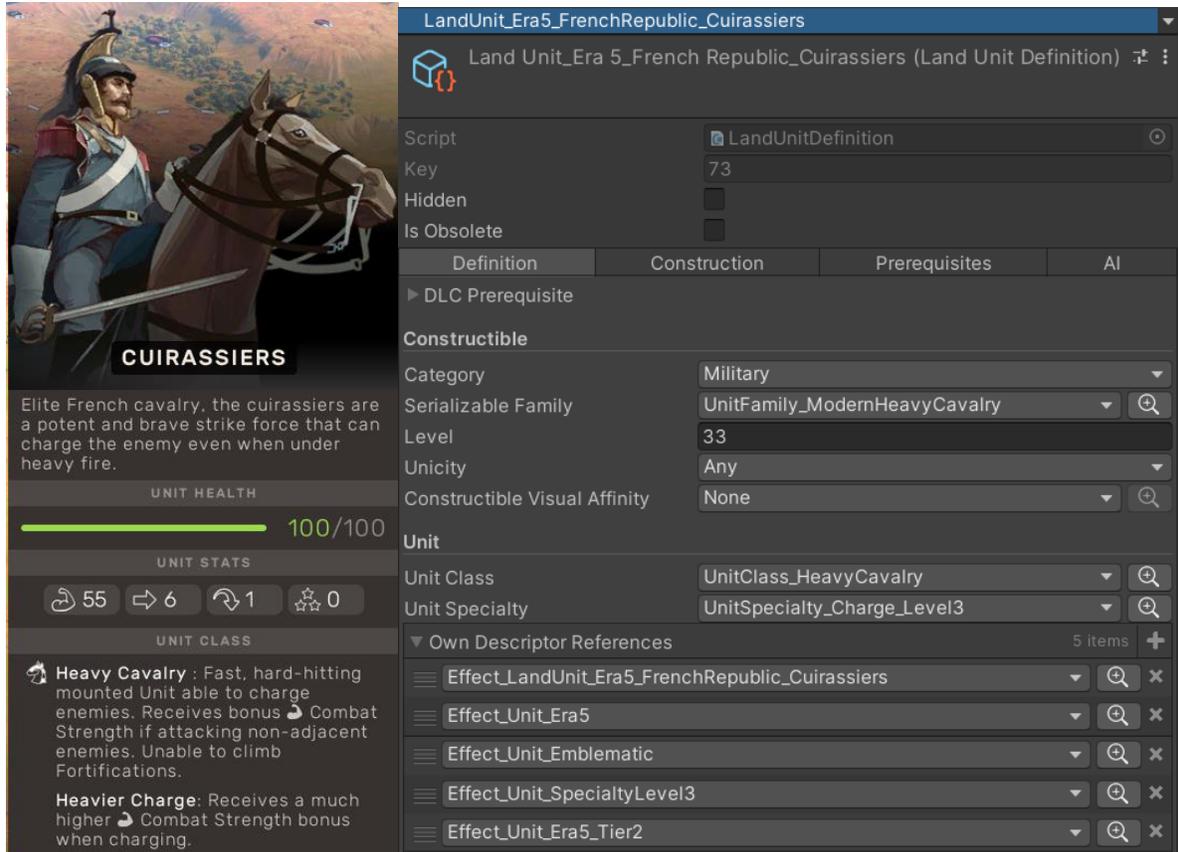


The table with the existing operators:

Operator	Inline	Description
Add	+	adds the number (calculated in the formula bar) to the property.
Sub	-	subtracts a number from the property.
Mult	*	multiplies the property by the number.
Div	/	divides the property by the number.
Pow	^	powers the property by the number.
Percent		takes the percent number of the property.
Max	max	returns max value of the property.
Min	min	returns min value of the property.

## 13.7 Balancing the battle ability

If we want to change the “Heavier Charge” ability of cuirassiers (French unit from the Industrial era), we have 2 main ways to do it:



The screenshot displays the unit editor for "Cuirassiers". The left panel shows a unit card with a health bar at 100/100, unit stats (55, 6, 1, 0), and class description. The right panel shows the unit's configuration, including script, key, and various abilities.

**UNIT HEALTH**  
100/100

**UNIT STATS**  
55 6 1 0

**UNIT CLASS**  
**Heavy Cavalry** : Fast, hard-hitting mounted Unit able to charge enemies. Receives bonus Combat Strength if attacking non-adjacent enemies. Unable to climb Fortifications.  
**Heavier Charge**: Receives a much higher Combat Strength bonus when charging.

**LandUnit\_Era5\_FrenchRepublic\_Cuirassiers**  
Land Unit\_Era 5\_French Republic\_Cuirassiers (Land Unit Definition)

Script: LandUnitDefinition  
Key: 73  
Hidden:   
Is Obsolete:

Definition Construction Prerequisites AI

► DLC Prerequisite

**Constructible**

Category: Military  
Serializable Family: UnitFamily\_ModernHeavyCavalry  
Level: 33  
Unicity: Any  
Constructible Visual Affinity: None

**Unit**

Unit Class: UnitClass\_HeavyCavalry  
Unit Specialty: UnitSpecialty\_Charge\_Level3

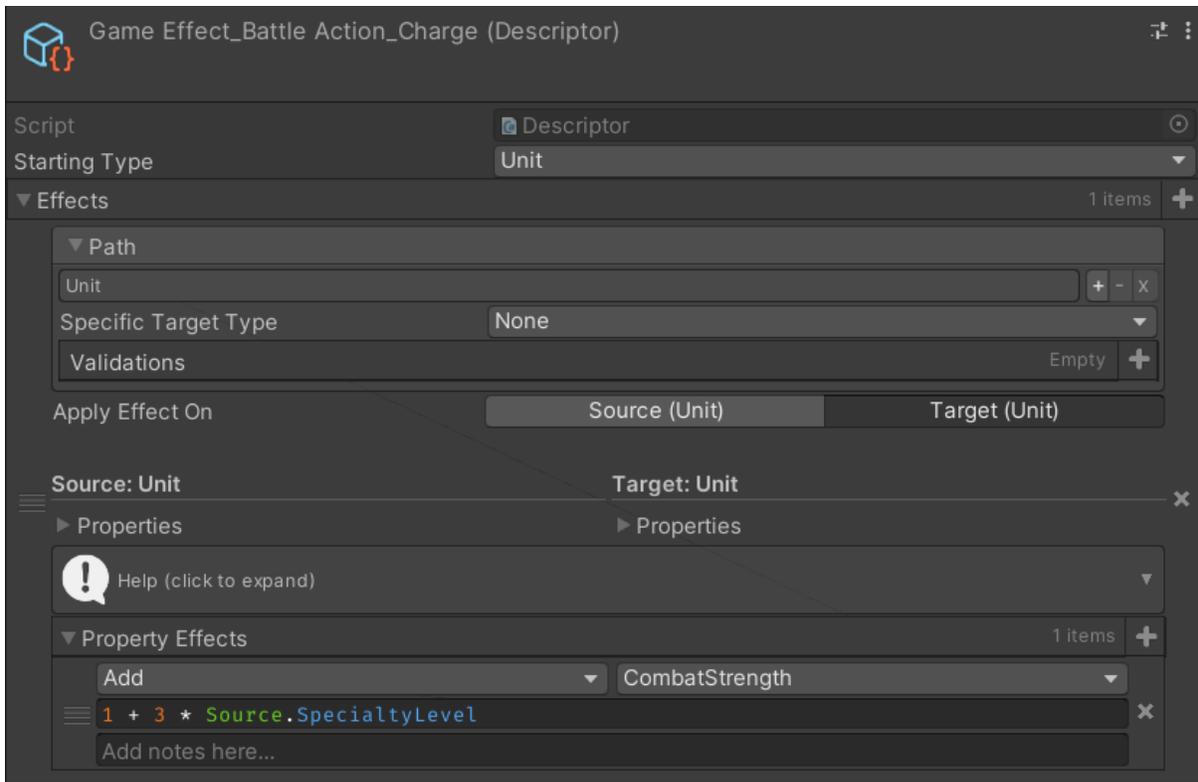
▼ Own Descriptor References (5 items)

- Effect\_LandUnit\_Era5\_FrenchRepublic\_Cuirassiers
- Effect\_Unit\_Era5
- Effect\_Unit\_Emblematic
- Effect\_Unit\_SpecialtyLevel3
- Effect\_Unit\_Era5\_Tier2

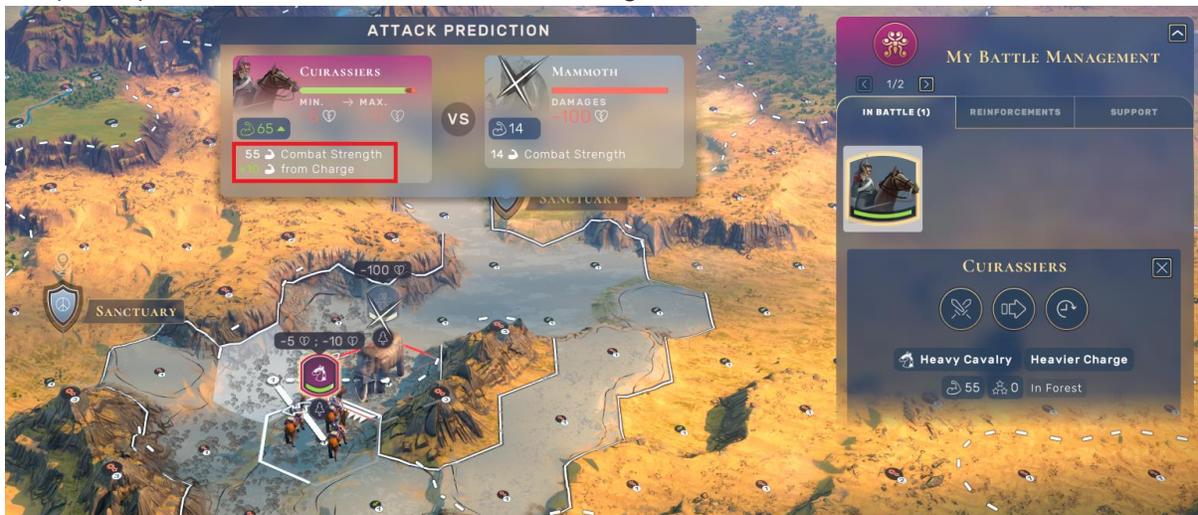
1. Rebalance the charge ability of all units.
2. Rebalance the cuirassiers' specialty to which the charge ability is referred.

### 13.7.1 Rebalancing the charge ability

Open the charge ability and change the calculated formula to the preferred one:



Now build the mod by clicking “Build and Run” and start a new game. The charge bonus is 10 ( $1+3*3$ ) now, because the formula was changed.



### 13.7.2 Rebalancing the unit specialty

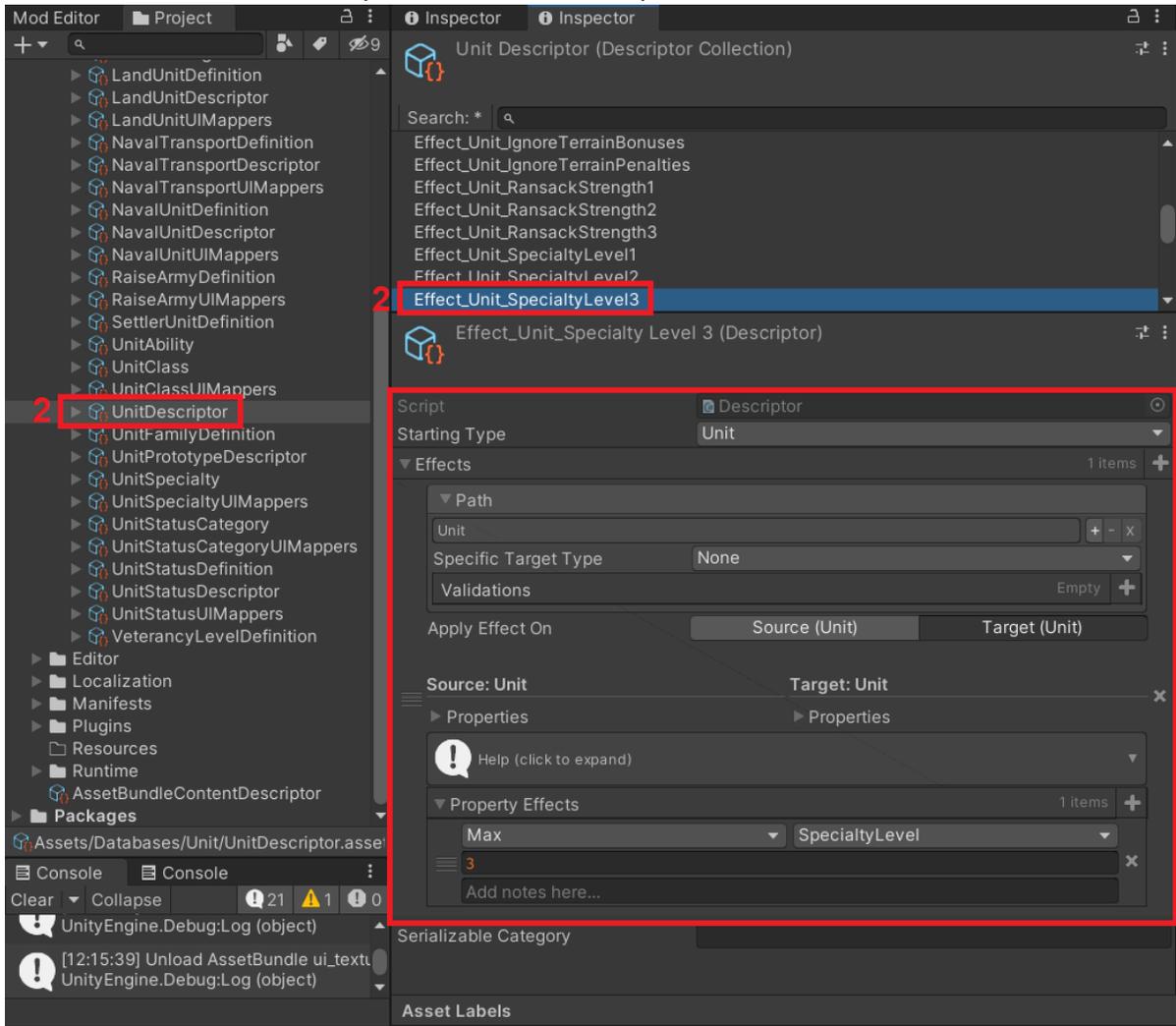
To modify the strength of the ability, the assigned descriptor must be changed.

1. Find the necessary unit's ability and its "Descriptor references".

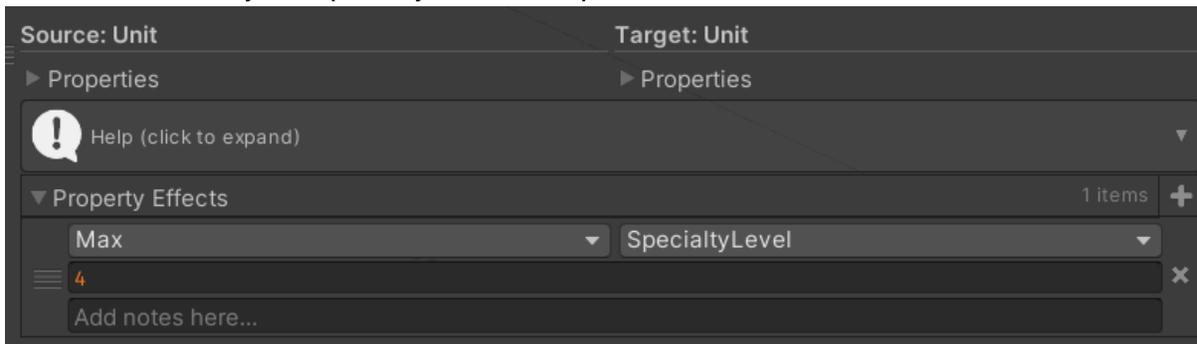
The screenshot shows the configuration panel for the ability 'UnitAbility\_Charge\_Level3'. The interface is divided into several sections:

- Script:** Set to 'UnitAbility'.
- Tags as Ability:** 'Tag As Ability' is set to 'None'.
- Pathfind:** 'Movement Ability' and 'Pathfinding Flags' are both set to 'None'.
- Simulation:** A list of 'Descriptor References' is shown, containing one item: 'Effect\_Unit\_SpecialtyLevel3'. This item is highlighted with a red box and a red '1' next to it.
- Battle:** A list of 'Battle Ability Reference' is shown, containing one item: 'BattleAbility\_Charge'.

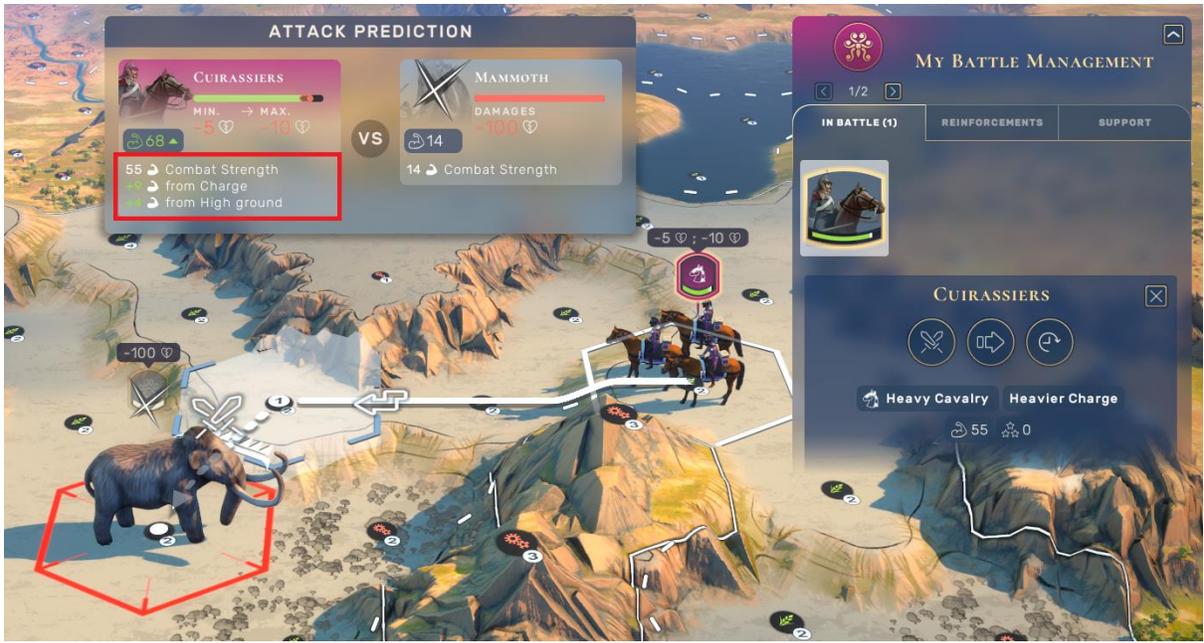
2. Find this descriptor in the “UnitDescriptor” collection.



3. Modify the specialty level as required.



Now build the mod by clicking “Build and Run” and start a new game. The charge bonus is 9 now (1+2\*4), because the specialty was changed.



# 14 Useful data

Table of resource names:

<b>Database Asset Name</b>	<b>Ingame Name</b>
ResourceDeposit01	Horse
ResourceDeposit02	Copper
ResourceDeposit03	Iron
ResourceDeposit04	Coal
ResourceDeposit05	Salt peter
ResourceDeposit06	Oil
ResourceDeposit07	Aluminum
ResourceDeposit08	Uranium
ResourceDeposit11	Salt
ResourceDeposit12	Sage
ResourceDeposit13	Coffee
ResourceDeposit14	Tea
ResourceDeposit15	Saffron
ResourceDeposit16	Dye
ResourceDeposit17	Ebony
ResourceDeposit18	Marble
ResourceDeposit19	Obsidian
ResourceDeposit20	Silk
ResourceDeposit21	Incense
ResourceDeposit22	Porcelain
ResourceDeposit23	Pearls
ResourceDeposit24	Gold
ResourceDeposit25	Gemstone
ResourceDeposit26	Ambergris
ResourceDeposit27	Papyrus
ResourceDeposit28	Lead
ResourceDeposit29	Mercury
ResourceDeposit30	Silver
ResourceDeposit31	Weapons